useCreateBatchSession

@biconomy/use-aa •Hook

@biconomy/use-aa / useCreateBatchSession

Description

Creates a sessions to be used when submitting batches of txs simultaneously in the context of a users smart account.

Parameters

type

SessionEpoch

=

{ /* *The time at which the session is no longer valid*/ validUntil ? :

number ; /* *The time at which the session becomes valid*/ validAfter ? :

number ; } ;

type

Policy

=

{ / **The address of the contract to be included in the policy** */ **contractAddress : Hex ;** /The address of the sessionKey upon which the policy is to be imparted */ *sessionKeyAddress : Hex ; /The specific function selector from the contract to be included in the policy* / functionSelector :

string

| AbiFunction ; /**The rules to be included in the policy** */ **rules : Rule [ ] ;** / The time interval within which the session is valid. If left unset the session will remain invalid indefinitely */ *interval ? : SessionEpoch ; /The maximum value that can be transferred in a single transaction* / valueLimit : bigint ; } ;

type

UseCreateBatchSessionProps

=

{ / **The array of policy elements to be applied to the session.** */ **policy : Policy [ ] ;** /The BuildUserOpOptions options. See https://bcnmy.github.io/biconomy-client-sdk/types/BuildUserOpOptions.html for further detail */ options ? : BuildUserOpOptions ; } ;

Returns

userOpReceipt

type

UserOpResponse

=

{ userOpHash :

string ; wait (_confirmations ? :

number) :

Promise < UserOpReceipt

; waitForTxHash ( ) :

Promise < UserOpStatus

; } ;

Example

import

{ useCreateBatchSession , useUserOpWait ,

Options

}

from

"@biconomy/useAA" ; import

{ polygonAmoy }

from

"viem/chains" ; import

{ encodeFunctionData , parseAbi }

from

"wagmi" ;

const

CreateBatchSession

=

({ userSmartAccountAddress })

=>

{ const leafPolicy :

Policy

=

{ interval :

{ validUntil :

0 , validAfter :

0 , } , contractAddress :

"0x1758f42Af7026fBbB559Dc60EcE0De3ef81f665e" , functionSelector :

"safeMint(address)" , rules :

[{ offset :

0 , condition :

0 , referenceValue : userSmartAccountAddress , } ,] , valueLimit :

0n , } ;

const policyLeaves :

Policy [ ]

=

[leafPolicy , leafPolicy] ;

```
const
{ mutate , data : userOpResponse , error , isPending , }
=
useCreateBatchSession ( ) ;
const
{ isLoading : waitIsLoading , isSuccess : waitIsSuccess , error : waitError , data : waitData , }
=
useUserOpWait ( userOpResponse ) ;
const
grantPermission
=
( )
=> mutate ( { policy : policyLeaves , options :
Options . Sponsored , } ) ;
useEffect ( ( )
=>
{ if
( waitIsSuccess && waitData ?. success ===
"true" )
{ console . log ( "Successful mint: "
+ { polygonAmoy . blockExplorers . default . url } /tx/ { waitData ?. receipt ?. transactionHash }  ) ; } } ,
[ waitIsSuccess ] ) ;
return
( < ErrorGuard
```

errors

```
{ [ error , waitError ] }
    < Button title = " Grant Permission " onClickFunc = { grantPermission } isLoading = { isPending || waitIsLoading }
    /> </ ErrorGuard
    ) ; } ;
```

Source

hooks/useCreateBatchSession.ts:99 Previous useBatchSession Next useCreateSession