DocVersionRetentionService

September 5, 2026

DocVersionRetentionService

Kind: Service

Source: `atloria-monorepo/apps/api/src/doc-version/retention.service.ts`

`DocVersionRetentionService` manages retention enforcement for document version history in the API. It runs retention cleanup across documents or policies, integrating with NestJS module lifecycle hooks to initialize and stop any scheduled enforcement work safely.

Methods

Method	Signature	Returns
`onModuleInit`	`onModuleInit()`	`unknown`
`onModuleDestroy`	`onModuleDestroy()`	`unknown`
`enforceAll`	`enforceAll()`	`unknown`
`enforcePolicy`	`enforcePolicy(projectId: string, policy: RetentionPolicy)`	`unknown`

Dependencies

- `PrismaService`
- `AuditService` *(optional)*

Where it refuses work

- `DocVersionRetentionService` stops the work with an early return when `totalVersions <= keepMin`.

When something fails

- `DocVersionRetentionService` handles failure in 1 place: it logs it and continues in all 1.

Diagram

```
sequenceDiagram
  participant Nest as NestJS Application
  participant Service as DocVersionRetentionService
  participant Store as Document Version Storage

  Nest->>Service: onModuleInit()
  Service->>Service: Initialize retention enforcement

  Note over Service: Scheduled or manual enforcement
  Service->>Service: enforceAll()
  Service->>Service: enforcePolicy()
  Service->>Store: Find versions exceeding retention rules
  Service->>Store: Remove eligible old versions

  Nest->>Service: onModuleDestroy()
  Service->>Service: Stop cleanup work and release resources
```

Usage

```
import { Injectable } from '@nestjs/common';
import { DocVersionRetentionService } from './doc-version/retention.service';

@Injectable()
export class DocumentMaintenanceService {
  constructor(
    private readonly retentionService: DocVersionRetentionService,
  ) {}

  async runRetentionCleanup(): Promise<void> {
    // Enforce retention rules for all configured document version policies.
    await this.retentionService.enforceAll();
  }
}
```

AI Coding Instructions

- Use `enforceAll()` for application-wide cleanup; keep policy-specific enforcement inside `enforcePolicy()`.
- Let NestJS invoke `onModuleInit()` and `onModuleDestroy()` through dependency injection rather than calling lifecycle hooks manually.

- Ensure retention deletion logic only removes versions that are eligible under the configured policy and preserves required current versions.
- Keep enforcement operations safe to rerun, since scheduled jobs or manual maintenance tasks may overlap.
- When changing retention behavior, verify integrations with document-version persistence and any background scheduling mechanism.

Relationships

- DEPENDS_ON → `PrismaService`
- DEPENDS_ON → `AuditService`

Referenced By

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