Providers Google

Version: v4

Google

Documentation

https://developers.google.com/identity/protocols/oauth2

Configuration

https://console.developers.google.com/apis/credentials

The "Authorized redirect URIs" used when creating the credentials must include your full domain and end in the callback path. For example;

- For production: `https://{YOUR_DOMAIN}/api/auth/callback/google`
- For development: `http://localhost:3000/api/auth/callback/google`

Options

The **Google Provider** comes with a set of default options:

- Google Provider options

You can override any of the options to suit your own use case.

Example

```
import GoogleProvider from "next-auth/providers/google";
...
providers: [
  GoogleProvider({
```

```
    clientId: process.env.GOOGLE_CLIENT_ID,
    clientSecret: process.env.GOOGLE_CLIENT_SECRET
  })
]
...
```

DANGER

Google only provides Refresh Token to an application the first time a user signs in.

To force Google to re-issue a Refresh Token, the user needs to remove the application from their account and sign in again: https://myaccount.google.com/permissions

Alternatively, you can also pass options in the `params` object of `authorization` which will force the Refresh Token to always be provided on sign in, however this will ask all users to confirm if they wish to grant your application access every time they sign in.

If you need access to the RefreshToken or AccessToken for a Google account and you are not using a database to persist user accounts, this may be something you need to do.

```
const options = {
  ...
  providers: [
    GoogleProvider({
      clientId: process.env.GOOGLE_ID,
      clientSecret: process.env.GOOGLE_SECRET,
      authorization: {
        params: {
          prompt: "consent",
          access_type: "offline",
          response_type: "code"
        }
      }
    })
  ],
  ...
}
```

TIP

Google also returns a `email_verified` boolean property in the OAuth profile.

You can use this property to restrict access to people with verified accounts at a particular domain.

```
const options = {
  ...
  callbacks: {
    async signIn({ account, profile }) {
      if (account.provider === "google") {
        return profile.email_verified &&
profile.email.endsWith("@example.com")
      }
      return true // Do different verification for other providers that
don't have `email_verified`
    },
  }
  ...
}
```

Edit this page

*Last updated on **May 16, 2024** by **Codie Newark***