Magic Eden Wallet on EVM with WAGMI

Learn how to Add Magic Eden Wallet to any EVM app using WAGMISuggest Edits

Getting Started

If you are interested in using WAGMI in your brand new react application, follow the instructionshere to initialize your app with the CLI. Otherwise, follow the instructions below.

First you will need to installWAGMI

Shell pnpm add wagmi @tanstack/react-query

WAGMI Config Setup

Next you will need to create a wagmi config file: In this example you can find it under wagmi-config.ts

Here you need to create the default WAGMI config, and under connector, add a Magic Eden Wallet Injector.

This is necessary to have WAGMI pick up the Magic Eden Wallet option, and add it to the Wallet Options a user can choose from.

JSX import { http, createConfig } from "wagmi"; import { base, mainnet } from "wagmi/chains"; import { injected } from "wagmi/connectors";

const ME_CONNECT = injected({ target() { return { id: "Magic Eden", name: "Magic Eden", provider: typeof window !== "undefined" ? window?.magicEden?.ethereum : undefined, }; }, });

export const config = createConfig({ chains: [mainnet, base], connectors: [injected(), ME_CONNECT], transports: { [mainnet.id]: http(), [base.id]: http(), }, });

Setup WAGMI Provider and QueryClient

Once you have the config setup, you need to wrap your page or app with

WagmiProvider using the config object we previously created, and QueryClient .

If using NextJS 14 with App Router, make sure you add use client; at the top of the file. JSX "use client"; import { QueryClient, QueryClientProvider } from "@tanstack/react-query";

export default function EvmDemoPage() { return (

{wallet component will go here}
); }

Display Wallet Options Component

Once we have the page setup, we need to create components that handle:

1. Detects Wallet Options from WAGMI.
2. Shows Wallet Options.
3. Handles selecting and authenticating a wallet.
4. Handles disconnecting a wallet.

Update Imports

Start by updating your imports:

JSX "use client"; import { Connector, WagmiProvider, useAccount, useConnect, useDisconnect, useEnsAvatar, useEnsName, } from "wagmi"; import { config } from "../../../wagmi-config"; import { QueryClient, QueryClientProvider } from "@tanstack/react-query"; import { useState, useEffect } from "react";

Display Wallet Options

Next create a component that detects and displays all wallets available.

This code creates a WalletOptions component that uses the useConnect hook from the wagmi library to get a list of

available wallet connectors. It then maps over these connectors to create a WalletOption component for each one.

The WalletOption component takes a connector and an onClick function as props. It uses the useState and useEffect hooks from React to check if the provider for the given connector is ready. If the provider is ready, it enables a button that displays the name of the connector and calls the onClick function when clicked.

This allows users to see all available wallet options and connect to their chosen wallet.

JSX export function WalletOptions() { const { connectors, connect } = useConnect();

return connectors.map((connector) => (connect({ connector })} />)); }

function WalletOption({ connector, onClick, }: { connector: Connector; onClick: () => void; }) { const [ready, setReady] = useState(false);

useEffect(() => { (async () => { const provider = await connector.getProvider(); setReady(!!provider); })(); }, [connector]);

return ({connector.name}); }

Connect an active wallet

Now create a component that handles a wallet being connected and displays the account

This Account component handles the display of the connected wallet's account information. It uses the useAccount hook from the wagmi library to get the address of the connected wallet. It also uses the useDisconnect hook to provide a function that can be called to disconnect the wallet.

The useEnsName and useEnsAvatar hooks are used to get the ENS name and avatar for the connected wallet, if they exist. The ENS name and avatar are then displayed along with the wallet's address.

A button is then provided to allow the user to disconnect their wallet. When this button is clicked, the disconnect function provided by the useDisconnect hook is called.

This component is designed to be used once a wallet has been connected. If no wallet is connected, it will not display anything.

JSX const queryClient = new QueryClient();

export function Account() { const { address } = useAccount(); const { disconnect } = useDisconnect(); const { data: ensName } = useEnsName({ address }); const { data: ensAvatar } = useEnsAvatar({ name: ensName! });

return (

{ensAvatar && } {address && (
{ensName ? {ensName} ({address}) : address}
disconnect()} className="px-4 py-2 rounded border">
)} Disconnect
); }

Create a Connect Wallet Component

Now, create a component that detects when a wallet is present, and if none are connected displays all available wallet options.

This ConnectWallet component checks if a wallet is already connected using the isConnected property from the useAccount hook. If a wallet is connected, it returns the Account component which displays the connected wallet's account information and provides a disconnect button. If no wallet is connected, it returns the WalletOptions component which displays all available wallet options for the user to connect to.

JSX function ConnectWallet() { const { isConnected } = useAccount(); if (isConnected) return ; return (

); }

Add Wallet Button to Page or Component

Finally, add your ConnectWallet component to your page, and you should now be able to see all wallet options, select Magic Eden Wallet and use it to transact.

JSX export default function EvmDemoPage() { return (

); }

Full Page

JSX "use client"; import { Connector, WagmiProvider, useAccount, useConnect, useDisconnect, useEnsAvatar, useEnsName, } from "wagmi"; import { config } from "../../../wagmi-config"; import { QueryClient, QueryClientProvider } from "@tanstack/react-query"; import { useState, useEffect } from "react";

const queryClient = new QueryClient();

export function WalletOptions() { const { connectors, connect } = useConnect();

return connectors.map((connector) => (connect({ connector })} />)); }

function WalletOption({ connector, onClick, }: { connector: Connector; onClick: () => void; }) { const [ready, setReady] = useState(false);

useEffect(() => { (async () => { const provider = await connector.getProvider(); setReady(!!provider); })(); }, [connector]);

return ({connector.name}); }

export function Account() { const { address } = useAccount(); const { disconnect } = useDisconnect(); const { data: ensName } = useEnsName({ address }); const { data: ensAvatar } = useEnsAvatar({ name: ensName! });

return (

{ensAvatar && } {address && (
{ensName ? {ensName} ({address}) : address}
disconnect()} className="px-4 py-2 rounded border">
Disconnect
)}
); }

function ConnectWallet() { const { isConnected } = useAccount(); if (isConnected) return ; return (

); }

export default function EvmDemoPage() { return (

); } Congratulations! You should now be able to select and use Magic Eden Wallet in your React app! Updated23 days ago * Table of Contents * * Getting Started * * * WAGMI Config Setup * * * Setup WAGMI Provider and QueryClient * * * Display Wallet Options Component * * * Add Wallet Button to Page or Component* * * Full Page