# ContentVisibilityService

September 4, 2026

## ContentVisibilityService

**Kind:** Service

**Source:** `atloria-monorepo/apps/api/src/project/content-visibility.service.ts`

ContentVisibilityService (D4 owner side) — manage per-document/category visibility rules,
the project's ReaderIdentityConfig, and mint owner "Preview as…" reader tokens. Org ownership is verified from the caller's JWT org (never from params — cross-tenant lesson).
A rule change invalidates the visibility cache and re-stamps the RAG chunks so segmentation
takes effect in chat immediately (metadata restamp — no re-embedding).

`ContentVisibilityService` manages owner-side visibility controls for documents and categories within a project. It also maintains the project’s `ReaderIdentityConfig` and mints short-lived “Preview as…” reader tokens, always deriving organization ownership from the caller’s JWT rather than request parameters. Rule updates invalidate cached visibility decisions and re-stamp RAG chunk metadata so chat segmentation reflects changes without re-embedding content.

### Methods

| Method | Signature | Returns | Description |
| --- | --- | --- | --- |
| `setDocumentRule` | `setDocumentRule(projectId: string, docId: string, rule: unknown, user: JwtPayload)` | `unknown` | |
| `setCategoryRule` | `setCategoryRule(projectId: string, categoryId: string, rule: unknown, user: JwtPayload)` | `unknown` | |

| Method | Signature | Returns | Description |
| --- | --- | --- | --- |
| `listRules` | `listRules(projectId: string, user: JwtPayload)` | `unknown` | Overview of everything with a rule (owner settings list). |
| `getReaderIdentity` | `getReaderIdentity(projectId: string, user: JwtPayload)` | `unknown` | |
| `updateReaderIdentity` | `updateReaderIdentity(projectId: string, dto: UpdateReaderIdentityDto, user: JwtPayload)` | `unknown` | |
| `mintPreviewToken` | `mintPreviewToken(projectId: string, attrs: Record<string, unknown> | undefined, user: JwtPayload)` | `Promise<{ token: string; attrs: Record<string, unknown>; expiresInSeconds: number }>` |

## Dependencies

- `PrismaService`
- `DocsVisibilityService`
- `ReaderTokenService`
- `TechDocsRagService`

## Where it refuses work

- `ContentVisibilityService` stops the work with `NotFoundException` when `!project` — “Project not found”.
- `ContentVisibilityService` stops the work with `ForbiddenException` when `project.organizationId !== user.organizationId` — “Access denied to this project”.
- `ContentVisibilityService` stops the work with `BadRequestException` when `problems.length`.
- `ContentVisibilityService` stops the work with `NotFoundException` when `!doc` — “Document not found”.
- `ContentVisibilityService` stops the work with `NotFoundException` when `!cat` — “Category not found”.

- `ContentVisibilityService` stops the work with `Error` when `u.protocol !== 'https:'` — “not https”.

## When something fails

- `ContentVisibilityService` handles failure in 1 place: it lets it reach the caller in all 1.

## Diagram

```
sequenceDiagram
    participant Owner as Project Owner
    participant API as NestJS Controller
    participant Service as ContentVisibilityService
    participant JWT as Caller JWT
    participant Rules as Visibility Rules Store
    participant Cache as Visibility Cache
    participant RAG as RAG Chunk Metadata
    participant Tokens as Preview Token Service

    Owner->>API: Set document/category visibility rule
    API->>Service: setDocumentRule() / setCategoryRule()
    Service->>JWT: Read authenticated org identity
    JWT-->>Service: orgId
    Service->>Rules: Verify project belongs to orgId
    Service->>Rules: Persist visibility rule
    Service->>Cache: Invalidate project visibility cache
    Service->>RAG: Re-stamp chunk visibility metadata
    Service-->>API: Updated rule result
    API-->>Owner: Rule updated

    Owner->>API: Preview as reader identity
    API->>Service: mintPreviewToken()
    Service->>JWT: Read authenticated org identity
    Service->>Rules: Load ReaderIdentityConfig
    Service->>Tokens: Mint short-lived reader token
    Tokens-->>Service: token, attrs, expiry
    Service-->>API: Preview token payload
    API-->>Owner: Token for reader preview
```

## Usage

```
import { Injectable } from '@nestjs/common';
import { ContentVisibilityService } from './content-visibility.service';

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

```
    private readonly contentVisibility: ContentVisibilityService,
  ) {}

  async restrictDocument(
    projectId: string,
    documentId: string,
    caller: { orgId: string; userId: string },
  ) {
    // The organization must come from the authenticated JWT/caller context.
    // Do not accept an orgId from route or request body parameters.
    return this.contentVisibility.setDocumentRule(
      projectId,
      documentId,
      {
        visibility: 'restricted',
        allowedAttributes: {
          department: ['engineering'],
        },
      },
      caller,
    );
  }

  async previewAsReader(
    projectId: string,
    caller: { orgId: string; userId: string },
  ) {
    const preview = await this.contentVisibility.mintPreviewToken(
      projectId,
      {
        department: 'engineering',
        role: 'member',
      },
      caller,
    );

    return {
      token: preview.token,
      readerAttributes: preview.attrs,
      expiresInSeconds: preview.expiresInSeconds,
    };
  }
}
```

## AI Coding Instructions

- Always derive organization ownership from the authenticated JWT/caller context; never trust `orgId` supplied through route params, query strings, or request bodies.

- After changing a document or category rule, preserve the cache invalidation and RAG metadata re-stamping flow so visibility changes apply to chat immediately.
- Re-stamp chunk metadata only; visibility-rule updates should not trigger unnecessary embedding regeneration.
- Keep `ReaderIdentityConfig` updates and preview-token minting scoped to projects verified as belonging to the caller’s organization.
- Treat preview tokens as short-lived, least-privilege reader credentials and return their expiry information to clients.

## Relationships

- DEPENDS_ON → `PrismaService`
- DEPENDS_ON → `DocsVisibilityService`
- DEPENDS_ON → `ReaderTokenService`
- DEPENDS_ON → `TechDocsRagService`

## Referenced By

- `ContentVisibilityController` (DEPENDS_ON)
- `ProjectModule` (MODULE_PROVIDES)