Using the Solana Wallet Adapter - Recommended

Learn how to add Magic Eden Wallet to your Solana App via the widely used Solana Wallet AdapterSuggest Edits

This guide follows the Solana Wallet adapter guide for React and this particular example makes use of NextJs and Typescript. Please refer to their documentation for other frameworks.

Note: If you already have Solana Wallet Adapter, Magic Eden Wallet should work out of the box. If you are creating a new Solana project and are using the popularcreate-solana-dapp library, this will also use Solana Wallet adapter by default and support the ME Wallet.

This is the recommended way of handling wallet connection because they abstract away a lot of the complexities. when used in tandem with libraries likeSolana Web3Js and theAnchor Framework , writing frontend code for Solana is easy and fun!

Install

Install the necessary dependencies:

npm install --save \ @solana/wallet-adapter-base \ @solana/wallet-adapter-react \ @solana/wallet-adapter-react-ui \ @solana/wallet-adapter-wallets \ @solana/web3.js \ react

Setup

Guide

This guide will help you setup your Wallet Adapter in 3 steps:

1. Configure your imports
2. Create a
3. Wallet
4. provider to wrap your app in.
5. Display buttons to connect and disconnect from a wallet.

1. Import Solana Wallet Adapter

JSX "use client"; import { ConnectionProvider, WalletProvider, } from "@solana/wallet-adapter-react"; import { WalletAdapterNetwork } from "@solana/wallet-adapter-base"; import { UnsafeBurnerWalletAdapter } from "@solana/wallet-adapter-wallets"; import { WalletModalProvider, WalletDisconnectButton, WalletMultiButton, } from "@solana/wallet-adapter-react-ui"; import { clusterApiUrl } from "@solana/web3.js"; import { useMemo } from "react";

// Default styles that can be overridden by your app require("@solana/wallet-adapter-react-ui/styles.css");

2. Create a Wallet Component:

Create a wallet component that will wrap your page or app.

This TypeScript code defines a React functional component named Wallet that takes React nodes as children. It sets up a Solana blockchain connection and provides wallet functionality for a React application.

It uses the Solana Wallet Adapter libraries to connect to the Devnet network, sets up an RPC endpoint, and configures a list of wallet adapters, in this case, an UnsafeBurnerWalletAdapter . The component wraps its children with providers that pass down the blockchain connection and wallet context, allowing child components to interact with the Solana blockchain.

JSX function Wallet({ children }: { children: React.ReactNode }) { // Can be set to 'devnet', 'testnet', or 'mainnet-beta' const network = WalletAdapterNetwork.Devnet;

// You can also provide a custom RPC endpoint const endpoint = useMemo(() => clusterApiUrl(network), []);

// @solana/wallet-adapter-wallets includes all the adapters but supports tree shaking -- // Only the wallets you configure here will be compiled into your application const wallets = useMemo(() => [new UnsafeBurnerWalletAdapter()], []);

return ({children}); }

3. Add Wallet Buttons to your Page or Component

Now wrap your page or app with your wallet component, and add the wallet buttons to be able to select and disconnect a wallet.

If using NextJS 14 with App Router, make sure you add use client; at the top of the file. JSX export default function SolanaDemo() { return (

); }

Full Component

For reference here is a full page from our demo app found here

JSX "use client"; import { ConnectionProvider, WalletProvider, } from "@solana/wallet-adapter-react"; import { WalletAdapterNetwork } from "@solana/wallet-adapter-base"; import { UnsafeBurnerWalletAdapter } from "@solana/wallet-adapter-wallets"; import { WalletModalProvider, WalletDisconnectButton, WalletMultiButton, } from "@solana/wallet-adapter-react-ui"; import { clusterApiUrl } from "@solana/web3.js"; import { useMemo } from "react";

// Default styles that can be overridden by your app require("@solana/wallet-adapter-react-ui/styles.css");

function Wallet({ children }: { children: React.ReactNode }) { // Can be set to 'devnet', 'testnet', or 'mainnet-beta' const network = WalletAdapterNetwork.Devnet;

// You can also provide a custom RPC endpoint const endpoint = useMemo(() => clusterApiUrl(network), []);

// @solana/wallet-adapter-wallets includes all the adapters but supports tree shaking -- // Only the wallets you configure here will be compiled into your application const wallets = useMemo(() => [new UnsafeBurnerWalletAdapter()], []);

return ({children}); }

export default function SolanaDemo() { return (

); }

Additional Resources

Please refer to theSolana Wallet Adapter Github for more information and detailed implementation guides. Updatedabout 1 month ago