DocumentVersionService

September 5, 2026

DocumentVersionService

Kind: Service

Source: `atloria-monorepo/apps/api/src/document/services/document-version.service.ts`

`DocumentVersionService` manages version history for documents in the API layer. It creates immutable snapshots, retrieves and updates version metadata, restores documents from a selected version, and handles version deletion and counting.

Methods

Method	Signature	Returns	Description
`createVersion`	`createVersion(documentId: string, dto: CreateVersionDto, userId: string)`	`Promise<DocumentVersion>`	Create a named version of a document
`getVersions`	`getVersions(documentId: string)`	`Promise<DocumentVersion[]>`	Get all versions for a document
`getVersion`	`getVersion(documentId: string, versionNumber: number)`	`Promise<DocumentVersion>`	Get a specific version by version number
`updateVersionMetadata`	`updateVersionMetadata(versionId: string, dto: UpdateVersionMetadataDto)`	`Promise<DocumentVersion>`	Update version metadata (label/comment)
`deleteVersion`	`deleteVersion(versionId: string)`	`Promise<void>`	Delete a version
`restoreFromVersion`	`restoreFromVersion(documentId: string, versionNumber: number, userId: string)`	`Promise<any>`	Restore document from a named version
`getVersionCount`	`getVersionCount(documentId: string)`	`Promise<number>`	Get version count for a document

Dependencies

- `PrismaService`

Where it refuses work

- `DocumentVersionService` stops the work with `NotFoundException` when `!document` — “Document not found”, in 2 places.
- `DocumentVersionService` stops the work with `NotFoundException` when `!version`.
- `DocumentVersionService` stops the work with `NotFoundException` when `!existing` — “Version not found”.
- `DocumentVersionService` stops the work with `NotFoundException` when `!version` — “Version not found”.
- `DocumentVersionService` stops the work with `BadRequestException` when `versionCount === 1` — “Cannot delete the only version”.
- `DocumentVersionService` stops the work with `BadRequestException` when `version.documentId !== documentId` — “Version does not belong to this document”.

Diagram

```
sequenceDiagram
  participant Client
  participant Controller
  participant VersionService as DocumentVersionService
  participant Repository
  participant DocumentService

  Client->>Controller: Create or restore document version
  Controller->>VersionService: createVersion() / restoreFromVersion()
  VersionService->>Repository: Read/write DocumentVersion records

  alt Create version
    VersionService->>DocumentService: Read current document state
    DocumentService-->>VersionService: Current document content
    VersionService->>Repository: Persist version snapshot
    Repository-->>VersionService: DocumentVersion
  else Restore version
    VersionService->>Repository: Fetch requested version
    Repository-->>VersionService: Stored version snapshot
    VersionService->>DocumentService: Apply snapshot to document
  end

  VersionService-->>Controller: Version result
  Controller-->>Client: API response
```

Usage

```
import { Injectable, NotFoundException } from '@nestjs/common';
import { DocumentVersionService } from './document-version.service';

@Injectable()
export class DocumentHistoryController {
  constructor(
```

```
    private readonly documentVersionService: DocumentVersionService,
  ) {}

  async createSnapshot(documentId: string, userId: string) {
    return this.documentVersionService.createVersion(
      documentId,
      userId,
    );
  }

  async listVersions(documentId: string) {
    return this.documentVersionService.getVersions(documentId);
  }

  async restoreVersion(documentId: string, versionId: string, userId: string) {
    const version = await this.documentVersionService.getVersion(
      documentId,
      versionId,
    );

    if (!version) {
      throw new NotFoundException('Document version not found');
    }

    return this.documentVersionService.restoreFromVersion(
      documentId,
      versionId,
      userId,
    );
  }
}
```

AI Coding Instructions

- Create versions from a complete document snapshot so restored content is independent of later document updates.
- Always scope version lookups, updates, and deletions to the parent document ID to prevent cross-document access.
- Preserve authorization checks at the controller or calling-service boundary before creating, restoring, or deleting versions.
- Treat `restoreFromVersion()` as a document mutation; ensure it uses the same validation, persistence, and audit patterns as normal document updates.
- Keep version metadata updates separate from immutable snapshot content updates unless the intended behavior explicitly permits content mutation.

Relationships

- DEPENDS_ON → `PrismaService`

Referenced By

- `DocumentController` (DEPENDS_ON)
- `DocumentModule` (MODULE_PROVIDES)
- `DocumentModule` (MODULE_EXPORTS)
- `DocumentService` (DEPENDS_ON)
- `SyncService` (DEPENDS_ON)