DraftDocumentService

September 4, 2026

DraftDocumentService

Kind: Service

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

Single reusable draft→publish backbone entry point: create a `state:'DRAFT'`, `version:1`
Document with a unique urlId + slug. Every draft producer (the recorder's flow→manual path,
grounded AI page assist, and the Wave-3 ticket→content loop) funnels through here so they all
land in the SAME review/publish flow (`POST documents/:id/publish`) — no forked publish paths.

`DraftDocumentService` is the shared backend entry point for creating new draft documents with `state: 'DRAFT'`, `version: 1`, and unique `urlId` and slug values. Recorder flows, grounded AI assistance, and ticket-to-content generation all use this service so every document follows the same review and `POST /documents/:id/publish` publishing path.

Methods

Method	Signature	Returns
`createDraftDocument`	`createDraftDocument(params: CreateDraftDocumentParams)`	`Promise<DraftDocumentResult>`

Dependencies

- `PrismaService`

Where it refuses work

- `DraftDocumentService` stops the work with an early return when `!audienceSlug`.
- `DraftDocumentService` stops the work with an early return when `!clash`.

Diagram

```
sequenceDiagram
  participant Producer as Draft Producer
  participant Service as DraftDocumentService
  participant DB as Document Repository
  participant Review as Review/Publish Flow

  Producer->>Service: createDraft(input)
  Service->>Service: Generate unique urlId and slug
  Service->>DB: Create Document<br/>state: DRAFT, version: 1
  DB-->>Service: Draft document
  Service-->>Producer: Draft document ID and metadata

  Producer->>Review: POST /documents/:id/publish
  Review->>DB: Validate and publish draft
  DB-->>Review: Published document
```

Usage

```
import { Injectable } from '@nestjs/common';
import { DraftDocumentService } from './services/draft-document.service';

@Injectable()
export class TicketContentService {
  constructor(
    private readonly draftDocumentService: DraftDocumentService,
  ) {}

  async createDraftFromTicket(ticket: {
    title: string;
    content: string;
    workspaceId: string;
    authorId: string;
  }) {
    const draft = await this.draftDocumentService.createDraft({
      title: ticket.title,
      content: ticket.content,
      workspaceId: ticket.workspaceId,
      authorId: ticket.authorId,
    });

    // Return the draft for the standard review/publish workflow.
    return {
      documentId: draft.id,
```

```
      state: draft.state,
      publishEndpoint: `/documents/${draft.id}/publish`,
    };
  }
}
```

AI Coding Instructions

- Route every new draft-producing workflow through `DraftDocumentService`; do not create documents directly from recorder, AI, or ticket feature code.
- Preserve the initial document invariants: `state: 'DRAFT'` and `version: 1`.
- Let the service own unique `urlId` and slug generation rather than accepting caller-generated values unless the service contract explicitly supports them.
- Do not add alternate publishing paths for generated content; published transitions must continue through `POST /documents/:id/publish`.
- When adding producer-specific metadata, keep it additive and avoid changing the shared draft/review lifecycle.

Relationships

- DEPENDS_ON → `PrismaService`

Referenced By

- `CaptureService` (DEPENDS_ON)
- `DraftDocumentModule` (MODULE_PROVIDES)
- `DraftDocumentModule` (MODULE_EXPORTS)
- `ManualsInsightsService` (DEPENDS_ON)
- `TechnicalDocsMcpController` (DEPENDS_ON)