

# Common Provider

## [@web3auth/base-provider](#)

â

For connecting to blockchains other than EVM, Solana, and XRPL, you need to use the private key from Web3Auth and manually make RPC calls to the blockchain. This flow is facilitated by @web3auth/base-provider package.

In this section, we'll explore more about how you can use this provider with our SDKs.

### Installationâ

#### [@web3auth/base-provider](#)

â

- npm
- Yarn
- pnpm

```
npm install --save @web3auth/base-provider yarn add @web3auth/base-provider pnpm add @web3auth/base-provider
```

### Initialisationâ

Import the `CommonPrivateKeyProvider` class from `@web3auth/base-provider`.

```
import
{
  CommonPrivateKeyProvider
}
from
"@web3auth/base-provider";
```

#### Assign the `CommonPrivateKeyProvider`

class to a variableâ

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

```
const privateKeyProvider =
```

```
new
```

```
CommonPrivateKeyProvider ( ) ;
```

Note \* The common private key provider only exposes one RPC method (i.e. 'private\_key') to get the private key of the logged-in user.

### Setting up the providerâ

#### For `Web3Auth PnP Web SDKs`â

If you are using `chainNamespace: "other"` while initializing `Web3Auth` or `Web3AuthNoModal` with the `OpenloginAdapter`, you need to add the `privateKeyProvider` to the `OpenLogin` instance.

```
const chainConfig =
{ chainId :
  "0x1" , chainNamespace :
  CHAIN_NAMESPACES . OTHER , rpcTarget :
  "https://any-rpc-endpoint.com" , } ;
const web3auth =
new
Web3AuthNoModal ( { clientId , chainConfig , web3AuthNetwork :
"sapphire_mainnet" , } ) ;
const privateKeyProvider =
new
CommonPrivateKeyProvider ( config : chainConfig ) ;
const openloginAdapter =
new
OpenloginAdapter ( { privateKeyProvider , adapterSettings :
{ ... } , mfaSettings :
{ ... } , loginSettings :
{ ... } , } ) ; web3auth . configureAdapter ( openloginAdapter ) ;
const web3authProvider =
await web3auth . connectTo ( WALLET_ADAPTERS . OPENLOGIN , { loginProvider :
"google" , } ) ;
// Use this provider to export the private key of the user
```

#### For `Single Factor Auth Web SDK`â

While using the SFA Web SDK, you need to pass the provider during the initialization of SDK, while calling the `init()` function.

```
const chainConfig =
{ chainId :
  "0x1" , chainNamespace :
  CHAIN_NAMESPACES . OTHER , rpcTarget :
```

```

"https://any-rpc-endpoint.com" , } ;

const web3authSfa =

new

Web3Auth ( { clientId ,

// Get your Client ID from the Web3Auth Dashboard chainConfig , web3AuthNetwork :

"sapphire_mainnet" , usePnPKey :

false ,

// Setting this to true returns the same key as PnP Web SDK, By default, this SDK returns CoreKitKey. } ) ;

const privateKeyProvider =

new

CommonPrivateKeyProvider ( config : chainConfig ) ;

web3authSfa . init ( privateKeyProvider ) ;

const web3authSfaprovider =

await web3auth . connect ( { verifier :

"web3auth-sfa-verifier" ,

// e.g. web3auth-sfa-verifier replace with your verifier name, and it has to be on the same network passed in init(). verifierId :

"name@email.com" ,

// e.g. Yux1873xnibdui Or name@email.com replace with your verifier id(sub or email)'s value. idToken :

"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6IiRZT2dnXy01RU9FYmxhWS1WVlJZcVZhREFncHRuZktWNU1aUEMwdzAifQ.eyJpYXQiOiJlE2ODY4OTMzMzYsImF1ZCI6Ikl9kS2N4QzBIY0tX-0sNIP4pda7aASKko0dm7EsmQvA-uq7cKFJWgAD8S9jC8ZogiEtJ6MRnjDYY8UdTwBL7mQ" ,

// replace with your newly created unused JWT Token. } ) ;

// use this provider to export the private key of the user

```

## Usage

On connection, you can use this provider as a private key provider to expose the user's private key in the frontend context

// Assuming the user is already logged in. async

```
getPrivateKey ( )
```

```
{ const privateKey =
```

```
await web3authSfaprovider . request ( { method :
```

```
"private_key" } ) ; // Do something with privateKey ) Edit this page Previous XRPL Provider Next SafeAuth
```