useTokenFees

@biconomy/use-aa •Hook

@biconomy/use-aa / useTokenFees

Description

This function will retrieve fees from the paymaster in erc20 mode

Parameters

type

Transaction

=

{ to :

string ; value : BigNumberish |

string ; data :

string ; } ;

type

UseTokenFeesProps

=

{ / **The transactions to be batched. */ transactions : Transaction | Transaction [ ] ;** /The BuildUserOpOptions options. See https://bcnmy.github.io/biconomy-client-sdk/types/BuildUserOpOptions.html for further detail */ options ? : BuildUserOpOptions ; } ;

Returns

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

FeeQuotesOrDataResponse

=

{ / **Array of results from the paymaster */ feeQuotes ? : PaymasterFeeQuote [ ] ;** /Normally set to the spender in the proceeding call to send the tx */ tokenPaymasterAddress ? : Hex ; /Relevant Data returned from the paymaster/* paymasterAndData ? : Uint8Array | Hex ; preVerificationGas ? : BigNumberish ; verificationGasLimit ? : BigNumberish ; callGasLimit ? : BigNumberish ; } ;

Example

import

{ useTokenFees , useUserOpWait ,

Options

}

from

"@biconomy/useAA" import

{ polygonAmoy }

from

"viem/chains" import

{ encodeFunctionData , parseAbi }

from

"wagmi"

export

const

TokenFees

=

( )

=>

{

const

{ smartAccountAddress }

=

useSmartAccount ( ) ;

const mintTx :

Transaction

=

{ to :

"0x1758f42Af7026fBbB559Dc60EcE0De3ef81f665e" , data :

encodeFunctionData ({ abi :

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

"safeMint" , args :

[smartAccountAddress] , }) , }

const

{ mutate , data : feeData , error , isLoading , }

=

useTokenFees ({ transactions : mintTx }) ;

return

(< ErrorGuard

errors

{ [error] }

    < div

    { isLoading ?

(< div

    Loading... </ div

)

:

(< div

    { feeData ?. feeQuotes . map ((quote)

=>

(< div

key

{ quote . token }

    { quote . token } : { quote . amount } </ div

)) } </ div

) } } </ div

    </ ErrorGuard

) ; } ;

Source