How to conditionally sponsor a user operation

The permissionless.js library allows you to conditionally sponsor a user operation. This can be useful in situations like:

- Sponsor only first 10 transactions
- Sponsor if they have a specific NFT

The use cases can be many, but the general idea is to use themiddleware.sponsorUserOperation parameter to conditionally sponsor a user operation.

Let's take an example where you want to sponsor only the first 10 transactions.

Create the clients

First we must create the public, (optionally) pimlico paymaster clients that will be used to interact with the SimpleAccount.

```

import{ createPublicClient, http }from"viem" import{ decodeNonce }from"permissionless/utils" import{ createPimlicoClient }from"permissionless/clients/pimlico" import{ entryPoint07Address }from"viem/account-abstraction"

constpublicClient=createPublicClient({ chain: sepolia, transport:http("https://rpc.ankr.com/eth_sepolia"), })

constpimlicoUrl=https://api.pimlico.io/v2/sepolia/rpc?apikey={apiKey}

constpimlicoClient=createPimlicoClient({ transport:http(pimlicoUrl), entryPoint: { address: entryPoint07Address, version:"0.7", }, })

```

Create an account

Now, create an account. This can any of the accounts supported by permissionless.js or custom accounts conforming to the interface. For this example, we'll use a Simple account.

```

import{ toSimpleSmartAccount }from"permissionless/accounts" import{ generatePrivateKey, privateKeyToAccount }from"viem/accounts"

exportconstsimpleSmartAccount=awaittoSimpleSmartAccount({ client: publicClient, owner:privateKeyToAccount(generatePrivateKey()), entryPoint: { address: entryPoint07Address, version:"0.7", }, })

```

Create the smart account client with the custom paymaster logic

When creating thesmartAccountClient , we can pass in amiddleware.sponsorUserOperation function that will be called before a user operation is signed and sent. This function can conditionally choose to add a paymaster to the user operation if the desired conditions are met.

```

import{ sepolia }from"viem/chains" import{ createSmartAccountClient }from"permissionless" import{ getAccountNonce }from"permissionless/actions"

constsmartAccountClient=createSmartAccountClient({ client: publicClient, account: simpleSmartAccount, chain: sepolia, bundlerTransport:http(pimlicoUrl), paymaster: { asyncgetPaymasterData(parameters) { constnonce=awaitgetAccountNonce(publicClient, { address: parameters.sender, entryPointAddress: parameters.entryPointAddress, })

const{sequence}=decodeNonce(nonce)

if(sequence<10) { // sponsor it for the user as the sequence is less than 10 // sequence is the number of transactions sent by the user for a given key // by default key is 0x0 returnpimlicoClient.getPaymasterData(parameters) }

return{ paymaster:"0x", paymasterData:"0x", paymasterVerificationGasLimit:0n, paymasterPostOpGasLimit:0n, } }, asyncgetPaymasterStubData(parameters) { constnonce=awaitgetAccountNonce(publicClient, { address: parameters.sender, entryPointAddress: parameters.entryPointAddress, })
```

const{sequence}=decodeNonce(nonce)

if(sequence<10) { // sponsor it for the user as the sequence is less than 10 // sequence is the number of transactions sent by the user for a given key // by default key is 0x0 returnpimlicoClient.getPaymasterStubData(parameters) }

return{ paymaster:"0x", paymasterData:"0x", paymasterVerificationGasLimit:0n, paymasterPostOpGasLimit:0n, } }, }, userOperation: { estimateFeesPerGas:async()=>{ return(awaitpimlicoClient.getUserOperationGasPrice()).fast }, }, })

```
```