How to use a Passport Netwrok signer with permissionless.js

Passport is an MPC-based programmable, distributed, and non-custodial key management system, that allows users to generate wallets, scoped to their application, either via user Passkeys or any developer defined authentication method. Our signer allows you to sign messages and transactions with a Passport Network account.

Setup

To use Passport with permissionless.js, you'll first need to make sure you have configured a scope for your application. For this you can follow the guides below:

- Refer to thePassport documentation
- for instructions on setting up your application with Passport.
- For a primer on setting up your scope you can checkhere
- .

Integration

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

Create the Passport signer

After following the Passport documentation, you will have access to a PassportWalletClient object that you can use to create aSmartAccountSigner object:

```

import{ Passport, TESTNET_RSA_PUBLIC_KEY }from"@0xpass/passport"; import{ WebauthnSigner }from"@0xpass/webauthn-signer"; import{ createPassportClient }from"@0xpass/passport-viem";= import{ walletClientToSmartAccountSigner }from"permissionless"; import{ http }from"viem"; import{ sepolia }from"viem/chains";

constpassport=newPassport({ scope_id:"scope_id", signer:newWebauthnSigner({ rpId:"rpId", rpName:"rpName", }), enclave_public_key:TESTNET_RSA_PUBLIC_KEY, });

constfallbackProvider=http("https://rpc.ankr.com/eth_sepolia");

awaitpassport.setupEncryption(); awaitpassport.register({ username:"test", userDisplayName:"test", });

const[authenticatedHeader]=awaitpassport.authenticate({ username:"test", userDisplayName:"test", });

constclient=awaitcreatePassportClient( authenticatedHeader, fallbackProvider, sepolia, "https://tiramisu.0xpass.io" );

constsigner=walletClientToSmartAccountSigner(client);

```

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, })

```
```