TranslationsService

September 4, 2026

TranslationsService

Kind: Service

Source: `atloria-monorepo/apps/api/src/l10n/translations.service.ts`

`TranslationsService` manages localized page content, translation status, and language-pair operations for the API. It also provides localization settings such as glossary terms and automatic retranslation behavior, acting as the backend coordination layer for translation workflows.

Methods

Method	Signature	Returns	Description
`hashContent`	`hashContent(content: string	null	undefined)`
`overview`	`overview(projectId: string)`	`Promise<L10nOverview>`	Per-language coverage/staleness rollup across the served doc set.
`pagesForLanguage`	`pagesForLanguage(projectId: string, language: string)`	`Promise<TranslationStatusRow[]>`	Per-page rows for one target language (the cockpit language drill-down / bulk source).
`getPair`	`getPair(documentId: string, language: string)`	`unknown`	Source document + its variant (if any) for the split-view editor.
`upsert`	`upsert(documentId: string, language: string, input: {		

```
  content: string;
  title?: string | null;
  status?: string;
  machineTranslated?: boolean;
```

```
  translatedBy?: string | null;
})` | `unknown` | Create/update a variant. |
```

`setStatus`	`setStatus(documentId: string, language: string, status: 'draft'	'published')`	`unknown`	Publish / unpublish an existing variant (draft ↔ published).
`remove`	`remove(documentId: string, language: string)`	`Promise<void>`	
`getSettings`	`getSettings(projectId: string)`	`Promise<{ glossary: Glossary; autoRetranslate: boolean }>`	
`updateSettings`	`updateSettings(projectId: string, input: { glossary?: Glossary; autoRetranslate?: boolean })`	`Promise<{ glossary: Glossary; autoRetranslate: boolean }>`	
`glossaryFor`	`glossaryFor(projectId: string, language: string)`	`Promise<GlossaryTerm[]>`	Glossary terms for one target language (prompt input).
`staleMachineVariants`	`staleMachineVariants(projectId: string)`	`Promise<Array<{ documentId: string; language: string }>>`	Machine-translated PUBLISHED variants whose source moved under them (`sourceContentHash != document.contentHash`) — the auto-retranslate work list.

Dependencies

- `PrismaService`
- `OutboxService` *(optional)*

Where it refuses work

- `TranslationsService` stops the work with `NotFoundException` when `!document` — “Document not found”, in 2 places.
- `TranslationsService` stops the work with `BadRequestException` when `!VARIANT_STATUSES.has(status)` , in 2 places.
- `TranslationsService` stops the work with `NotFoundException` when `!existing` — “Translation not found”, in 2 places.
- `TranslationsService` stops the work with `BadRequestException` when `typeof input.content !== 'string' || !input.content.trim()` — “content is required”.
- `TranslationsService` stops the work with `BadRequestException` when `(version?.language || 'en') === lang` .
- `TranslationsService` stops the work with `NotFoundException` when `!project` — “Project not found”.

Diagram

```
sequenceDiagram
  participant Client
  participant Controller
  participant Service as TranslationsService
  participant Store as Translation Storage
```

```
  participant Settings as L10n Settings

  Client->>Controller: Request translation overview
  Controller->>Service: overview()
  Service->>Store: Load translation statuses
  Store-->>Service: TranslationStatusRow[]
  Service-->>Controller: L10nOverview
  Controller-->>Client: Overview response

  Client->>Controller: Update translation content/status
  Controller->>Service: upsert() / setStatus()
  Service->>Store: Persist translation changes
  Store-->>Service: Updated record
  Service-->>Controller: Result

  Client->>Controller: Update localization settings
  Controller->>Service: updateSettings()
  Service->>Settings: Save glossary and auto-retranslate flag
  Settings-->>Service: Updated settings
  Service-->>Controller: Settings response
```

Usage

```
import { TranslationsService } from './translations.service';

export class TranslationAdminController {
  constructor(
    private readonly translationsService: TranslationsService,
  ) {}

  async getOverview() {
    return this.translationsService.overview();
  }

  async getLanguagePages(language: string) {
    return this.translationsService.pagesForLanguage(language);
  }

  async saveSettings() {
    return this.translationsService.updateSettings({
      glossary: {
        terms: [
          { source: 'Atloria', target: 'Atloria' },
          { source: 'workspace', target: 'espace de travail' },
        ],
      },
      autoRetranslate: true,
    });
  }

  async getGlossary(language: string) {
    return this.translationsService.glossaryFor(language);
  }
}
```

AI Coding Instructions

- Use `overview()` and `pagesForLanguage()` for read-oriented translation dashboard queries rather than reconstructing status data in controllers.
- Keep translation writes centralized through `upsert()`, `setStatus()`, and `remove()` so status and content lifecycle rules remain consistent.
- Preserve the content hashing behavior provided by `hashContent()` when comparing source content or determining whether a translation is stale.
- Read and update glossary and automatic retranslation configuration through `getSettings()` and `updateSettings()`; avoid bypassing the service with direct settings persistence.
- Treat `getPair()` and `upsert()` return values according to their concrete implementation types, since their exposed metadata currently identifies them as `unknown`.

Relationships

- DEPENDS_ON → `PrismaService`
- DEPENDS_ON → `OutboxService`

Referenced By

- `L10nAutomationService` (DEPENDS_ON)
- `L10nController` (DEPENDS_ON)
- `L10nModule` (MODULE_PROVIDES)
- `L10nModule` (MODULE_EXPORTS)