Tutorial 1 — Send your first gasless transaction

In this tutorial, you will submit your first fully-gasless transaction from an external smart account.

You will set up the necessary @permissionless/wagmi' PermissionlessProvider, paymasterService, ask Pimlico's verifying paymaster to sponsor it, and then ask external smartAccount to submit it on-chain.

Steps

Get a Pimlico API key

To get started, please go to ourdashboard and generate a Pimlico API key.

Create a wagmi app

For new projects, it is recommended to set up your Wagmi app using the create-wagmi command line interface (CLI). This will create a new Wagmi project using TypeScript and install the required dependencies.

pnpm npm yarn bun ```

Terminal pnpm pnpmcreatewagmi

``` Once the command runs, you'll see some prompts to complete. Make sure you selectReact as@permissionless/wagmi only supports React as of now.

After the prompts, create-wagmi will create a directory with your project name and install the required dependencies. Check out the README.md for further instructions (if required).

### Install @permissionless/wagmi

pnpm npm yarn bun ```

Terminal pnpm pnpminstall@permissionless/wagmi

```

Create capabilities & config

Create capabilities for the chains that you would like to sponsor the transactions for the users.

```

exportconstconfig=createConfig({ chains: [baseSepolia], connectors: [ // coinbase wallet is one of the smart accounts that supports ERC-7677 coinbaseWallet({ appName:"Pimlico", preference:"smartWalletOnly"}) ], transports: {

} })

constcapabilities={ paymasterService: {

url:https://api.pimlico.io/v2{baseSepolia.id}/rpc?apikey={pimlicoApiKey} } } }

```

In this example, you can integrate any service provider that complies with the ERC-7677 standard. For demonstration purposes, we are using Pimlico as the paymaster service provider.

Ensure you configure the service for all the blockchain networks where you plan to sponsor transactions. We are showcasing sponsorship for baseSepolia.

Wrap App in Context Provider

Wrap your app in thePermissionlessProvider React Context Provider and pass thecapabilities you created earlier.

App.tsx wagmi.ts ```

File App.tsx import{ WagmiProvider }from"wagmi" import{ baseSepolia }from"wagmi/chains" import{ PermissionlessProvider }from"@permissionless/wagmi" import{ config, capabilities }from"./wagmi.ts"

ReactDOM.createRoot(root).render(<PermissionlessProvider capabilities={capabilities}

{/* ... /})

``` Check out thePermissionlessProvider docs to learn more about React Context in @permissionless/wagmi.

## Use @permissionless/wagmi

Now that everything is set up, every component inside the Permissionless Provider can use Permissionless React Hooks.

```

File App.tsx import{ useSendTransaction, useWaitForTransactionReceipt }from"wagmi" import{ useSendTransaction, useWaitForTransactionReceipt }from"@permissionless/wagmi"

functionApp() { const{ sendTransaction, data:transactionReference, isPending }=useSendTransaction()

const{data:receipt,isPending:isReceiptPending}= useWaitForTransactionReceipt({ id: transactionReference })

constsendTransactionCallback=useCallback(async()=>{ console.log("Sending transaction...") sendTransaction({ to:"0xd8da6bf26964af9d7eed9e03e53415d37aa96045", data:"0x1234" }) }, [sendTransaction])

return(

Send test transaction

{isPending&&

Sending transaction...
}

{transactionReference&&(

Awaiting confirmation:{transactionReference}

)}

{receipt&&

{receipt.status}
}

Send Transaction) }

```

As you notice,@permissionless/wagmi exposes the same api aswagmi .@permissionless/wagmi is a drop-in replacement forwagmi that provides the same functionality, but with the added benefit of being able to use sponsor transactions.

The above code will sponsor transactions only when the external account is a smart account and supports ERC-7677. If the external account doesn't have the capability, the transaction will be sent as a normal transaction.

To test the above code connect using Coinbase SmartAccount and try sending your first transaction. By sending this transaction, you will have:

- Made Pimlico's verifying paymaster sponsor the user operation's gas fees
- Executed a simple sponsored transaction tovitalik.eth
- 's address from the Coinbase SmartAccount

All in a couple lines of code.

Congratulations, you are now a pioneer of Account Abstraction!

Pleaseget in touch if you have any questions or if you'd like to share what you're building!
```