Implementing Fee Currency in Wallets

Celo allows paying gas fees in currency other than the native currency. The tokens that can be used to pay gas fees is controlled via governance and the list of tokens allowed is maintained inFeeCurrencyWhitelist.sol .

Alternate fee currency works with EOAs and no paymaster is required!

This works by specifying a token/adapter address as a value for thefeeCurrency property in the transaction object. ThefeeCurrency property in the transaction object is exclusive to Celo and allows paying gas fees using assets other than the native currency of the network.

The below steps describe how wallets can implement the alternate fee currency feature in order to allow users to use alternate assets in the user's wallet to pay for gas fees.

Enabling Transactions with ERC20 Token as Fee Currency

We recommend using theviem library as it has support for thefeeCurrency field in the transaction required for sending transaction where the gas fees will be paid in ERC20 tokens.

Estimating gas price

To estimate gas price use the token address (in case of cUSD, cEUR and cREAL) or the adapter address (in case of USDC and USDT) as the value for feeCurrency field in the transaction.

Estimating gas price is important because if the user is trying to transfer the entire balance in an asset and using the same asset to pay for gas fees, the user shouldn't be able to transfer the entire amount as a small portion will be utilized to pay for gas fees.

Example: If the user has 10 USDC and is trying to transfer the entire 10 USDC and chooses to use USDC as the currency to pay for gas, the user shouldn't be allowed to transfer the entire 10 USDC as a small portion has to be used for gas fees.

The following code snippet calculates the transaction fee (in USDC) for a USDC transfer transaction.

Feel free to modify to support the currency of your choice.

import

{ createPublicClient , hexToBigInt , http }

from

"viem" ; import

{ celo }

from

"viem/chains" ;

// USDC is 6 decimals and hence requires the adapter address instead of the token address const

USDC_ADAPTER_MAINNET

=

"0x2F25deB3848C207fc8E0c34035B3Ba7fC157602B" ;

const publicClient =

createPublicClient ({ chain : celo , transport :

http ( ) , }) ;

const transaction =

{ from :

"0xccc9576F841de93Cd32bEe7B98fE8B9BD3070e3D" , to :

"0xcebA9300f2b948710d2653dD7B07f33A8B32118C" , data :

"0xa9059cbb000000000000000000000000ccc9576f841de93cd32bee7b98fe8b9bd3070e3d0000000000000000000000000000000000000000000000000000000000000000f4240" , feeCurrency :

USDC_ADAPTER_MAINNET , } ;

async

function

getGasPriceInUSDC ( )

{ const priceHex =

await publicClient . request ({ method :

"eth_gasPrice" , params :

[USDC_ADAPTER_MAINNET] , }) ;

return

hexToBigInt (priceHex) ; }

async

function

estimateGasPriceInUSDC (transaction)

{ const estimatedGasInHex =

await publicClient . estimateGas ({ ... transaction ,

// just in case the transaction itself does not have feeCurrency property in it. feeCurrency :

USDC_ADAPTER_MAINNET , }) ;

return

hexToBigInt (estimatedGasInHex) ; }

async

function

main ( )

{ const gasPriceInUSDC =

await

getGasPriceInUSDC ( ) ;

const estimatedGasPrice =

await

estimateGasInUSDC (transaction) ;

/ *Transaction fee in USDC to perform the above transaction. This amount should not be transferrable in case the user tries to transfer the entire amount.*const transactionFeeInUSDC =

formatEther (gasPriceInUSDC * estimatedGasPrice ,) . toString ( ) ;

return transactionFeeInUSDC ; }

Sending transaction using alternate fee currency

Sending transaction with fee currency other than the native currency of the network is pretty straightforward all you need to do is set thefeeCurrency property in the transaction object with the address of the token/adapter you want to use to pay for gas fees.

The below code snippets demonstrates transferring 1 USDC using USDC as gas currency.

import

{ createWalletClient , http }

from

"viem" ; import

{ celo }

from

"viem/chains" ; import

{ privateKeyToAccount }

from

"viem/accounts" ; import

{ stableTokenAbi }

from

"@celo/abis" ;

// Creating account from private key, you can choose to do it any other way. const account =

privateKeyToAccount ("0x432c...") ;

// WalletClient can perform transactions. const client =

createWalletClient ({ account ,

// Passing chain is how viem knows to try serializing tx as cip42. chain : celo , transport :

http ( ) , }) ;

const

USDC_ADAPTER_MAINNET

=

"0x2F25deB3848C207fc8E0c34035B3Ba7fC157602B" ; const

USDC_MAINNET

=

"0xcebA9300f2b948710d2653dD7B07f33A8B32118C" ;

/ *The UI of the wallet should calculate the transaction fees, show it and consider the amount to not be part of the asset that the user i.e the amount corresponding to transaction fees should not be transferrable.* / async

function

calculateTransactionFeesInUSDC (transaction)

{ // Implementation of getGasPriceInUSDC is in the above code snippet const gasPriceInUSDC =

await

getGasPriceInUSDC ( ) ;

// Implementation of estimateGasInUSDC is in the above code snippet const estimatedGasPrice =

await

estimateGasInUSDC (transaction) ;

return gasPriceInUSDC * estimatedGasPrice ; }

async

function

send (amountInBigInt)

{ const to =

USDC_MAINNET ;

// Data to perform an ERC20 transfer const data =

encodeAbiParameters ([{

name :

"to" ,

type :

"address"

} , {

name :

"amount" ,

type :

"uint256"

} ,] , ["0xccc9576F841de93Cd32bEe7B98fE8B9BD3070e3D" ,

// USDC receiver address amount ,] ,) ;

const transactionFee =

await

calculateTransactionFeesInUSDC ({ to , data }) ;

const tokenReceivedbyReceiver =

parseEther ("1")

- transactionFee ;

/ *Now the data has to be encode again but with different transfer value because the receiver receives the amount minus the transaction fee*.const
dataAfterFeeCalculation =

encodeAbiParameters ([{

name :

"to" ,

type :

"address"

} , {

name :

"amount" ,

type :

"uint256"

} ,] , ["0xccc9576F841de93Cd32bEe7B98fE8B9BD3070e3D" ,

// USDC receiver address tokenReceivedbyReceiver ,] ,) ;

// Transaction hash const hash =

await client . sendTransaction ({ ... { to ,

data : dataAfterFeeCalculation } ,

/* In case the transaction request does not include the feeCurrency property, the wallet can add it or change it to a different currency based on the user balance.

Notice that we use the USDC_ADAPTER_MAINNET address not the token address this is because at the protocol level only 18 decimals tokens are supported, but USDC is 6 decimals, the adapter acts a unit converter. */ feeCurrency :

USDC_ADAPTER_MAINNET , }) ;

return hash ; } If you have any questions, please reach out here . Edit this page Previous MiniPay Deeplinks Next Celo Bridges