TechDocsSourcesService

September 4, 2026

TechDocsSourcesService

Kind: Service

Source: `atloria-monorepo/apps/api/src/technical-docs/techdocs-sources.service.ts`

`TechDocsSourcesService` manages technical documentation sources for the backend. It imports content from URLs, PDFs, Zendesk, and Intercom; pulls support tickets from external providers; ingests them into the documentation pipeline; and supports listing and removing configured sources.

Methods

Method	Signature	Returns	Description
`importUrl`	`importUrl(projectId: string, userId: string, rawUrl: string)`	`unknown`	Import a public web page (HTML or markdown/plain) as an external source.
`importPdf`	`importPdf(projectId: string, userId: string, buffer: Buffer, filename: string)`	`unknown`	Import an uploaded PDF as an external source (pdf-parse text extraction).
`importZendesk`	`importZendesk(projectId: string, userId: string, dto: { subdomain?: string; email?: string; apiToken?: string; max?: number })`	`unknown`	Ticket ingestion (4.3): pull recently-solved Zendesk tickets into the corpus.
`importIntercom`	`importIntercom(projectId: string, userId: string,`	`unknown`	Ticket ingestion (4.3): Intercom conversations →

Method	Signature	Returns	Description
	`dto: { accessToken?: string; max?: number })`		corpus, same digest shape.
`pullZendesk`	`pullZendesk(config: { subdomain?: string; email?: string; apiToken?: string; max?: number }, since: string, max: unknown)`	`Promise<TicketItem[]>`	Reusable Zendesk pull (used by both the one-shot import and the managed-connector cron).
`pullIntercom`	`pullIntercom(config: { accessToken?: string; max?: number }, since: string, max: unknown)`	`Promise<TicketItem[]>`	
`ingestTickets`	`ingestTickets(projectId: string, organizationId: string, userId: string, provider: 'zendesk' | 'intercom', tickets: TicketItem[], connectorId: string)`	`unknown`	
`list`	`list(projectId: string, userId: string)`	`unknown`	
`remove`	`remove(projectId: string, userId: string, documentId: string)`	`unknown`	

Dependencies

- `PrismaService`
- `TechnicalDocsQueue`
- `TechDocsRagService`

Where it refuses work

- `TechDocsSourcesService` stops the work with `BadRequestException` when `!res.ok`, in 3 places.
- `TechDocsSourcesService` stops the work with `NotFoundException` when `!project` — “Project not found”.
- `TechDocsSourcesService` stops the work with `BadRequestException` when `!content.trim()` — “No text could be extracted from this source.”.

- `TechDocsSourcesService` stops the work with `BadRequestException` when `Buffer.byteLength(raw, 'utf8') > MAX_FETCH_BYTES` — “Source too large (max 2MB).”.
- `TechDocsSourcesService` stops the work with `BadRequestException` when `!buffer?.length` — “Empty upload.”.
- `TechDocsSourcesService` stops the work with `BadRequestException` when `buffer.length > MAX_PDF_BYTES` — “PDF too large (max 25MB).”.

When something fails

- `TechDocsSourcesService` handles failure in 4 places: it lets it reach the caller in all 4.

Diagram

```
sequenceDiagram
  participant Client
  participant Service as TechDocsSourcesService
  participant Zendesk
  participant Intercom
  participant Ingestion as Documentation Ingestion

  Client->>Service: importZendesk() / importIntercom()
  Service->>Zendesk: pullZendesk()
  Zendesk-->>Service: TicketItem[]
  Service->>Intercom: pullIntercom()
  Intercom-->>Service: TicketItem[]
  Service->>Ingestion: ingestTickets(tickets)
  Ingestion-->>Service: Indexed documentation
  Service-->>Client: Import result

  Client->>Service: list()
  Service-->>Client: Configured sources

  Client->>Service: remove(source)
  Service-->>Client: Source removed
```

Usage

```
import { Injectable } from '@nestjs/common';
import { TechDocsSourcesService } from './techdocs-sources.service';

@Injectable()
export class SupportDocsSyncService {
  constructor(
    private readonly techDocsSourcesService: TechDocsSourcesService,
  ) {}
```

```
  async syncZendeskTickets() {
    // Retrieve normalized tickets from the configured Zendesk integration.
    const tickets = await this.techDocsSourcesService.pullZendesk();

    // Send tickets through the technical documentation ingestion flow.
    await this.techDocsSourcesService.ingestTickets(tickets);

    return tickets;
  }

  async syncIntercomTickets() {
    const tickets = await this.techDocsSourcesService.pullIntercom();

    await this.techDocsSourcesService.ingestTickets(tickets);

    return tickets;
  }
}
```

AI Coding Instructions

- Keep external-provider logic behind `pullZendesk()` and `pullIntercom()` so downstream ingestion works with normalized `TicketItem` values.
- Route imported ticket content through `ingestTickets()` rather than bypassing the service with direct persistence or indexing calls.
- Preserve separate import paths for URL, PDF, Zendesk, and Intercom sources; each source type may require different validation and parsing behavior.
- Handle provider failures, missing credentials, and empty ticket results before triggering ingestion.
- Use `list()` and `remove()` for source lifecycle management instead of modifying configured source records directly.

Relationships

- DEPENDS_ON → `PrismaService`
- DEPENDS_ON → `TechnicalDocsQueue`
- DEPENDS_ON → `TechDocsRagService`

Referenced By

- `TechDocsConnectorsService` (DEPENDS_ON)
- `TechnicalDocsController` (DEPENDS_ON)

- `TechnicalDocsModule` (MODULE_PROVIDES)