ScheduledPublishService

September 4, 2026

ScheduledPublishService

Kind: Service

Source: `atloria-monorepo/apps/api/src/doc-version/scheduled-publish.service.ts`

D5 — Scheduled / timed publish (the cron half).

Every minute, ONE api replica (Redis SET NX PX single-fire lock, mirroring the commit-lock pattern in docs-repo/commit-lock.ts) sweeps:

1. DocVersions whose scheduledActivateAt is due → go live via the ONLY safe path, DocVersionService.activate() (approval gate, scope-safe demotion, doc-state restore, index eviction, RAG re-index). NEVER a raw status write — a prior production incident ("no active version") came from hand-rolled status writes. On success the schedule fields are cleared and 'version.activated_scheduled' is audited. If activate() rejects (e.g. the approval gate), the schedule is KEPT so it fires once approval lands, and 'version.schedule_blocked' is audited at most once per version per day (Redis-deduped). Unexpected errors: log, keep the schedule, continue.
2. Per-document Document.scheduledPublishAt (state DRAFT/REVIEW) → direct state=PUBLISHED + isPublic + publishedAt. This lane has no version machinery, so a direct update IS the correct publish path here.

Lock semantics: the tick lock ('scheduled-publish:tick', 55s PX < the 60s cron period) is deliberately NOT released — it expires on its own, so exactly one replica works each minute. Fail-closed: if Redis is unreachable the tick is skipped (never double-fired); the next minute retries.

`ScheduledPublishService` runs the timed publishing workflow once per minute. It uses a Redis single-fire lock so only one API replica processes due version activations and direct document publishes, while safely retrying scheduled work on later ticks when Redis, approvals, or activation failures block processing.

Methods

Method	Signature	Returns	Description
`tick`	`tick()`	`Promise<void>`	
`processDueVersions`	`processDueVersions()`	`Promise<void>`	Due DocVersions → activate() via the safe path; per-version error isolation.
`processDueDocuments`	`processDueDocuments()`	`Promise<void>`	Per-document lane: Document.scheduledPublishAt due + state DRAFT/REVIEW → PUBLISHED.
`onModuleDestroy`	`onModuleDestroy()`	`void`	

Dependencies

- `ConfigService`
- `PrismaService`
- `DocVersionService`
- `AuditService` *(optional)*

Where it refuses work

- `ScheduledPublishService` stops the work with an early return when `process.env.SCHEDULED_PUBLISH_DISABLED === 'true'`.
- `ScheduledPublishService` stops the work with an early return when `!(await this.acquireTickLock())`.
- `ScheduledPublishService` stops the work with an early return when `due.length === 0`.
- `ScheduledPublishService` stops the work with an early return when `!firstToday`.

When something fails

- `ScheduledPublishService` handles failure in 5 places: it logs it and continues in 4, and turns it into a return value in 1.

Diagram

```
sequenceDiagram
    participant Cron as NestJS Cron Tick
```

```
    participant Service as ScheduledPublishService
    participant Redis as Redis
    participant Versions as DocVersionService
    participant DB as Database
    participant Audit as Audit Log
    participant RAG as Search/RAG Index

    Cron->>Service: tick()
    Service->>Redis: SET scheduled-publish:tick NX PX 55000

    alt Lock not acquired or Redis unavailable
        Redis-->>Service: false / error
        Service-->>Cron: Skip tick (fail closed)
    else Lock acquired
        Redis-->>Service: OK

        Service->>DB: Find due DocVersions
        loop Each due version
            Service->>Versions: activate(version)
            alt Activation succeeds
                Versions->>DB: Activate version and demote prior active version
                Versions->>RAG: Evict and re-index content
                Service->>DB: Clear scheduledActivateAt fields
                Service->>Audit: version.activated_scheduled
            else Approval gate or expected rejection
                Service->>Redis: Check daily blocked-event dedupe key
                Service->>Audit: version.schedule_blocked (max once/day)
                Note over Service,DB: Keep schedule for a later retry
            else Unexpected error
                Note over Service,DB: Log error and keep schedule
            end
        end

        Service->>DB: Find due DRAFT/REVIEW documents
        Service->>DB: Set state=PUBLISHED, isPublic=true, publishedAt=now
        Note over Service,Redis: Lock is not released; it expires after 55 seconds
    end
```

Usage

```
import { Injectable } from '@nestjs/common';
import { ScheduledPublishService } from './scheduled-publish.service';

@Injectable()
export class PublishAdminService {
  constructor(
    private readonly scheduledPublishService: ScheduledPublishService,
  ) {}

  /**
```

```
   * Useful for an internal admin action or integration test.
   * Normal production execution is performed by the service's cron schedule.
   */
  async processScheduledPublishes(): Promise<void> {
    await this.scheduledPublishService.tick();
  }
}
```

AI Coding Instructions

- Always activate scheduled `DocVersion` records through `DocVersionService.activate()`; never directly update version status fields.
- Preserve `scheduledActivateAt` when activation is blocked by approval requirements or unexpected errors so the next tick can retry.
- Keep the Redis tick lock fail-closed: if the lock cannot be acquired or Redis is unavailable, skip processing rather than risking duplicate publishes.
- Do not release `scheduled-publish:tick` manually; its 55-second TTL is intentional and prevents overlapping replicas.
- Direct document publishing is separate from version activation: only due `DRAFT` / `REVIEW` documents should receive the direct `PUBLISHED`, `isPublic`, and `publishedAt` update.

Relationships

- DEPENDS_ON → `configservice`
- DEPENDS_ON → `PrismaService`
- DEPENDS_ON → `DocVersionService`
- DEPENDS_ON → `AuditService`

Referenced By

- `DocVersionModule` (MODULE_PROVIDES)