# DocAutomationService

September 4, 2026

## DocAutomationService

**Kind:** Service

**Source:** `atloria-monorepo/apps/api/src/documentation/services/doc-automation.service.ts`

`DocAutomationService` is a NestJS backend service responsible for managing documentation automation configuration and scenarios. It coordinates configuration lifecycle operations, exposes screen baseline metadata, and triggers documentation-generation jobs from webhook events.

### Methods

| Method | Signature | Returns | Description |
|---|---|---|---|
| `onModuleInit` | `onModuleInit()` | `Promise<void>` | On startup: clean up orphaned temp directories from previous crashed runs |
| `onModuleDestroy` | `onModuleDestroy()` | `Promise<void>` | On shutdown: cancel all running pipelines and clean up their work directories |
| `createConfig` | `createConfig(dto: CreateDocAutomationConfigDto, user: JwtPayload)` | `unknown` | Create a new documentation automation config for a project |
| `getConfig` | `getConfig(projectId: string, user: JwtPayload)` | `unknown` | Get config by project ID |

| Method | Signature | Returns | Description |
| --- | --- | --- | --- |
| `listScenarios` | `listScenarios(projectId: string, user: JwtPayload)` | `unknown` | Scenario Library: candidates mined by runs (flagship first, then newest). |
| `addScenario` | `addScenario(projectId: string, dto: CreateScenarioDto, user: JwtPayload)` | `unknown` | Scenario Library: owner-authored scenario (source=custom). |
| `getScreenBaseline` | `getScreenBaseline(projectId: string, user: JwtPayload)` | `Promise<{ screens: unknown[]; generatorFingerprint: string | null }>` | |
| `updateConfig` | `updateConfig(projectId: string, dto: UpdateDocAutomationConfigDto, user: JwtPayload)` | `unknown` | Update config |
| `deleteConfig` | `deleteConfig(projectId: string, user: JwtPayload)` | `unknown` | Delete config |
| `triggerJobFromWebhook` | `triggerJobFromWebhook(opts: { | | |

```
projectId: string;
organizationId: string;
branch?: string;
commitHash?: string;
provider: string;          // 'github' | 'gitlab' | 'azure-devops'
event: string;             // 'push' | 'tag_create' | 'release'
```

}) | Promise<{ jobId: string; status: string }> | `Webhook-driven trigger.` |
| startJob | startJob(dto: StartDocJobDto, user: JwtPayload) | unknown | `Start a new documentation job` | | getJob | getJob(jobId: string, user: JwtPayload) | unknown | `Get job by ID` | | listJobs | listJobs(filters: JobFiltersDto, user: JwtPayload) | unknown | `List jobs with filters` |
| scheduledReconcileSweep | scheduledReconcileSweep() | Promise | `Enqueue an atlorix job.` | | reportExternalProgress | reportExternalProgress(jobId: string, body: {
stage: string;
progress: number;
stageProgress?: number;
currentItem?: string;

eventType?: string;
}, user: JwtPayload) | Promise<{ ok: true }> | Report progress for an externally-driven job (atlorix-runner consumer). | | reportExternalFailure | reportExternalFailure(jobId: string, body: { error: string; errorStage?: string; errorStack?: string }, user: JwtPayload) | Promise<{ ok: true }> | Terminal failure reported by an external generator (atlorix-runner) after it exhausts its retries. |
| runUploadOnExistingJob | runUploadOnExistingJob(jobId: string, externalOutputPath: string, user: JwtPayload, options: {
updateAssetsOnly?: boolean;
onlySlugs?: string[];
// S3.1 delta: forwarded to runExternalOutputUpload so a change-aware run patches the
// ACTIVE version in place (partial-merge) instead of a full replace (page loss).
deltaMerge?: boolean;
removedSlugs?: string[];
/** D1: patch the ACTIVE version's video directives in place from a fresh videos/ dir. */
videoOnly?: boolean;
}) | Promise<{ ok: true }> | Run the stage-13 upload against an EXISTING DocumentationJob. |
| cancelJob | cancelJob(jobId: string, user: JwtPayload) | unknown | Cancel a running or pending job | | updateJobProgress | updateJobProgress(jobId: string, update: JobProgressUpdate) | unknown | Update job progress (optimized - single DB call) |
| completeJob | completeJob(jobId: string, data: JobCompletionData) | unknown | Mark job as completed | | failJob | failJob(jobId: string, data: JobFailureData) | unknown | Mark job as failed |
| deleteJobDocuments | deleteJobDocuments(jobId: string, user: JwtPayload) | unknown | Delete all documents and categories created by a specific job. |
| triggerBackgroundScreenshots | triggerBackgroundScreenshots(jobId: string, user: JwtPayload) | unknown` | Trigger background screenshots for an existing completed job. |

## Dependencies

- `PrismaService`
- `DocAutomationGateway`
- `UrlService`
- `AssetsService`
- `DocumentIndexingService`
- `AIUsageService`

- `Queue` *(optional)*
- `AzureClaudeProvider` *(optional)*
- `DocVersionService` *(optional)*
- `VersionCarryForwardService` *(optional)*
- `NotificationService` *(optional)*
- `EmailService` *(optional)*
- `OutboxService` *(optional)*
- `ChangeRequestService` *(optional)*
- `DocVersionExportService` *(optional)*
- `ChangelogDraftsService` *(optional)*
- `L10nAutomationService` *(optional)*
- `JobReconcileService` *(optional)*
- `PlanService` *(optional)*

## Where it refuses work

- `DocAutomationService` stops the work with `NotFoundException` when `!job` — “Documentation job not found”, in 9 places.
- `DocAutomationService` stops the work with `ForbiddenException` when `job.organizationId !== user.organizationId && !this.isServiceAccount(user)` — “You do not have access to this job”, in 4 places.
- `DocAutomationService` stops the work with `NotFoundException` when `!config` — “Documentation automation config not found for this project”, in 3 places.
- `DocAutomationService` stops the work with `ForbiddenException` when `job.organizationId !== user.organizationId` — “You do not have access to this job”, in 3 places.
- `DocAutomationService` stops the work with `BadRequestException` when `!this.atlorixQueue` — “atlorix-jobs queue is not configured (Redis unavailable or queue not registered)”, in 2 places.
- `DocAutomationService` stops the work with `NotFoundException` when `!project` — “Project not found”.

## When something fails

- `DocAutomationService` handles failure in 48 places: it logs it and continues in 32, discards it silently in 11, lets it reach the caller in 3, and turns it into a return value

in 2. A failure discarded silently leaves no trace for whoever debugs this later.

## Diagram

```
sequenceDiagram
  participant Client
  participant Controller
  participant Service as DocAutomationService
  participant ConfigStore
  participant JobRunner

  Client->>Controller: Create/update documentation config
  Controller->>Service: createConfig() / updateConfig()
  Service->>ConfigStore: Persist configuration
  ConfigStore-->>Service: Saved configuration
  Service-->>Controller: Configuration response

  Client->>Controller: Request screen baseline
  Controller->>Service: getScreenBaseline()
  Service->>ConfigStore: Load screen definitions and fingerprint
  ConfigStore-->>Service: Screens + generator fingerprint
  Service-->>Controller: Baseline response

  Client->>Controller: Webhook event
  Controller->>Service: triggerJobFromWebhook()
  Service->>JobRunner: Create documentation job
  JobRunner-->>Service: jobId + status
  Service-->>Controller: Job response
```

## Usage

```
import { Injectable } from '@nestjs/common';
import { DocAutomationService } from './documentation/services/doc-automation.service';

@Injectable()
export class DocumentationWebhookHandler {
  constructor(
    private readonly docAutomationService: DocAutomationService,
  ) {}

  async handleWebhook(payload: unknown) {
    const { jobId, status } =
      await this.docAutomationService.triggerJobFromWebhook();

    return {
      accepted: status === 'queued',
      jobId,
      status,
```

```
    };
  }

  async getBaseline() {
    const { screens, generatorFingerprint } =
      await this.docAutomationService.getScreenBaseline();

    return {
      screenCount: screens.length,
      generatorFingerprint,
      screens,
    };
  }
}
```

## AI Coding Instructions

- Use the service through NestJS dependency injection; do not instantiate `DocAutomationService` directly.
- Preserve lifecycle behavior in `onModuleInit()` and `onModuleDestroy()` when adding resources such as clients, workers, or subscriptions.
- Keep configuration mutations within `createConfig()`, `updateConfig()`, and `deleteConfig()` so configuration handling remains centralized.
- Treat `generatorFingerprint` as nullable when consuming `getScreenBaseline()` results.
- Route webhook-triggered generation through `triggerJobFromWebhook()` to ensure jobs receive the expected creation and status handling.

## Relationships

- DEPENDS_ON → `PrismaService`
- DEPENDS_ON → `DocAutomationGateway`
- DEPENDS_ON → `UrlService`
- DEPENDS_ON → `AssetsService`
- DEPENDS_ON → `DocumentIndexingService`
- DEPENDS_ON → `AIUsageService`
- DEPENDS_ON → `queue`
- DEPENDS_ON → `AzureClaudeProvider`
- DEPENDS_ON → `DocVersionService`
- DEPENDS_ON → `VersionCarryForwardService`
- DEPENDS_ON → `NotificationService`

- DEPENDS_ON → `EmailService`
- DEPENDS_ON → `OutboxService`
- DEPENDS_ON → `ChangeRequestService`
- DEPENDS_ON → `DocVersionExportService`
- DEPENDS_ON → `ChangelogDraftsService`
- DEPENDS_ON → `L10nAutomationService`
- DEPENDS_ON → `JobReconcileService`
- DEPENDS_ON → `PlanService`

## Referenced By

- `DocAutomationController` (DEPENDS_ON)
- `DocumentationModule` (MODULE_PROVIDES)
- `DocumentationModule` (MODULE_EXPORTS)
- `FreshnessService` (DEPENDS_ON)
- `WebhookService` (DEPENDS_ON)