Version: v4

Upgrade Guide (v4)

NextAuth.js version 4 includes a few breaking changes from the last major version (3.x). So we're here to help you upgrade your applications as smoothly as possible. It should be possible to upgrade from any version of 3.x to the latest 4 release by following the next few migration steps.

> **NOTE**
>
> Version 4 has been released to GA 🚨
>
> We encourage users to try it out and report any and all issues they come across.

You can upgrade to the new version by running:

npm yarn pnpm

```
npm install next-auth
```

`next-auth/jwt`

We no longer have a default export in `next-auth/jwt`. To comply with this, change the following:

```
- import jwt from "next-auth/jwt"
+ import { getToken } from "next-auth/jwt"
```

`next-auth/react`

We've renamed the client-side import source to `next-auth/react`. To comply with this change, you will simply have to rename anywhere you were using `next-auth/client`.

For example:

```
- import { useSession } from "next-auth/client"
+ import { useSession } from "next-auth/react"
```

We've also made the following changes to the names of the exports:

- `setOptions`: Not exposed anymore, use `SessionProvider` props
- `options`: Not exposed anymore, use `SessionProvider` props
- `session`: Renamed to `getSession`
- `providers`: Renamed to `getProviders`
- `csrfToken`: Renamed to `getCsrfToken`
- `signin`: Renamed to `signIn`
- `signout`: Renamed to `signOut`
- `Provider`: Renamed to `SessionProvider`

Introduced in https://github.com/nextauthjs/next-auth/releases/tag/v4.0.0-next.12

`SessionProvider`

Version 4 makes using the `SessionProvider` mandatory. This means that you will have to wrap any part of your application using `useSession` in this provider, if you were not doing so already. The `SessionProvider` has also undergone a few further changes:

- `Provider` is renamed to `SessionProvider`
- The options prop is now flattened as the props of SessionProvider.
- `keepAlive` has been renamed to `refetchInterval`.
- `clientMaxAge` has been removed in favor of `refetchInterval`, as they overlap in functionality, with the difference that `refetchInterval` will keep re-fetching the session periodically in the background.

The best practice for wrapping your app in Providers is to do so in your `pages/_app.jsx` file.

An example use-case with these new changes:

```
import { SessionProvider } from "next-auth/react"

export default function App({
  Component,
  pageProps: { session, ...pageProps },
}) {
  return (
    // `session` comes from `getServerSideProps` or `getInitialProps`.
    // Avoids flickering/session loading on first load.
    <SessionProvider session={session} refetchInterval={5 * 60}>
      <Component {...pageProps} />
    </SessionProvider>
  )
}
```

Introduced in https://github.com/nextauthjs/next-auth/releases/tag/v4.0.0-next.12

Providers

Providers now need to be imported individually.

```
- import Provider from "next-auth/providers"
- Providers.Auth0({...})
- Providers.Google({...})
+ import Auth0Provider from "next-auth/providers/auth0"
+ import GoogleProvider from "next-auth/providers/google"
+ Auth0Provider({...})
+ GoogleProvider({...})
```

1. The `AzureADB2C` provider has been renamed `AzureAD`.
2. The `Basecamp` provider has been removed, see explanation here.
3. The GitHub provider by default now will not request full write access to user profiles. If you need this scope, please add `user` to the scope option manually.

The following new options are available when defining your Providers in the configuration:

1. `authorization` (replaces `authorizationUrl`, `authorizationParams`, `scope`)
2. `token` replaces (`accessTokenUrl`, `headers`, `params`)
3. `userinfo` (replaces `profileUrl`)
4. `issuer`(replaces `domain`)

For more details on their usage, please see **options** section of the OAuth Provider documentation.

When submitting a new OAuth provider to the repository, the `profile` callback is expected to only return these fields from now on: `id`, `name`, `email`, and `image`. If any of these are missing values, they should be set to `null`.

Also worth noting is that `id` is expected to be returned as a `string` type (For example if your provider returns it as a number, you can cast it by using the `.toString()` method). This makes the returned profile object comply across all providers/accounts/adapters, and hopefully cause less confusion in the future.

Implemented in: **#2411** Introduced in **https://github.com/nextauthjs/next-auth/releases/tag/v4.0.0-next.20**

`useSession` Hook

The `useSession` hook has been updated to return an object. This allows you to test states much more cleanly with the new `status` option.

```
- const [ session, loading ] = useSession()
+ const { data: session, status } = useSession()
+ const loading = status === "loading"
```

Check the docs for the possible values of both `session.status` and `session.data`.

Introduced in **https://github.com/nextauthjs/next-auth/releases/tag/v4.0.0-next.18**

Named Parameters

We have changed the arguments to our callbacks to the named parameters pattern. This way you don't have to use dummy `_` placeholders or other tricks.

Callbacks

The signatures for the callback methods now look like this:

```
- signIn(user, account, profileOrEmailOrCredentials)
+ signIn({ user, account, profile, email, credentials })
```

```
- redirect(url, baseUrl)
+ redirect({ url, baseUrl })
```

```
- session(session, tokenOrUser)
+ session({ session, token, user })
```

```
- jwt(token, user, account, OAuthProfile, isNewUser)
+ jwt({ token, user, account, profile, isNewUser })
```

Introduced in https://github.com/nextauthjs/next-auth/releases/tag/v4.0.0-next.17

Events

Two event signatures have changed to also use the named parameters pattern, `signOut` and `updateUser`.

```
// [...nextauth].js
...
events: {
- signOut(tokenOrSession),
+ signOut({ token, session }), // token if using JWT, session if DB persisted
sessions.
- updateUser(user)
+ updateUser({ user })
}
```

Introduced in https://github.com/nextauthjs/next-auth/releases/tag/v4.0.0-next.20

JWT configuration

We have removed some of the configuration options when using JSON Web Tokens, here's the PR for more context.

```
export default NextAuth({
  // ...
  jwt: {
    secret,
    maxAge,
-   encryptionKey
-   signingKey
-   encryptionKey
-   verificationOptions
    encode({
        token
        secret
        maxAge
-       signingKey
-       signingOptions
-       encryptionKey
-       encryptionOptions
-       encryption
    }) {},
    decode({
        token
        secret
-       maxAge
-       signingKey
-       verificationKey
-       verificationOptions
-       encryptionKey
-       decryptionKey
-       decryptionOptions
-       encryption
    }) {}
  }
})
```

Logger API

The logger API has been simplified to use at most two parameters, where the second is usually an object (`metadata`) containing an `error` object. If you are not using the logger settings you can ignore this change.

```
// [...nextauth.js]
import log from "some-logger-service"
...
logger: {
- error(code, ...message) {},
+ error(code, metadata) {},
- warn(code, ...message) {},
+ warn(code) {}
- debug(code, ...message) {}
+ debug(code, metadata) {}
}
```

Introduced in https://github.com/nextauthjs/next-auth/releases/tag/v4.0.0-next.19

`nodemailer`

Like `typeorm` and `prisma`, `nodemailer` is no longer included as a dependency by default. If you are using the Email provider you must install it in your project manually, or use any other Email library in the `sendVerificationRequest` callback. This reduces bundle size for those not actually using the Email provider. Remember, when using the Email provider, it is mandatory to also use a database adapter due to the fact that verification tokens need to be persisted longer term for the magic link functionality to work.

Introduced in https://github.com/nextauthjs/next-auth/releases/tag/v4.0.0-next.2

Theme

We have added some basic customization options to our built-in pages like `signin`, `signout`, etc.

These can be set under the `theme` configuration key. This used to be a string which only controlled the color scheme option. Now it is an object with the following options:

```
theme: {
  colorScheme: "auto", // "auto" | "dark" | "light"
  brandColor: "", // Hex color value
  logo: "" // Absolute URL to logo image
}
```

The hope is that with some minimal configuration / customization options, users won't immediately feel the need to replace the built-in pages with their own.

More details and screenshots of the new theme options can be found under **configuration/pages**.

Introduced in #2788

Session

The `session.jwt: boolean` option has been renamed to `session.strategy: "jwt" | "database"`. The goal is to make the user's options more intuitive:

1. No adapter, `strategy: "jwt"`: This is the default. The session is saved in a cookie and never persisted anywhere.
2. With Adapter, `strategy: "database"`: If an Adapter is defined, this will be the implicit setting. No user config is needed.
3. With Adapter, `strategy: "jwt"`: The user can explicitly instruct `next-auth` to use JWT even if a database is available. This can result in faster lookups in compromise of lowered security. Read more about: **https://next-auth.js.org/faq#json-web-tokens**

Example:

```
session: {
- jwt: true,
+ strategy: "jwt",
}
```

Introduced in #3144

Adapters

Most importantly, the core `next-auth` package no longer ships with `typeorm` or any other database adapter by default. This brings the default bundle size down significantly for those not needing to persist user data to a database.

You can find the official Adapters in the `packages` directory in the primary monorepo (**nextauthjs/next-auth**). Although you can still **create your own** with a new, **simplified Adapter API**.

If you have a database that was created with a `3.x.x` or earlier version of NextAuth.js, you will need to run a migration to update the schema to the new version 4 database model. See the bottom of this migration guide for database specific migration examples.

1. If you use the built-in TypeORM or Prisma adapters, these have been removed from the core `next-auth` package. Thankfully the migration is easy; you just need to install the external packages for your database and change the import in your `[...nextauth].js`.

The `database` option has been removed, you must now do the following instead:

```
// [...nextauth].js
import NextAuth from "next-auth"
+ import { TypeORMLegacyAdapter } from "@next-auth/typeorm-legacy-adapter"

...
export default NextAuth({
-  database: "yourconnectionstring",
+  adapter: TypeORMLegacyAdapter("yourconnectionstring")
})
```

2. The `prisma-legacy` adapter has been removed, please use the `@next-auth/prisma-adapter` instead.

3. The `typeorm-legacy` adapter has been upgraded to use the newer adapter API, but has retained the `typeorm-legacy` name. We aim to migrate this to individual lighter weight adapters for each database type in the future, or switch out `typeorm`.

4. MongoDB has been moved to its own adapter under `@next-auth/mongodb-adapter`. See the MongoDB Adapter docs.

Introduced in https://github.com/nextauthjs/next-auth/releases/tag/v4.0.0-next.8 and #2361

Adapter API

This does not require any changes from the user - these are adapter specific changes only

The Adapter API has been rewritten and significantly simplified in NextAuth.js v4. The adapters now have less work to do as some functionality has been migrated to the core of NextAuth, like hashing the verification token.

If you are an adapter maintainer or are interested in writing your own adapter, you can find more information about this change in #2361 and release https://github.com/nextauthjs/next-auth/releases/tag/v4.0.0-next.22.

Schema changes

The way we save data with adapters have slightly changed. With the new Adapter API, we wanted to make it easier to extend your database with additional fields. For example if your User needs an extra `phone` field, it should be enough to add that to your database's schema, and no changes will be necessary in your adapter.

- `created_at`/`createdAt` and `updated_at`/`updatedAt` fields are removed from all Models.
- `user_id`/`userId` consistently named `userId`.
- `compound_id`/`compoundId` is removed from Account.
- `access_token`/`accessToken` is removed from Session.
- `email_verified`/`emailVerified` on User is consistently named `emailVerified`.
- `provider_id`/`providerId` renamed to `provider` on Account
- `provider_type`/`providerType` renamed to `type` on Account
- `provider_account_id`/`providerAccountId` on Account is consistently named `providerAccountId`
- `access_token_expires`/`accessTokenExpires` on Account renamed to `expires_at`
- New fields on Account: `token_type`, `scope`, `id_token`, `session_state`
- `verification_requests` table has been renamed to `verification_tokens`

▸ See the changes

For more info, see the Models page.

Database migration

NextAuth.js v4 has a slightly different database schema compared to v3. If you're using any of our adapters and want to upgrade, you can use on of the below schemas.

They are designed to be run directly against the database itself. So instead of having one in Prisma syntax, one in TypeORM syntax, etc. we've decided to just make one for each underlying database type. i.e. one for Postgres, one for MySQL, one for MongoDB, etc.

MySQL

```
/* ACCOUNT */
ALTER TABLE accounts
CHANGE "access_token_expires" "expires_at" int
CHANGE "user_id" "userId" varchar(255)
ADD CONSTRAINT fk_user_id FOREIGN KEY (userId) REFERENCES users(id)
RENAME COLUMN "provider_id" "provider"
RENAME COLUMN "provider_account_id" "providerAccountId"
DROP COLUMN "provider_type"
DROP COLUMN "compound_id"
/* The following two timestamp columns have never been necessary for NextAuth.js
to function, but can be kept if you want */
DROP COLUMN "created_at"
DROP COLUMN "updated_at"

ADD COLUMN "token_type" varchar(255) NULL
ADD COLUMN "scope" varchar(255) NULL
ADD COLUMN "id_token" varchar(255) NULL
ADD COLUMN "session_state" varchar(255) NULL

/* Note: These are only needed if you're going to be using the old Twitter OAuth
1.0 provider. */
ADD COLUMN "oauth_token_secret" varchar(255) NULL
ADD COLUMN "oauth_token" varchar(255) NULL

/* USER */
```

```
ALTER TABLE users
RENAME COLUMN "email_verified" "emailVerified"
/* The following two timestamp columns have never been necessary for NextAuth.js
to function, but can be kept if you want */
DROP COLUMN "created_at"
DROP COLUMN "updated_at"

/* SESSION */
ALTER TABLE sessions
RENAME COLUMN "session_token" "sessionToken"
CHANGE "user_id" "userId" varchar(255)
ADD CONSTRAINT fk_user_id FOREIGN KEY (userId) REFERENCES users(id)
DROP COLUMN "access_token"
/* The following two timestamp columns have never been necessary for NextAuth.js
to function, but can be kept if you want */
DROP COLUMN "created_at"
DROP COLUMN "updated_at"

/* VERIFICATION REQUESTS */
ALTER TABLE verification_requests RENAME verification_tokens
ALTER TABLE verification_tokens
DROP COLUMN id
/* The following two timestamp columns have never been necessary for NextAuth.js
to function, but can be kept if you want */
DROP COLUMN "created_at"
DROP COLUMN "updated_at"
```

Postgres

```
/* ACCOUNT */
ALTER TABLE accounts RENAME COLUMN "user_id" TO "userId";
ALTER TABLE accounts RENAME COLUMN "provider_id" TO "provider";
ALTER TABLE accounts RENAME COLUMN "provider_account_id" TO "providerAccountId";
ALTER TABLE accounts RENAME COLUMN "access_token_expires" TO "expires_at";
ALTER TABLE accounts RENAME COLUMN "provider_type" TO "type";

/* Do conversion of TIMESTAMPTZ to BIGINT */
ALTER TABLE accounts ALTER COLUMN "expires_at" TYPE TEXT USING
CAST(extract(epoch FROM "expires_at") AS BIGINT)*1000;

/* Keep id as SERIAL with autoincrement when using ORM. Using new v4 uuid format
won't work because of incompatibility */
/* ALTER TABLE accounts ALTER COLUMN "id" TYPE TEXT; */
```

```
/* ALTER TABLE accounts ALTER COLUMN "userId" TYPE TEXT; */
ALTER TABLE accounts ALTER COLUMN "type" TYPE TEXT;
ALTER TABLE accounts ALTER COLUMN "provider" TYPE TEXT;
ALTER TABLE accounts ALTER COLUMN "providerAccountId" TYPE TEXT;

ALTER TABLE accounts ADD CONSTRAINT fk_user_id FOREIGN KEY ("userId") REFERENCES
users(id);
ALTER TABLE accounts
DROP COLUMN IF EXISTS "compound_id";
/* The following two timestamp columns have never been necessary for NextAuth.js
to function, but can be kept if you want */
ALTER TABLE accounts
DROP COLUMN IF EXISTS "created_at",
DROP COLUMN IF EXISTS "updated_at";

ALTER TABLE accounts
ADD COLUMN IF NOT EXISTS "token_type" TEXT NULL,
ADD COLUMN IF NOT EXISTS "scope" TEXT NULL,
ADD COLUMN IF NOT EXISTS "id_token" TEXT NULL,
ADD COLUMN IF NOT EXISTS "session_state" TEXT NULL;
/* Note: These are only needed if you're going to be using the old Twitter OAuth
1.0 provider. */
/* ALTER TABLE accounts
ADD COLUMN IF NOT EXISTS "oauth_token_secret" TEXT NULL,
ADD COLUMN IF NOT EXISTS "oauth_token" TEXT NULL; */

/* USER */
ALTER TABLE users RENAME COLUMN "email_verified" TO "emailVerified";

/* Keep id as SERIAL with autoincrement when using ORM. Using new v4 uuid format
won't work because of incompatibility */
/* ALTER TABLE users ALTER COLUMN "id" TYPE TEXT; */
ALTER TABLE users ALTER COLUMN "name" TYPE TEXT;
ALTER TABLE users ALTER COLUMN "email" TYPE TEXT;
ALTER TABLE users ALTER COLUMN "image" TYPE TEXT;
/* Do conversion of TIMESTAMPTZ to BIGINT and then TEXT */
ALTER TABLE users ALTER COLUMN "emailVerified" TYPE TEXT USING
CAST(CAST(extract(epoch FROM "emailVerified") AS BIGINT)*1000 AS TEXT);
/* The following two timestamp columns have never been necessary for NextAuth.js
to function, but can be kept if you want */
ALTER TABLE users
DROP COLUMN IF EXISTS "created_at",
DROP COLUMN IF EXISTS "updated_at";
```

```
/* SESSION */
ALTER TABLE sessions RENAME COLUMN "session_token" TO "sessionToken";
ALTER TABLE sessions RENAME COLUMN "user_id" TO "userId";

/* Keep id as SERIAL with autoincrement when using ORM. Using new v4 uuid format
won't work because of incompatibility */
/* ALTER TABLE sessions ALTER COLUMN "id" TYPE TEXT; */
/* ALTER TABLE sessions ALTER COLUMN "userId" TYPE TEXT; */
ALTER TABLE sessions ALTER COLUMN "sessionToken" TYPE TEXT;
ALTER TABLE sessions ADD CONSTRAINT fk_user_id FOREIGN KEY ("userId") REFERENCES
users(id);
/* Do conversion of TIMESTAMPTZ to BIGINT and then TEXT */
ALTER TABLE sessions ALTER COLUMN "expires" TYPE TEXT USING
CAST(CAST(extract(epoch FROM "expires") AS BIGINT)*1000 AS TEXT);
ALTER TABLE sessions DROP COLUMN IF EXISTS "access_token";
/* The following two timestamp columns have never been necessary for NextAuth.js
to function, but can be kept if you want */
ALTER TABLE sessions
DROP COLUMN IF EXISTS "created_at",
DROP COLUMN IF EXISTS "updated_at";

/* VERIFICATION REQUESTS */
ALTER TABLE verification_requests RENAME TO verification_tokens;
/* Keep id as ORM needs it */
/* ALTER TABLE verification_tokens DROP COLUMN IF EXISTS id; */
ALTER TABLE verification_tokens ALTER COLUMN "identifier" TYPE TEXT;
ALTER TABLE verification_tokens ALTER COLUMN "token" TYPE TEXT;
/* Do conversion of TIMESTAMPTZ to BIGINT and then TEXT */
ALTER TABLE verification_tokens ALTER COLUMN "expires" TYPE TEXT USING
CAST(CAST(extract(epoch FROM "expires") AS BIGINT)*1000 AS TEXT);
/* The following two timestamp columns have never been necessary for NextAuth.js
to function, but can be kept if you want */
ALTER TABLE verification_tokens
DROP COLUMN IF EXISTS "created_at",
DROP COLUMN IF EXISTS "updated_at";
```

MongoDB

MongoDB is a document database and as such new fields will be automatically populated. You do, however, need to update the names of existing fields which are going to be reused.

```
db.getCollection('accounts').updateMany({}, {
  $rename: {
    "provider_id": "provider",
    "provider_account_id": "providerAccountId",
    "user_id": "userId",
    "access_token_expires": "expires_at"
  }
})
db.getCollection('users').updateMany({}, {
  $rename: {
    "email_verified": "emailVerified"
  }
})
db.getCollection('sessions').updateMany({}, {
  $rename: {
    "session_token": "sessionToken",
    "user_id": "userId"
  }
})
```

Missing `secret`

NextAuth.js used to generate a secret for convenience, when the user did not define one. This might have been useful in development, but can be a concern in production. We have always been clear about that in the docs, but from now on, if you forget to define a `secret` property in production, we will show the user an error page. Read more about this option **here**

You can generate a secret to be placed in the `secret` configuration option via the following command:

```
$ openssl rand -base64 32
```

Therefore, your NextAuth.js config should look something like this:

/pages/api/auth/[...nextauth].js

```
...
export default NextAuth({
  ...
  providers: [...],
  secret: "LlKq6ZtYbr+hTC073mAmAh9/h2HwMfsFo4hrfCx5mLg=",
  ...
})
```

Introduced in #3143

Session `strategy`

We have always supported two different session strategies. The first being our most popular and default strategy - the JWT based one. The second is the database adapter persisted session strategy. Both have their advantages/disadvantages, you can learn more about them on the FAQ page.

Previously, the way you configured this was through the `jwt: boolean` flag in the `session` option. The names `session` and `jwt` might have been a bit overused in the options, and so for a clearer message, we renamed this option to `strategy: "jwt" | "database"`, it is still in the `session` object. This will hopefully better indicate the purpose of this option as well as make very explicit which type of session you are going to use.

See the `session` option docs for more details.

Introduced in #3144

Summary

We hope this migration goes smoothly for each and every one of you! If you have any questions or get stuck anywhere, feel free to create a new issue on GitHub.

Edit this page

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