

Wallet Services

[@web3auth/wallet-services-plugin](#)

[^](#)

Wallet Services Plugin Enable allows your application to use templated wallet UI screens. Take control of and personalize the wallet interface, and optionally, activate an embedded wallet for your users.

This Documentation is based on the 8.0.0 SDK Version.

Note Access to Wallet Services is gated and available on the Scale plan and above. You can use this feature in the development environment for free.

Wallet Services is currently available for all EVM chains.

Installation[^](#)

- npm
- Yarn
- pnpm

```
npm install --save @web3auth/wallet-services-plugin yarn add @web3auth/wallet-services-plugin pnpm add @web3auth/wallet-services-plugin
```

Initialization[^](#)

Import the `WalletServicesPlugin` class from `@web3auth/wallet-services-plugin`.

```
import
```

```
{
```

```
  WalletServicesPlugin
```

```
}
```

```
from
```

```
"@web3auth/wallet-services-plugin";
```

Assign the `WalletServicesPlugin`

class to a variable [^](#)

After creating your `Web3Auth` instance, you need to initialize the Wallet Services Plugin and add it to a class for further usage.

```
const walletServicesPlugin =
```

```
new
```

```
WalletServicesPlugin ( options );
```

 This constructor takes an object with `wsEmbedOpts` & `walletInitOptions` as input.

Arguments[^](#)

While initializing the Wallet Services plugin, you need to pass on the following parameters to the constructor:

- Table
- Interface

Parameter Description `wsEmbedOpts?` Configuration options for Wallet Services. It accepts `Partial` as a value.

`walletInitOptions?` Parameters to customize your Wallet Services UI within the application. It accepts `Partial` as a value.

```
constructor(options?: { wsEmbedOpts?: Partial; walletInitOptions?: Partial; });
```

1. `wsEmbedOpts`

[^](#)

wsEmbedOpts is an optional parameter that allows you to pass in the configuration options for Wallet Services. It takes theCtorArgs object as input which contains the following parameters:

- Table
- Interface

CtorArgs

Parameter Description modalZIndex? Z-index of the modal and iframe. Default value is 99999 web3AuthClientId Web3Auth Client ID instring . web3AuthNetwork Web3auth network to be used. It acceptsOPENLOGIN_NETWORK_TYPE as a value. export

interface

CtorArgs

```
{ /* * Z-index of the modal and iframe * @defaultValue 99999 modalZIndex ? :
```

```
number ; /* * You can get your clientId/projectId by registering your * dapp on {@link "https://dashboard.web3auth.io"/ developer dashboard} / web3AuthClientId :
```

```
string ; /* * network specifies the web3auth network to be used/ web3AuthNetwork :
```

```
OPENLOGIN_NETWORK_TYPE ; }
```

```
export
```

```
type
```

```
OPENLOGIN_NETWORK_TYPE
```

```
=
```

```
( typeof
```

```
OPENLOGIN_NETWORK ) [ keyof
```

```
typeof
```

```
OPENLOGIN_NETWORK ] ; export
```

```
declare
```

```
const
```

```
OPENLOGIN_NETWORK :
```

```
{ readonly
```

```
MAINNET :
```

```
"mainnet" ; readonly
```

```
TESTNET :
```

```
"testnet" ; readonly
```

```
CYAN :
```

```
"cyan" ; readonly
```

```
AQUA :
```

```
"aqua" ; readonly
```

```
CELESTE :
```

```
"celeste" ; readonly
```

```
SAPPHIRE_DEVNET :
```

```
"sapphire_devnet" ; readonly
```

SAPPHIRE_MAINNET :

```
"sapphire_mainnet" ; } ;
```

2.walletInitOptions

[a](#)

The walletInitOptions from Wallet Services is an optional parameter that allows you to customise your Wallet Services UI within the application. It takes the object WsEmbedParams as input which contains multiple parameters that allow you to customise the UI of the plugin.

- Table
- Interface

WsEmbedParams

Parameter Description chainConfig? Chain config of the Blockchain to connect with. It accepts EthereumProviderConfig . whiteLabel? White Label parameters to whitelisting the Wallet Services UI. It accepts WhiteLabelParams . export

interface

WsEmbedParams

```
{ /* * Chain to connect with/ chainConfig ? :
```

```
EthereumProviderConfig ; /* * Build Environment of WsEmbed. * * production uses https://wallet.web3auth.io, * * staging uses https://staging-wallet.web3auth.io, * * testing uses https://develop-wallet.web3auth.io (latest internal build) * * development uses http://localhost:4050 (expects wallet-services-frontend to be run locally), * * @defaultValue production / buildEnv ? :
```

```
WS_EMBED_BUILD_ENV_TYPE ; /* * Enables or disables logging. * * Defaults to false in prod and true in other environments / enableLogging ? :
```

```
boolean ; /* * url of widget to load * * Defaults to true * @defaultValue true/ walletUrls ? :
```

```
Partial < Record < WS_EMBED_BUILD_ENV_TYPE ,
```

WalletUrlConfig

```
; /* Determines how to show confirmation screens * * @defaultValue none for wallet services * @defaultValue popup for safe-auth, mocaverse */ /* Allows you to customize the look & feel of the widget */ whiteLabel ? :
```

```
{ /* * whether to show/hide ws-embed widget. * * Defaults to true * @defaultValue true/ showWidgetButton ? :
```

```
boolean ; /* * Determines where the wsEmbed widget is visible on the page. * @defaultValue bottom-left buttonPosition ? :
```

```
BUTTON_POSITION_TYPE ; hideNftDisplay ? :
```

```
boolean ; hideTokenDisplay ? :
```

```
boolean ; hideTransfers ? :
```

```
boolean ; hideTopup ? :
```

```
boolean ; hideReceive ? :
```

```
boolean ; defaultPortfolio ? :
```

```
"token"
```

```
|
```

```
"nft" ; }
```

```
&
```

```
WhiteLabelData ; }
```

Add the walletServicesPlugin

class to your Web3Auth instance[^](#)

After initializing the walletServicesPlugin , use the addPlugin() function to your Web3Auth instance, you can use the Wallet Services Plugin for your dApp.

```
web3AuthInstance.addPlugin(walletServicesPlugin);
```

Example[^](#)

In this example, we're adding the walletServicesPlugin with a very basic configuration.

```
import
{
  WalletServicesPlugin
}
from
"@web3auth/wallet-services-plugin" ;

const walletServicesPlugin =
new
WalletServicesPlugin ( ) ; web3auth . addPlugin ( walletServicesPlugin ) ;

// Show the wallet services plugin await walletServicesPlugin . showWalletUi ( ) ;

// Show the wallet connect scanner await walletServicesPlugin . showWalletConnectScanner ( ) ;

// Show the checkout await walletServicesPlugin . showCheckout ( ) ;
```

Using Wallet Services[^](#)

showWalletConnectScanner()

[^](#)

Shows the Wallet Connect Scanner to connect with dApps having Wallet Connect login option. This is useful for interoperability with dApps having Wallet Connect login option.

Example[^](#)

```
import
{
  WalletServicesPlugin
}
from
"@web3auth/wallet-services-plugin" ;

const walletServicesPlugin =
new
WalletServicesPlugin ( ) ; web3auth . addPlugin ( walletServicesPlugin ) ;

// Add the plugin to web3auth

await walletServicesPlugin . showWalletConnectScanner ( ) ;
```

showCheckout()

[^](#)

Shows the TopUp modal to select local currency and amount to top up the wallet.

Example[^](#)

```
import
{
  WalletServicesPlugin
}
from
"@web3auth/wallet-services-plugin" ;

const walletServicesPlugin =
new
WalletServicesPlugin ( ) ; web3auth . addPlugin ( walletServicesPlugin ) ;

// Add the plugin to web3auth

await walletServicesPlugin . showCheckout ( ) ;

// Opens the TopUp modal
```

showWalletUi()

[^](#)

Shows the Wallet Services modal UI to be used as a wallet UI.

Example[^](#)

```
import
{
  WalletServicesPlugin
}
from
"@web3auth/wallet-services-plugin" ;

const walletServicesPlugin =
new
WalletServicesPlugin ( ) ; web3auth . addPlugin ( walletServicesPlugin ) ;

// Add the plugin to web3auth

await walletServicesPlugin . showWalletUi ( ) ;

// Opens the TopUp modal
```

Using with Web3Auth Modal[^](#)

```
import
{
  CHAIN_NAMESPACES ,
  WEB3AUTH_NETWORK ,
  IProvider
```

```

}

from
"@web3auth/base" ; import
{
Web3Auth
}

from
"@web3auth/modal" ; import
{
EthereumPrivateKeyProvider
}

from
"@web3auth/ethereum-provider" ; import
{
WalletServicesPlugin
}

from
"@web3auth/wallet-services-plugin" ;

const clientId =
"BPi5PB_UiIZ-cPz1GtV5i1I2iOSOHuimiXBI0e-Oe_u6X3oVAbCiAZOTEBtTXw4tsluTITPqA8zMsfxIKMjjiqNQ" ;

const chainConfig =
{ chainId :
"0x1" ,

// Please use 0x1 for Mainnet rpcTarget :
"https://rpc.ankr.com/eth" , displayName :
"Ethereum Mainnet" , blockExplorerUrl :
"https://etherscan.io/" , ticker :
"ETH" , tickerName :
"Ethereum" , logo :
"https://images.toruswallet.io/eth.svg" , } ;

const privateKeyProvider =
new
EthereumPrivateKeyProvider ( {
config :
{ chainConfig }
} ) ;

const web3AuthOptions =

```

```
{ clientId , chainConfig :  
{  
... chainConfig ,  
chainNamespace :  
CHAIN_NAMESPACES . EIP155  
} , web3AuthNetwork :  
WEB3AUTH_NETWORK . SAPPHIRE_MAINNET , privateKeyProvider : privateKeyProvider , } ;  
const web3auth =  
new  
Web3Auth ( web3AuthOptions as  
Web3AuthOptions ) ;  
const walletServicesPlugin =  
new  
WalletServicesPlugin ( ) ; web3auth . addPlugin ( walletServicesPlugin ) ;  
// Add the plugin to web3auth  
// Show the wallet services plugin await walletServicesPlugin . showWalletUi ( ) ;  
// Show the wallet connect scanner await walletServicesPlugin . showWalletConnectScanner ( ) ;  
// Show the checkout await walletServicesPlugin . showCheckout ( ) :Edit this page Previous Multi Factor Authentication Next Pregenerated Wallets API
```