VersionOptions

August 17, 2026

VersionOptions

Kind: Interface

Source: `packages/common/interfaces/version-options.interface.ts`

Part of: Common

`VersionOptions` defines the version configuration used by components that need to target or describe a specific API or application version. It provides a single `version` property typed as `VersionValue`, ensuring version values remain consistent across the system.

Properties

Property	Type
`version`	`VersionValue`

Diagram

```
graph LR
  A[Consumer Configuration] --> B[VersionOptions]
  B --> C[version: VersionValue]
  C --> D[Version-Aware Component]
```

Usage

```
import type { VersionOptions } from '@package/common';

const options: VersionOptions = {
  version: 'v1',
};

function configureVersion(options: VersionOptions) {
```

```
  console.log(`Using version: ${options.version}`);
}

configureVersion(options);
```

AI Coding Instructions

- Provide a valid `VersionValue` when constructing `VersionOptions`; do not use arbitrary strings unless they are supported by that type.
- Use `VersionOptions` for function parameters and configuration objects that require version-specific behavior.
- Keep version handling centralized through this interface rather than introducing duplicate `version` field definitions.
- Import `VersionOptions` as a type-only import when it is used exclusively for TypeScript type checking.

How it works

`VersionOptions` is a public TypeScript interface for attaching an optional API version to a controller. It declares one property, `version?: VersionValue`. `packages/common/interfaces/version-options.interface.ts:21-32`

- `version` may be a string, the `VERSION_NEUTRAL` symbol, or an array containing strings and/or that symbol. `packages/common/interfaces/version-options.interface.ts:8-16`
- `VERSION_NEUTRAL` denotes routes that work with any request version, including no version. `packages/common/interfaces/version-options.interface.ts:3-8`
- `ControllerOptions` extends `VersionOptions`, so it can be passed as `@Controller({ version: ... })`. `packages/common/decorators/core/controller.decorator.ts:8-16` The decorator stores this value under `VERSION_METADATA`; when the value is an array, it removes duplicate entries first. `packages/common/decorators/core/controller.decorator.ts:151-177`
- A method-level version takes precedence over the controller-level version during route construction. `packages/core/router/route-path-factory.ts:80-84`
- Version-aware routing runs only when a route has either a method or controller version **and** application versioning options exist. Non-URI strategies are passed to the HTTP adapter's version filter. `packages/core/router/router-explorer.ts:192-208`
- With URI versioning, string versions become path segments using the configured prefix (default `v`); neutral versions add no version segment.

`packages/core/router/route-path-factory.ts:27-52` `packages/core/router/route-path-factory.ts:86-95`

- The interface itself contains no runtime validation, error handling, or side effects; it is only a type declaration. `packages/common/interfaces/version-options.interface.ts:21-32`