Platform Express

September 14, 2026

Platform Express

What it is responsible for

Platform Express manages the Express-oriented integration surface whose stated work entry is `MulterModule`. Its named members place `MulterModule` alongside `FileInterceptor`, `FilesInterceptor`, `AnyFilesInterceptor`, `FileFieldsInterceptor`, and `NoFilesInterceptor`; together, those symbols identify the subsystem's interception-facing API. The available evidence does not state the behavior, configuration, or output of these members, so their documented responsibility should stop at that boundary. In the surrounding project material, the authors describe Nest as a framework for building Node.js server-side applications and direct readers to the guide at docs.nestjs.com.

What it refuses

Platform Express refuses work when `!next` and the HTTP adapter does not support filtering on version. The recorded refusal condition is: "HTTP adapter does not support filtering on version." No other rejection message or condition is supplied by the evidence.

What it needs, and who needs it

Platform Express depends on Common and Core. `MulterModule` is the declared work entry within that dependency context. The `sample` subsystem depends on Platform Express; without Platform Express, `sample` cannot satisfy that declared dependency. The evidence does not say which Common or Core contracts are required, nor does it identify a more specific failure mode for `sample`.

Notable members

`MulterModule` is the entry point for work in Platform Express. `FileInterceptor` and `FilesInterceptor` are named interception members; their names distinguish singular and plural forms, but the evidence gives no operational description. These three are notable because the entry point is identified and the two interceptors are named. The evidence does not rank the remaining members, specify configuration, or describe outputs. Project documentation covers questions, issues, consulting, support, sponsors, and license material; those comments do not establish responsibilities.

52 entities in `packages/platform-express`. Nothing else in this repository depends on it.

What it is made of

Its 52 entities sit in 19 files under `packages/platform-express`: 29 doc comments, 7 functions, 7 interfaces, 4 constants and 5 more.
`files-upload-module.interface.ts` holds 3 of them — more than any other file here.
`ExpressAdapter` declares 36 methods, the widest surface here.

Where work enters

- `MulterModule` — `packages/platform-express/multer/multer.module.ts`:14

How it refuses and fails

1 of its components records a refusal or a failure handler.
It refuses work outright, under a condition written into the component itself.
Its `catch` blocks handle a failure that already happened in 2 places.

Boundaries

It depends on `Common`, `Core`, and on nothing else in this repository.

How this code is named

These conventions cover most of the codebase. Learning them is faster than reading an index —
each one lets you find any member of its family without looking it up.

Pattern	Where	Count	Examples
`*.interface.ts`	across the repository	6	`multer-options.interface.ts`, `files-upload-`

Pattern	Where	Count	Examples
			`module.interface.ts`, `serve-static-options.interface.ts`, `nest-express-application.interface.ts`
`*.interceptor.ts`	`platform-express/multer/interceptors/`	5	`file.interceptor.ts`, `files.interceptor.ts`, `no-files.interceptor.ts`, `any-files.interceptor.ts`