How to create and use a Thirdweb account with permissionless.js

Picking an EntryPoint

Thirdweb is compatible with EntryPoint versions v0.6 and v0.7. In this guide, we will use EntryPoint v0.7.

Steps

Import the required packages

```
import{ createSmartAccountClient }from"permissionless" import{ createPublicClient, getContract, http, parseEther }from"viem" import{ sepolia }from"viem/chains"
```

Create the clients

First we must create the public, (optionally) pimlico paymaster clients that will be used to interact with the account.

Get your client ID from theThirdweb dashboard for free RPC access.

```
exportconstpublicClient=createPublicClient({ transport:http("https://11155111.rpc.thirdweb.com"), })

exportconstpaymasterClient=createPimlicoClient({ transport:http("https://api.pimlico.io/v2/sepolia/rpc?apikey=API_KEY"), entryPoint: { address: entryPoint07Address, version:"0.7", }, })
```

Create the signer

Thirdweb's accounts can work with any Viem signing accountavailable in permissionless.js .

In this guide, we'll use a private key to create the signer:

```
import{ privateKeyToAccount }from"viem/accounts" import{ createPimlicoClient }from"permissionless/clients/pimlico" import{ entryPoint07Address }from"viem/account-abstraction" import{ toThirdwebSmartAccount }from"permissionless/accounts"

constowner=privateKeyToAccount("0xPRIVATE_KEY")
```

Create the thirdweb account

With your new signer, you can create a thirdweb account.

```
constthirdwebAccount=awaittoThirdwebSmartAccount({ client: publicClient, entryPoint: { address: entryPoint07Address, version:"0.7", }, owner, salt:"0x",// optional address:"0x...",// optional, only if you are using an already created account })
```

The account's address is computed deterministically from the signer, but you can optionally pass ansalt to create any number of different accounts using the same signer. You can also pass anaddress to use an already created thirdweb account.

Create the smart account client

The smart account client is a permissionless.js client that is meant to serve as an almost drop-in replacement for viem'swalletClient .

```
```

```
constsmartAccountClient=createSmartAccountClient({ account: thirdwebAccount, chain: sepolia, paymaster: paymasterClient, bundlerTransport:http("https://api.pimlico.io/v2/sepolia/rpc?apikey=API_KEY"), userOperation: { estimateFeesPerGas:async()=>(awaitpaymasterClient.getUserOperationGasPrice()).fast, }, })
```

Send a transaction

Transactions using permissionless.js simply wrap around user operations. This means you can switch to permissionless.js from your existing viem EOA codebase with minimal-to-no changes.

```
consttxHash=awaitsmartAccountClient.sendTransaction({ to:"0xd8da6bf26964af9d7eed9e03e53415d37aa96045", value:parseEther("0.1"), })
```

This also means you can also use viem Contract instances to transact without any modifications.

```
constnftContract=getContract({ address:"0xFBA3912Ca04dd458c843e2EE08967fC04f3579c2", abi: nftAbi, client: { public: publicClient, wallet: smartAccountClient, }, })

consttxHash=awaitnftContract.write.mint()
```

You can also send an array of transactions in a single batch.

```
constuserOpHash=awaitsmartAccountClient.sendUserOperation({ calls: [ { to:"0xd8da6bf26964af9d7eed9e03e53415d37aa96045", value:parseEther("0.1"), data:"0x", }, { to:"0x1440ec793aE50fA046B95bFeCa5aF475b6003f9e", value:parseEther("0.1"), data:"0x1234", }, ], })
```