ManualsInsightsService

September 4, 2026

ManualsInsightsService

Kind: Service

Source: `atloria-monorepo/apps/api/src/manuals-insights/manuals-insights.service.ts`

`ManualsInsightsService` provides backend analytics and automation for the manuals knowledge base. It aggregates insight and deflection data, identifies ticket-topic clusters, generates draft documentation from those clusters, and reports freshness status for tracked manual content.

Methods

Method	Signature	Returns	Description
`insights`	`insights(projectId: string, organizationId: string)`	`unknown`	Authed, org-scoped insights for a project: weakPages — published pages with 👎 feedback and/or search misses that match them.
`deflection`	`deflection(projectId: string, organizationId: string, days: unknown)`	`unknown`	Deflection & CSAT cockpit (WS-D) for one project.
`ticketTopics`	`ticketTopics(projectId: string, organizationId: string, days: unknown)`	`Promise<TicketTopicProposal[]>`	Authed, org-scoped list of content-loop proposals for a project: clusters of resolved QUESTION tickets that recur enough to deserve a

Method	Signature	Returns	Description
			page and aren't already…
`generateDraftFromCluster`	`generateDraftFromCluster(projectId: string, organizationId: string, userId: string, topicKey: string)`	`Promise<{ documentId: string; slug: string; state: string }>`	Turn one proposed cluster (by topicKey) into a GROUNDED DRAFT document: - build a neutral, PII-free question from the cluster's shared terms, - answer it via…
`freshness`	`freshness(projectId: string, slug: string)`	`Promise<{ stale: boolean; tracked: boolean; since?: string; staleCount?: number; commit?: string }>`	PUBLIC reader-facing freshness for one manual page slug.

Dependencies

- `PrismaService`
- `TechDocsRagService`
- `DraftDocumentService`

Where it refuses work

- `ManualsInsightsService` stops the work with `ForbiddenException` when `!project` — “Project not found in your organization.”, in 4 places.
- `ManualsInsightsService` stops the work with `BadRequestException` when `!cluster` — “No qualifying ticket topic for that key (too few tickets, or already documented).”.
- `ManualsInsightsService` stops the work with an early return when `!entitiesJson`.
- `ManualsInsightsService` stops the work with an early return when `!Array.isArray(entities) || !entities.length`.
- `ManualsInsightsService` stops the work with an early return when `!topicTerms.length`.
- `ManualsInsightsService` stops the work with an early return when `!slug`.

When something fails

- `ManualsInsightsService` handles failure in 1 place: it discards it silently in all 1.

Diagram

```
sequenceDiagram
  participant Client
  participant Controller
  participant Service as ManualsInsightsService
  participant Tickets as Ticket Data Source
  participant Manuals as Manuals Repository

  Client->>Controller: Request insights, topics, or freshness
  Controller->>Service: Call service method

  alt Ticket topic analysis
    Service->>Tickets: Load and cluster ticket data
    Tickets-->>Service: Ticket clusters
    Service-->>Controller: TicketTopicProposal[]
  else Generate draft
    Service->>Tickets: Read selected topic cluster
    Service->>Manuals: Create draft manual document
    Manuals-->>Service: documentId, slug, state
    Service-->>Controller: Draft metadata
  else Freshness check
    Service->>Manuals: Compare tracked content with source state
    Manuals-->>Service: Tracking and commit information
    Service-->>Controller: Freshness result
  end

  Controller-->>Client: API response
```

Usage

```
import { Injectable } from '@nestjs/common';
import { ManualsInsightsService } from './manuals-insights.service';

@Injectable()
export class ManualsInsightsController {
  constructor(
    private readonly manualsInsightsService: ManualsInsightsService,
  ) {}

  async getTicketTopics() {
    return this.manualsInsightsService.ticketTopics();
  }

  async createDraftFromTopicCluster() {
    const draft =
      await this.manualsInsightsService.generateDraftFromCluster();

    return {
      documentId: draft.documentId,
      slug: draft.slug,
      state: draft.state,
    };
  }
```

```
  async getFreshness() {
    const freshness = await this.manualsInsightsService.freshness();

    return {
      stale: freshness.stale,
      tracked: freshness.tracked,
      staleCount: freshness.staleCount ?? 0,
      lastTrackedCommit: freshness.commit,
    };
  }
}
```

AI Coding Instructions

- Inject `ManualsInsightsService` through NestJS dependency injection; do not instantiate it directly.
- Use `ticketTopics()` before generating documentation drafts so draft creation is based on validated ticket clusters.
- Treat `generateDraftFromCluster()` output as draft metadata; consumers should handle the returned `state` before publishing or exposing content.
- Handle optional `freshness()` fields such as `since`, `staleCount`, and `commit` defensively when the manual is not tracked.
- Keep insight and deflection endpoints read-oriented, and avoid adding write side effects outside explicit draft-generation workflows.

Relationships

- DEPENDS_ON → `PrismaService`
- DEPENDS_ON → `TechDocsRagService`
- DEPENDS_ON → `DraftDocumentService`

Referenced By

- `ManualsInsightsController` (DEPENDS_ON)
- `ManualsInsightsModule` (MODULE_PROVIDES)