VersionAccessGuard

September 4, 2026

VersionAccessGuard

Kind: Service

Source: `atloria-monorepo/apps/api/src/doc-version/version-access.guard.ts`

`VersionAccessGuard` is a NestJS authorization guard that determines whether the current request is allowed to access a document version. It runs before protected route handlers, using request context and document-version ownership or permission data to allow the request or reject it with an authorization error.

Methods

Method	Signature	Returns
`canActivate`	`canActivate(context: ExecutionContext)`	`Promise<boolean>`

Dependencies

- `PrismaService`

Where it refuses work

- `VersionAccessGuard` stops the work with `ForbiddenException` when `!request.user` — “Authentication required”, in 2 places.
- `VersionAccessGuard` stops the work with `ForbiddenException` when `!pwd` — “Password required”.
- `VersionAccessGuard` stops the work with `ForbiddenException` when `pwd !== access.password` — “Invalid password”.
- `VersionAccessGuard` stops the work with an early return when `!versionId`.
- `VersionAccessGuard` stops the work with an early return when `!access || access.accessType === 'PUBLIC'`.

- `VersionAccessGuard` stops the work with an early return when `access.allowedUserIds.includes(userId)`.

Diagram

```
sequenceDiagram
  participant Client
  participant Controller
  participant Guard as VersionAccessGuard
  participant Service as Document Version Service
  participant Handler as Route Handler

  Client->>Controller: Request protected version endpoint
  Controller->>Guard: canActivate(context)
  Guard->>Guard: Read authenticated user and route parameters
  Guard->>Service: Resolve version and verify access
  Service-->>Guard: Access result
  alt Access granted
    Guard-->>Controller: true
    Controller->>Handler: Execute handler
    Handler-->>Client: Return version data
  else Access denied
    Guard-->>Controller: false / throw exception
    Controller-->>Client: 403 Forbidden
  end
```

Usage

```
import { Controller, Get, Param, UseGuards } from '@nestjs/common';
import { VersionAccessGuard } from './version-access.guard';

@Controller('document-versions')
export class DocumentVersionController {
  @Get(':versionId')
  @UseGuards(VersionAccessGuard)
  async getVersion(@Param('versionId') versionId: string) {
    // The guard has already verified that the authenticated user
    // can access this document version.
    return {
      id: versionId,
    };
  }
}
```

AI Coding Instructions

- Apply `VersionAccessGuard` with `@UseGuards()` on every endpoint that exposes, modifies, or deletes protected document-version data.
- Keep authorization logic inside the guard or delegated domain services; controllers should assume access has already been validated.
- Read authenticated user data from the NestJS execution context consistently with the application's authentication strategy.
- Ensure route parameter names used by the guard match the controller route, such as `:versionId`.
- Return `true` only after verifying access; use NestJS authorization exceptions for denied access when detailed error handling is required.

Relationships

- DEPENDS_ON → `PrismaService`

Referenced By

- `DocVersionModule` (MODULE_PROVIDES)
- `DocVersionModule` (MODULE_EXPORTS)