useCreateSession

Description

Creates a session to be used when submitting tx 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{ /* *The time at which the session is no longer valid*/ validUntil ?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 ;** *T*he 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

UseCreateSessionProps

=

{ / **The array of policy elements to be applied to the session.** */ **policy : Policy [ ] ;** *T*he 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

{ useCreateSession , useUserOpWait ,

Options

}

from

"@biconomy/useAA" ; import

{ polygonAmoy }

from

"viem/chains" ; import

{ encodeFunctionData , parseAbi }

from

"wagmi" ;

const

CreateSession

=

( { userSmartAccountAddress } )

=>

{ const policy =

[ { contractAddress :

"0x1758f42Af7026fBbB559Dc60EcE0De3ef81f665e" , functionSelector :

"safeMint(address)" , rules :

[ { offset :

0 , condition :

0 , referenceValue : userSmartAccountAddress , } , ] , interval :

{ validUntil :

0 , validAfter :

0 , } , valueLimit :

0n , } , ] ;

const

{ mutate , data : userOpResponse , error , isPending }

=

useCreateSession ( ) ;

const

{ isLoading : waitIsLoading , isSuccess : waitIsSuccess , error : waitError , data : waitData , }
```

```
=

useUserOpWait ( userOpResponse ) ;

useEffect ( ( )

=>

{ if

( waitIsSuccess && waitData ?. success ===

"true" )

{ console . log ( "Successful mint: "

+ { polygonAmoy . blockExplorers . default . url } /tx/ { waitData ?. receipt ?. transactionHash }   ) ; } } ,

[ waitIsSuccess ] ) ;

const

createSessionHandler

=

( )

=> mutate ( { policy , options :

Options . Sponsored , } ) ;

return

( < ErrorGuard
```

errors

```
{ [ error , waitError ] }
```

```
    < Button title = " Create a session " onClickFunc = { createSessionHandler } isLoading = { isPending ||
    waitIsLoading } /> </ ErrorGuard

    ) ; } ;
```

Source