How to use a custom paymaster with accounts in permissionless.js

permissionless.js allows you to define custom logic to select the paymaster (if any) to use with the accounts you control. This allows you to use your own paymaster or even use multiple paymasters in different scenarios.

Create the clients

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

```
exportconstpublicClient=createPublicClient({ transport:http("https://rpc.ankr.com/eth_sepolia"), entryPoint:ENTRYPOINT_ADDRESS_V07 });

exportconstbundlerClient=createPimlicoBundlerClient({ transport:http( "https://api.pimlico.io/v2/sepolia/rpc?apikey=API_KEY", ), entryPoint:ENTRYPOINT_ADDRESS_V07 });

exportconstpaymasterClient=createPimlicoPaymasterClient({ transport:http( "https://api.pimlico.io/v2/sepolia/rpc?apikey=API_KEY", ), entryPoint:ENTRYPOINT_ADDRESS_V07 });
```

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.

```
constsimpleAccount=awaitprivateKeyToSimpleSmartAccount(publicClient, { privateKey:"0xPRIVATE_KEY", factoryAddress:"0x91E60e0613810449d098b0b5Ec8b51A0FE8c8985", entryPoint:ENTRYPOINT_ADDRESS_V07 });
```

Create the smart account client with the custom paymaster logic

When creating thesmartAccountClient , we can pass in asponsorUserOperation function that will be called before a user operation is signed and sent. This function must return thepaymasterAndData ,preVerificationGas ,verificationGasLimit , andcallGasLimit fields.

Pimlico Paymaster

ERC-20 Paymaster Dynamic Paymaster Logic

```
Pimlico Paymaster constsmartAccountClient=createSmartAccountClient({ account: simpleAccount, entryPoint:ENTRYPOINT_ADDRESS_V07, chain: sepolia, bundlerTransport:http( "https://api.pimlico.io/v2/sepolia/rpc?apikey=API_KEY", ), middleware: { gasPrice:async()=>{ return(awaitbundlerClient.getUserOperationGasPrice()).fast// if using pimlico bundlers }, sponsorUserOperation: paymasterClient.sponsorUserOperation,// optional } });
```

Send a transaction

You can now send transactions as normal. ThesponsorUserOperation function will be called before each transaction is signed and sent, applying the custom paymaster logic you have set.

```
consttxHash=awaitsmartAccountClient.sendTransaction({ to:"0xd8da6bf26964af9d7eed9e03e53415d37aa96045", value:parseEther("0.1") });
```