How to use an Arcana Auth signer with permissionless.js

Arcana Auth offers a self-custodial Web3 wallet embedded within applications, utilizing asynchronous distributed key generation algorithms for enhanced security and privacy. This wallet, accessible without the need for a browser extension, allows authenticated users to instantly access and sign blockchain transactions within the app environment.

Setup

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

- Refer to theArcana Auth documentation site
- for instructions on setting up an application with the Arcana Auth.
- For a quick start, Arcana Auth provides a web app guide, availablehere
- .

Integration

Integrating permissionless.js with Arcana Auth is straightforward after setting up the project. Arcana Auth provides an Externally Owned Account (EOA) wallet to use as a signer with accounts created using permissionless.js.

Create the authProvider object

After following the Arcana Auth documentation, you will have access to aauthProvider object as shown below which you can use to create aSmartAccountSigner object:

```

import{ AuthProvider,typeConstructorParams }from"@arcana/auth" import{ providerToSmartAccountSigner }from"permissionless"

// Param options here will be specific to your project. See the Arcana Auth docs for more info. constparams:ConstructorParams={} constauthProvider=newAuthProvider(clientId, params)

// Convert the authProvider to a SmartAccountSigner constsmartAccountSigner=awaitproviderToSmartAccountSigner(authProvider.provider)

```

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, })

```
```