How to create and use a Trust smart account with permissionless.js

Trust Wallet , the author ofBarz , provides a smart contract security monitoring service for each and every Barz deployed on-chain including Barz created by SDK. This is a service that TrustWallet provides to builders to build innovative products on top of a secure foundation. Monitoring will automatically start as soon as the Barz account is deployed on-chain; for projects wanting to get security monitoring information, reach out to the smart wallet channel inTrustWallet Discord .

Trust Wallet is one of the most trusted wallet provider empowering more than 122 million users and is the first Web3 wallet to becertified by ISO . After thorough development and extensive security audits, Trust Wallet launched theSmart Wallet powered by account abstraction.

With the recent opensource of their Smart Wallet system Barz, their smart account solution, together with their security infrastructure is disclosed for public to support builders build products on a secure foundation.

This guide will walk you through how to create and use a Barz account with permissionless.js

Steps

Import the required packages

```

import{ ENTRYPOINT_ADDRESS_V06, createSmartAccountClient }from"permissionless" import{ signerToTrustSmartAccount }from"permissionless/accounts" import{ createPimlicoBundlerClient, createPimlicoPaymasterClient, }from"permissionless/clients/pimlico" import{ createPublicClient, getContract, http, parseEther }from"viem" import{ sepolia }from"viem/chains"

```

Create the clients

First we must create the public, bundler, and (optionally) paymaster clients that will be used to interact with the Trust Smart Account.

```

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

exportconstpaymasterClient=createPimlicoPaymasterClient({ transport:http("https://api.pimlico.io/v2/sepolia/rpc?apikey=API_KEY"), entryPoint:ENTRYPOINT_ADDRESS_V06, })

exportconstpimlicoBundlerClient=createPimlicoBundlerClient({ transport:http("https://api.pimlico.io/v2/sepolia/rpc?apikey=API_KEY"), entryPoint:ENTRYPOINT_ADDRESS_V06, })

```

Create the TrustAccount

For a full list of options for creating a TrustAccount, take a look at the reference documentation page forsignerToTrustSmartAccount . You can also pass anaddress to use an already created TrustAccount.

```

consttrustAccount=awaitsignerToTrustSmartAccount(publicClient, { entryPoint:ENTRYPOINT_ADDRESS_V06, signer: signer, index:0n,// optional address:"0x...",// optional, only if you are using an already created 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: trustAccount, entryPoint:ENTRYPOINT_ADDRESS_V06, chain: sepolia, bundlerTransport:http("https://api.pimlico.io/v2/sepolia/rpc?apikey=API_KEY"), middleware: {
```

sponsorUserOperation: paymasterClient.sponsorUserOperation,// optional gasPrice:async()=> (awaitpimlicoBundlerClient.getUserOperationGasPrice()).fast,// if using pimlico bundler }, })

```

## 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.

```

constnftAbi=[{ inputs: [], name:"mint", outputs: [], stateMutability:"nonpayable", type:"function", },]asconst

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.

```

consttxHash=awaitsmartAccountClient.sendTransactions({ transactions: [{ to:"0xd8da6bf26964af9d7eed9e03e53415d37aa96045", value:parseEther("0.1"), data:"0x", }, { to:"0x1440ec793aE50fA046B95bFeCa5aF475b6003f9e", value:parseEther("0.1"), data:"0x1234", },], })

```
```