TechnicalDocsMaterializerService

September 4, 2026

TechnicalDocsMaterializerService

Kind: Service

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

`TechnicalDocsMaterializerService` builds and exposes generated technical documentation artifacts for a project. It materializes full or delta documentation updates, produces RAG and MCP corpora, generates static search indexes, and provides internal and public `llms.txt` / `llms-full.txt` content for AI consumers.

Methods

Method	Signature	Returns	Description
`materialize`	`materialize(projectId: string, organizationId: string, createdById: string)`	`Promise<MaterializeResult>`	Build + activate a browsable technical doc set from the project's TechnicalSnapshot.
`materializeDelta`	`materializeDelta(projectId: string, organizationId: string, createdById: string)`	`Promise<(MaterializeResult & { mode: 'delta'; changed: number; added: number; removed: number })`	null>`
`buildRagCorpus`	`buildRagCorpus(projectId: string)`	`Promise<{`	

```
projectName: string;
organizationId: string;
pages: Array<LlmsPage & { docType: 'technical' | 'user_manual' | 'external'; entityId?: string | nu
```

} | null> `| Assemble the UNIFIED RAG corpus for a project: technical pages AND user-manual pages, each tagged with its docType — so one retrieval layer serves techdocs a… |`

| getStaticSearchIndex | getStaticSearchIndex(projectId: string, slugWithId: string, hiddenSlugs: Set | null) | Promise<StaticSearchIndex | null> | Build the PREBUILT STATIC search index a published site fetches once and searches client-side when the live search API is

unavailable (F4). | | buildMcpCorpus | buildMcpCorpus(projectId: string, hiddenSlugs: Set | null) | Promise<DocsCorpus | null> | Assemble the DocsCorpus an MCP server serves for a project: the published pages plus the entity/relationship graph from the snapshot. |
getLlmsTxt	getLlmsTxt(projectId: string, hiddenSlugs: Set	null)	Promise<string	null>	The curated llms.txt index for a project's technical docs (null if none published).
getLlmsFullTxt	getLlmsFullTxt(projectId: string, hiddenSlugs: Set	null)	Promise<string	null>	The full concatenated corpus for single-ingest by an agent.
getPublicLlmsTxt	getPublicLlmsTxt(projectId: string, hiddenSlugs: Set | null)	Promise<string	null>	llms.txt index for a published site (technical + user manuals; excludes external sources).		getPublicLlmsFullTxt	getPublicLlmsFullTxt(projectId: string, hiddenSlugs: Set | null)	Promise<string	null>	Full concatenated corpus of a published site for single-ingest by an agent.
buildPublicMcpCorpus	buildPublicMcpCorpus(projectId: string, hiddenSlugs: Set | null)	Promise<DocsCorpus	null>	The DocsCorpus for a PUBLIC slug-fronted MCP server: published pages across technical AND user manuals (external excluded, SF-7) plus the entity/relationship…		getPublicPageMarkdown	getPublicPageMarkdown(projectId: string, slug: string, hiddenSlugs: Set | null)	Promise<string	null>	Raw markdown twin of one published page (technical or user manual), or null if not found.
getPageMarkdown	getPageMarkdown(projectId: string, slug: string, hiddenSlugs: Set	null)	Promise<string	null>	The clean markdown twin of a single page (the .md` URL), or null when not found.

Dependencies

- `PrismaService`
- `DocVersionService`
- `DocVersionExportService` *(optional)*
- `SnippetsService` *(optional)*
- `ChangelogDraftsService` *(optional)*
- `TechnicalDocsEnrichmentService` *(optional)*

Where it refuses work

- `TechnicalDocsMaterializerService` stops the work with `NotFoundException` when `!snapshot` — “No technical snapshot found for this project. Run the technical-docs parser first.”.
- `TechnicalDocsMaterializerService` stops the work with an early return when `!d`, in 5 places.
- `TechnicalDocsMaterializerService` stops the work with an early return when `!version`, in 3 places.

- `TechnicalDocsMaterializerService` stops the work with an early return when `!pages.length`, in 3 places.
- `TechnicalDocsMaterializerService` stops the work with an early return when `!snapshot`, in 2 places.
- `TechnicalDocsMaterializerService` stops the work with an early return when `!agent`, in 2 places.

When something fails

- `TechnicalDocsMaterializerService` handles failure in 2 places: it logs it and continues in all 2.

Diagram

```
sequenceDiagram
  participant Caller
  participant Service as TechnicalDocsMaterializerService
  participant Sources as Documentation Sources
  participant Store as Generated Artifacts

  Caller->>Service: materialize() / materializeDelta()
  Service->>Sources: Collect entities, manuals, and external docs
  Service->>Service: Build documentation pages and metadata
  Service->>Store: Persist materialized artifacts
  Store-->>Service: MaterializeResult
  Service-->>Caller: Materialization status

  Caller->>Service: buildRagCorpus() / buildMcpCorpus()
  Service->>Store: Read generated documentation
  Service-->>Caller: RAG corpus or MCP corpus

  Caller->>Service: getLlmsTxt() / getStaticSearchIndex()
  Service->>Store: Read published artifacts
  Service-->>Caller: AI discovery text or search index
```

Usage

```
import { Injectable } from '@nestjs/common';
import { TechnicalDocsMaterializerService } from './technical-docs-materializer.service';

@Injectable()
export class DocsPublicationJob {
  constructor(
    private readonly materializer: TechnicalDocsMaterializerService,
  ) {}
```

```
  async publishDocs() {
    const result = await this.materializer.materialize();

    const searchIndex = await this.materializer.getStaticSearchIndex();
    const ragCorpus = await this.materializer.buildRagCorpus();
    const publicLlmsTxt = await this.materializer.getPublicLlmsTxt();

    return {
      materialized: result,
      searchable: searchIndex !== null,
      ragPageCount: ragCorpus?.pages.length ?? 0,
      publicLlmsTxt,
    };
  }

  async publishChangesOnly() {
    const delta = await this.materializer.materializeDelta();

    if (!delta) {
      return { updated: false, reason: 'No documentation changes detected.' };
    }

    return {
      updated: true,
      changed: delta.changed,
      added: delta.added,
      removed: delta.removed,
    };
  }
}
```

AI Coding Instructions

- Prefer `materializeDelta()` for scheduled or incremental updates; handle its `null` result as a valid “no changes” outcome.
- Use `materialize()` when a complete rebuild is required, such as initial publication or recovery from missing generated artifacts.
- Treat corpus, search-index, and `llms` retrieval methods as nullable; callers must gracefully handle unavailable documentation.
- Keep public and internal outputs separate: use `getPublicLlmsTxt()`, `getPublicLlmsFullTxt()`, and `buildPublicMcpCorpus()` only for externally safe content.
- Preserve page metadata when consuming `buildRagCorpus()`, especially `docType` and optional `entityId`, for filtering and retrieval attribution.

Relationships

- DEPENDS_ON → `PrismaService`

- DEPENDS_ON → `DocVersionService`
- DEPENDS_ON → `DocVersionExportService`
- DEPENDS_ON → `SnippetsService`
- DEPENDS_ON → `ChangelogDraftsService`
- DEPENDS_ON → `TechnicalDocsEnrichmentService`

Referenced By

- `PublicProjectController` (DEPENDS_ON)
- `TechDocsRagService` (DEPENDS_ON)
- `TechDocsExportService` (DEPENDS_ON)
- `TechDocsNotionExportService` (DEPENDS_ON)
- `TechnicalDocsGenerationService` (DEPENDS_ON)
- `TechnicalDocsMcpController` (DEPENDS_ON)
- `TechnicalDocsQueue` (DEPENDS_ON)
- `TechnicalDocsModule` (MODULE_PROVIDES)
- `TechnicalDocsModule` (MODULE_EXPORTS)