ParseUUIDPipeOptions

August 17, 2026

ParseUUIDPipeOptions

Kind: Interface

Source: `packages/common/pipes/parse-uuid.pipe.ts`

Part of: Common

`ParseUUIDPipeOptions` configures the behavior of NestJS's `ParseUUIDPipe`, which validates incoming values as UUIDs. It lets callers restrict accepted UUID versions, customize validation errors, provide a custom exception factory, and allow optional values to bypass validation.

Properties

Property	Type
`version`	`'3'
`errorHttpStatusCode`	`ErrorHttpStatusCode`
`exceptionFactory`	`(errors: string) => any`
`optional`	`boolean`

Diagram

```
graph LR
  A[Incoming route value] --> B[ParseUUIDPipe]
  B --> C{Value is optional<br/>and empty?}
  C -->|Yes| D[Return value unchanged]
  C -->|No| E{Valid UUID?}
  E -->|Yes| F[Return validated UUID]
  E -->|No| G[Create validation exception]

  H[ParseUUIDPipeOptions] --> B
  H --> I[version: 3 | 4 | 5 | 7]
  H --> J[optional: boolean]
```

```
  H --> K[errorHttpStatusCode]
  H --> L[exceptionFactory]
  K --> G
  L --> G
```

Usage

```
import {
  BadRequestException,
  ParseUUIDPipe,
} from '@nestjs/common';

const uuidPipe = new ParseUUIDPipe({
  version: '4',
  optional: false,
  exceptionFactory: (errors) =>
    new BadRequestException({
      message: 'Invalid user identifier',
      errors,
    }),
});

// Example controller usage
@Get(':id')
findOne(
  @Param('id', uuidPipe) id: string,
) {
  return this.usersService.findOne(id);
}
```

AI Coding Instructions

- Use `version` to enforce the UUID format expected by the API, such as `'4'` for randomly generated identifiers.
- Set `optional: true` only for parameters that may legitimately be absent; valid non-empty values must still be UUIDs.
- Prefer `exceptionFactory` when the application requires a shared or custom error response format.
- Use `errorHttpStatusCode` for standard HTTP error customization when a custom exception body is unnecessary.
- Apply `ParseUUIDPipe` at controller boundaries (`@Param`, `@Query`, or `@Body`) before passing identifiers to services or repositories.

How it works

Type

`ParseUUIDPipeOptions` is the optional configuration interface accepted by `ParseUUIDPipe`'s constructor. The pipe stores these options and reads `version`, `exceptionFactory`, `errorHttpStatusCode`, and `optional` during construction or transformation. `packages/common/pipes/parse-uuid.pipe.ts:17-38` `packages/common/pipes/parse-uuid.pipe.ts:59-71`

Members

- `version?: '3' | '4' | '5' | '7'` selects the UUID version pattern checked by the pipe. If omitted, the pipe checks against its general UUID pattern instead. `packages/common/pipes/parse-uuid.pipe.ts:19-21` `packages/common/pipes/parse-uuid.pipe.ts:67` `packages/common/pipes/parse-uuid.pipe.ts:77` `packages/common/pipes/parse-uuid.pipe.ts:87-92`
- `errorHttpStatusCode?: ErrorHttpStatusCode` selects the status-specific exception class used when no `exceptionFactory` is supplied; it defaults to `HttpStatus.BAD_REQUEST` (`400`). `packages/common/pipes/parse-uuid.pipe.ts:23-25` `packages/common/pipes/parse-uuid.pipe.ts:61-70` `packages/common/enums/http-status.enum.ts:26` The `ErrorHttpStatusCode` type is limited to the status codes mapped in `HttpErrorByCode`, including bad request, unauthorized, forbidden, not found, conflict, several gateway/server errors, and others listed in its union. `packages/common/utils/http-error-by-code.util.ts:27-48` `packages/common/utils/http-error-by-code.util.ts:50-72`
- `exceptionFactory?: (errors: string) => any` replaces the default exception creation function. On invalid input, the pipe calls this function with either `"The value passed as UUID is not a string"` or a version-aware `"Validation failed (uuid ... is expected)"` message, then throws its return value. `packages/common/pipes/parse-uuid.pipe.ts:27-32` `packages/common/pipes/parse-uuid.pipe.ts:68-70` `packages/common/pipes/parse-uuid.pipe.ts:77-82` `packages/common/pipes/parse-uuid.pipe.ts:87-90`
- `optional?: boolean` changes handling only for `null` and `undefined`: when it is truthy, `transform()` returns either value unchanged without UUID checking. `packages/common/pipes/parse-uuid.pipe.ts:33-37` `packages/common/pipes/parse-uuid.pipe.ts:73-76` `packages/common/utils/shared.utils.ts:48-49`

Validation behavior controlled by the options

The accepted UUID patterns require hyphenated hexadecimal text with `8-4-4-4-12` groups. Version-specific patterns require `3`, `4`, `5`, or `7` at the beginning of the third group; versions 4, 5, and 7 also require `8`, `9`, `A`, or `B` at the beginning of the fourth group. Matching is case-insensitive. `packages/common/pipes/parse-uuid.pipe.ts:49-55`

Except for the `optional` null/undefined path, a value must be a string before pattern matching. Non-string values cause the configured exception factory's result to be thrown. `packages/common/pipes/parse-uuid.pipe.ts:73-84` `packages/common/pipes/parse-uuid.pipe.ts:87-92` `packages/common/utils/shared.utils.ts:45`