How to use a Magic signer with permissionless.js

Magic is a popular embedded wallet provider that supports social logins. While social logins are great, your users still need to onramp in order to pay for gas, which introduces significant friction.

By combining permissionless.js with Magic, you can use Magic to enable a smooth social login experience, while using permissionless.js accounts as the smart wallets to sponsor gas for users, batch transactions, and more.

Setup

To use Magic with permissionless.js, first create an application that integrates with Magic.

- Refer to theMagic documentation site
- for instructions on setting up an application with the Magic SDK.
- For a quick start, Magic provides a CLI to create a starter project, available here
- .

Integration

Integrating permissionless.js with Magic is straightforward after setting up the Magic SDK. Magic provides an Externally Owned Account (EOA) wallet to use as a signer with permissionless.js accounts.

Create the Magic object

After following the Magic documentation, you will have access to aMagicBase object as shown below that you can use to create theSmartAccountSigner object:

```

import{ OAuthExtension }from"@magic-ext/oauth" import{ MagicasMagicBase }from"magic-sdk" import{ providerToSmartAccountSigner }from"permissionless"

constmagic=newMagicBase(process.env.MAGIC_API_KEYasstring, { network: { rpcUrl:"https://rpc.ankr.com/eth_sepolia", chainId:11155111, }, extensions: [newOAuthExtension()], })

// Get the Provider from Magic and convert it to a SmartAccountSigner constmagicProvider=awaitmagic.wallet.getProvider() constsmartAccountSigner=awaitproviderToSmartAccountSigner(magicProvider)

```

Use with permissionless.js

SimpleAccount Safe Account Kernel Account Biconomy Account ```

SimpleAccount import{ signerToSimpleSmartAccount }from"permissionless/accounts" import{ createPublicClient, http }from"viem" import{ generatePrivateKey, privateKeyToAccount }from"viem/accounts" import{ sepolia }from"viem/chains"

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

constsmartAccount=awaitsignerToSimpleSmartAccount(publicClient, { signer: smartAccountSigner, factoryAddress:"0x9406Cc6185a346906296840746125a0E44976454", entryPoint:ENTRYPOINT_ADDRESS_V06, })

```
```