Batch ERC20 Transfer with Session Router

Let's execute a batch of user operations using session keys by leveraging the Batched Session Validation Module

Create new component

Let's Create a new component calledBatchERC20Transfer.tsx and place it in the components folder.

The imports will be as follows:

import

React

from

"react" ; import

{ ethers }

from

"ethers" ; import

{

BiconomySmartAccountV2 ,

DEFAULT_SESSION_KEY_MANAGER_MODULE ,

DEFAULT_BATCHED_SESSION_ROUTER_MODULE , createSessionKeyManagerModule , createBatchedSessionRouterModule }

from

"@biconomy/account" import

usdcAbi

from

"@/utils/usdcAbi.json" import

{ toast }

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 BatchERC20Transfer : React . FC < props

=

({ smartAccount , provider , address })

=>

{ return (< button

Batch Transfer 1

USDC < / button

) }

export

default BatchERC20Transfer ; This is going to be a basic button that simply transfers 1 USDC to two different recipients.

{ isSessionActive &&

(< BatchERC20Transfer smartAccount = { smartAccount } provider = { provider } address = { address } /

) ; }

Create transfer function

Let's create the function now to handle the transfer:

const

batchErc20Transfer

=

async

( )

=>

{ if

(! address ||

! smartAccount ||

! address)

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

{ toast . info ('Transferring 1 USDC to two recipients...' ,

{ position :

"top-right" , autoClose :

15000 , hideProgressBar :

false , closeOnClick :

true , pauseOnHover :

true , draggable :

true , progress :

undefined , theme :

"dark" , }) ; const erc20ModuleAddr =

"0x000000D50C68705bd6897B2d17c7de32FB519fDA" ; const mockSessionModuleAddr =

"0x7Ba4a7338D7A90dfA465cF975Cc6691812C3772E" ; // 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) ; console . log ("sessionSigner" , sessionSigner) ;

// generate sessionModule const sessionModule =

```
await

createSessionKeyManagerModule ( { moduleAddress :

DEFAULT_SESSION_KEY_MANAGER_MODULE , smartAccountAddress : address , } ) ;

// genereate batched session router module const sessionRouterModule =

await

createBatchedSessionRouterModule ( { moduleAddress :

DEFAULT_BATCHED_SESSION_ROUTER_MODULE , sessionKeyManagerModule : sessionModule ,
smartAccountAddress : address , } ) ;

// set active module to session router module smartAccount = smartAccount . setActiveValidationModule (
sessionRouterModule ) ;

const tokenContract =

new

ethers . Contract ( // polygon mumbai usdc address "0xdA5289fCAAF71d52a80A254da614a192b693e977" , usdcAbi ,
provider ) ; let decimals =

18 ;

try

{ decimals =

await tokenContract . decimals ( ) ; }

catch

( error )

{ throw

new

Error ( "invalid token address supplied" ) ; }

const

{ data : data1 }

=

await tokenContract . populateTransaction . transfer ( "0x322Af0da66D00be980C7aa006377FCaaEee3BDFD" ,

// receiver address ethers . utils . parseUnits ( "1" , decimals ) ) ;

const

{ data : data2 }

=

await tokenContract . populateTransaction . transfer ( "0xFA66E705cf2582cF56528386Bb9dFCA119767262" ,

// receiver address ethers . utils . parseUnits ( "1" , decimals ) ) ;

// generate tx data to first erc20 transfer const tx1 =

{ to :

"0xdA5289fCAAF71d52a80A254da614a192b693e977" ,

//erc20 token address data : data1 , value :

"0" , } ;

// generate tx data to second erc20 transfer const tx2 =
```

{ to :

"0xdA5289fCAAF71d52a80A254da614a192b693e977" ,

//erc20 token address data : data2 , value :

"0" , } ;

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

await smartAccount . sendTransaction ([tx1 , tx2] ,

{ params :

{ batchSessionParams :

[{ sessionSigner : sessionSigner , sessionValidationModule : erc20ModuleAddr , } , { sessionSigner : sessionSigner , sessionValidationModule : mockSessionModuleAddr , } ,] , } , }) ;

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) ; } } Add this method to the onClick event of the button.

return(batchErc20Transfer}>Batch Transfer 1 USDC) Congrats! you have successfully integrated the batched session router module. Expand the code below to see the entire code:

Details import React from

"react" ; import

{ ethers }

from

"ethers" ; import

{ BiconomySmartAccountV2 ,

DEFAULT_SESSION_KEY_MANAGER_MODULE ,

DEFAULT_BATCHED_SESSION_ROUTER_MODULE , createSessionKeyManagerModule , createBatchedSessionRouterModule }

from

"@biconomy/account" import usdcAbi from

"@/utils/usdcAbi.json" import

{ toast }

from

'react-toastify' ; import

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

interface

props

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

string ; }

const BatchERC20Transfer : React . FC < props

=

({ smartAccount , provider , address })

=>

{

const

batchErc20Transfer

=

async

( )

=>

{ if

(! address ||

! smartAccount ||

! address)

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

{ toast . info ('Transferring 1 USDC to two recipients...' ,

{ position :

"top-right" , autoClose :

15000 , hideProgressBar :

false , closeOnClick :

true , pauseOnHover :

true , draggable :

true , progress :

undefined , theme :

"dark" , }) ; const erc20ModuleAddr =

"0x000000D50C68705bd6897B2d17c7de32FB519fDA" ; const mockSessionModuleAddr =

"0x7Ba4a7338D7A90dfA465cF975Cc6691812C3772E" ; // 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) ; console . log ("sessionSigner" , sessionSigner) ;

// generate sessionModule const sessionModule =

await

createSessionKeyManagerModule ({ moduleAddress :

DEFAULT_SESSION_KEY_MANAGER_MODULE , smartAccountAddress : address , }) ;

// genereate batched session router module const sessionRouterModule =

await

createBatchedSessionRouterModule ({ moduleAddress :

DEFAULT_BATCHED_SESSION_ROUTER_MODULE , sessionKeyManagerModule : sessionModule , smartAccountAddress : address , }) ;

// set active module to session router module smartAccount = smartAccount . setActiveValidationModule (sessionRouterModule) ;

const tokenContract =

new

ethers . Contract (// polygon mumbai usdc address "0xdA5289fCAAF71d52a80A254da614a192b693e977" , usdcAbi , provider) ; let decimals =

18 ;

try

{ decimals =

await tokenContract . decimals ( ) ; }

catch

(error)

{ throw

new

Error ("invalid token address supplied") ; }

const

{ data : data1 }

=

await tokenContract . populateTransaction . transfer ("0x322Af0da66D00be980C7aa006377FCaaEee3BDFD" ,

// receiver address ethers . utils . parseUnits ("1" , decimals)) ;

const

```
{ data : data2 }

=

await tokenContract . populateTransaction . transfer ( "0xFA66E705cf2582cF56528386Bb9dFCA119767262" ,

// receiver address ethers . utils . parseUnits ( "1" , decimals ) ) ;

// generate tx data to first erc20 transfer const tx1 =

{ to :

"0xdA5289fCAAF71d52a80A254da614a192b693e977" ,

//erc20 token address data : data1 , value :

"0" , } ;

// generate tx data to second erc20 transfer const tx2 =

{ to :

"0xdA5289fCAAF71d52a80A254da614a192b693e977" ,

//erc20 token address data : data2 , value :

"0" , } ;

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

await smartAccount . sendTransaction ( [ tx1 , tx2 ] ,

{ params :

{ batchSessionParams :

[ { sessionSigner : sessionSigner , sessionValidationModule : erc20ModuleAddr , } , { sessionSigner : sessionSigner ,
sessionValidationModule : mockSessionModuleAddr , } , ] , } , } ) ;

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 = { ( )

=> batchErc20Transfer }

Batch Transfer 1

USDC < / button

) }

export

default BatchERC20Transfer ; Previous Create Session Router Next Custom Session Storage Tutorial