How to use a Privy signer with permissionless.js

permissionless.js allows you to plug in custom signers to control the accounts that you create. Privy is an embedded wallet provider that allows you to easily onboard users to your dapp. It is possible to use Privy as a signer with permissionless.js, allowing you to use Privy to create and control smart accounts and sign transactions.

Install the dependencies

```

npmi @privy-io/react-auth @privy-io/wagmi-connector permissionless viem wagmi

```

Create the Privy provider

Following Privy'squickstart guide , set up the Privy provider in your app. Also integrate the PrivyWagmiConnector, which will allow you to use Privy as a signer with permissionless.js.

```

import{ PrivyProvider }from'@privy-io/react-auth'; import{ PrivyWagmiConnector }from'@privy-io/wagmi-connector';

<PrivyProvider appId={""} config={{ embeddedWallets: { createOnLogin:"all-users", }, }}

    {children} ;

```

Set Privy as the active wallet

In your app, set Privy's embedded wallet as the active wallet for wagmi

```

import{ useWallets }from"@privy-io/react-auth"; import{ usePrivyWagmi }from"@privy-io/wagmi-connector"; import{ useEffect }from"react";

const{wallets}=useWallets(); constembeddedWallet=wallets.find( (wallet)=>wallet.walletClientType==="privy" ); const{setActiveWallet}=usePrivyWagmi()

useEffect(()=>setActiveWallet(embeddedWallet), [embeddedWallet])

```

Create the SmartAccountClient

Create the smart account client using the Privy signer. This is where you would configure what smart account implementation (e.g.Safe ,Kernel , Biconomy,SimpleAccount) and what paymaster logic you want to use.

```

import{ createSmartAccountClient, walletClientToSmartAccountSigner }from"permissionless"; import{ signerToSimpleSmartAccount }from"permissionless/accounts"; import{ useWalletClient }from"wagmi";

const{ data:walletClient }=useWalletClient()

constsigner=walletClientToSmartAccountSigner(walletClient)

constsimpleSmartAccountClient=awaitsignerToSimpleSmartAccount(publicClient, { entryPoint:"0x5FF137D4b0FDCD49DcA30c7CF57E578a026d2789", signer: signer, factoryAddress:"0x9406Cc6185a346906296840746125a0E44976454" })

constsmartAccountClient=createSmartAccountClient({ account: safeAccount, entryPoint:ENTRYPOINT_ADDRESS_V06, chain: sepolia, bundlerTransport:http("https://api.pimlico.io/v1/sepolia/rpc?apikey=API_KEY"), middleware: { gasPrice:async()=>(awaitpimlicoBundlerClient.getUserOperationGasPrice()).fast,// use pimlico bundler to get gas prices, if using pimlico 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: zeroAddress, data:"0x", value:BigInt(0) })

```

For a full example, see theexample permissionless + Privy app .