useSendGasTokenTransaction

@biconomy/use-aa •Hook

@biconomy/use-aa / useSendGasTokenTransaction

Description

Sends a transaction, taking an erc20 token of choice as payment for the gas. token can be specified in advance by providing the tokenParam as a Hex address, or user can set the token dynamically by setting the tokenParam to afeeQuote

Parameters

/* *Response from smartAccount.getTokenFees() method*/ type

PaymasterFeeQuote

=

{ /* *symbol: Token symbol*/ symbol :

string ; /* *tokenAddress: Token address*/ tokenAddress :

string ; /* *decimal: Token decimal*/ decimal :

number ; logoUrl ? :

string ; /* *maxGasFee: in wei*/ maxGasFee :

number ; /* *maxGasFee: in dollars*/ maxGasFeeUSD ? :

number ; usdPayment ? :

number ; /* *The premium paid on the token*/ premiumPercentage :

number ; /* *validUntil: Unix timestamp*/ validUntil ? :

number ; } ;

type

UseSendGasTokenTransactionProps

=

{ / **The transaction or transactions to send */ transactions : Transaction** | **Transaction [ ] ;** /The preferred token to use for gas payment, or a feeQuote */ tokenParam : Hex* | *PaymasterFeeQuote ; /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

{ useSendGasTokenTransaction , useUserOpWait , Options , useSmartAccount , mergeOptions , }

from

"@biconomy/useAA" ; import

{ polygonAmoy }

from

"viem/chains" ; import

{ encodeFunctionData , parseAbi }

from

"wagmi" ;

const

SendGasTokenTx

=

( )

=>

{ const

{ smartAccountAddress }

=

useSmartAccount ( ) ;

const

{ mutate , data : userOpResponse , error , isPending , }

=

useSendGasTokenTransaction ( ) ;

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 transactions =

useMemo (( )

=>

({ to :

"0x1758f42Af7026fBbB559Dc60EcE0De3ef81f665e" , data :

encodeFunctionData ({ abi :

parseAbi (["function safeMint(address _to)"]) , functionName :

"safeMint" , args :

[smartAccountAddress as

Hex] , }) , }) , [smartAccountAddress]) ;

const

{ data : feeQuoteResponse , isLoading : isLoadingFee }

=

useTokenFees ({ transactions , }) ;

const

mintNftTx

=

( )

=>

mutate ({ transactions , tokenParam : feeQuotes [0]

}) ;

// OR if you know in advance which token you'd like to use

const

alternativeMintTx

=

( )

=> mutate ({ transactions , tokenParam :

"0xf555F8d9Cf90f9d95D34488e6C852796D9acBd31" , }) ;

return

(< ErrorGuard

errors

{ [error , waitError] }

< Button title = " Mint NFT " onClickFunc = { mintNftTx } isLoading = { isPending || waitIsLoading } /> </ ErrorGuard

) ; } ;

Source

hooks/useSendGasTokenTransaction.ts:96 Previous useGasEstimate Next useSendSponsoredTransaction