How to use a DFNS signer with permissionless.js

Dfns is an MPC/TSS Wallet-as-a-Service API/SDK provider. Dfns aims to optimize the balance of security and UX by deploying key shares into a decentralized network on the backend while enabling wallet access via biometric open standards on the frontend like Webauthn. Reach outhere to set up a sandbox environment to get started.

Setup

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

- Refer to theDfns documentation site
- for instructions on setting up an application with the Dfns.

Integration

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

Set up Dfns

After following the Dfns documentation, you will have access to adfnsWallet object as shown below:

```

import{ DfnsWallet }from"@dfns/lib-viem" import{ DfnsApiClient }from"@dfns/sdk" import{ AsymmetricKeySigner }from"@dfns/sdk-keysigner" import{ walletClientToSmartAccountSigner }from"permissionless" import{ AccountSource, createWalletClient, http }from"viem" import{ toAccount }from"viem/accounts"

constinitDfnsWallet=(walletId:string)=>{ constsigner=newAsymmetricKeySigner({ privateKey:DFNS_PRIVATE_KEY, credId:DFNS_CRED_ID, appOrigin:DFNS_APP_ORIGIN, })

constdfnsClient=newDfnsApiClient({ appId:DFNS_APP_ID, authToken:DFNS_AUTH_TOKEN, baseUrl:DFNS_API_URL, signer, })

returnDfnsWallet.init({ walletId, dfnsClient, maxRetries:10, }) }

constsepoliaWallet=awaitinitDfnsWallet(SEPOLIA_WALLET_ID) constaccount=toAccount(sepoliaWalletasAccountSource) constwalletClient=createWalletClient({ account, transport:http("https://rpc.ankr.com/eth_sepolia"), })

constsmartAccountSigner=walletClientToSmartAccountSigner(walletClient)

```

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, })

```
```