PdfExportLockService

September 4, 2026

PdfExportLockService

Kind: Service

Source: `atloria-monorepo/apps/api/src/doc-version/pdf-export-lock.service.ts`

Short-lived render mutex for reader-side PDF exports (C6).

Same lazy-ioredis shape as the docs-repo CommitLockService, but FAIL-OPEN (the opposite policy): a whole-manual Chromium render is expensive but idempotent — if Redis is unreachable we let the render proceed WITHOUT the lock (worst case: a duplicate render), instead of breaking every download during a Redis blip. Committing without a lock could corrupt a git tree; rendering without one just burns a few worker-seconds.

`PdfExportLockService` provides a short-lived, Redis-backed mutex for reader-side PDF exports. It prevents duplicate whole-manual Chromium renders when Redis is available, but intentionally fails open when Redis is unavailable so PDF downloads continue even if multiple renders occur.

Methods

Method	Signature	Returns	Description
`acquire`	`acquire(key: string, ttlMs: number)`	`Promise<string	null>`	
`release`	`release(key: string, token: string)`	`Promise<void>`	Best-effort token-checked release (DEL).
`onModuleDestroy`	`onModuleDestroy()`	`void`	

Dependencies

- `ConfigService`

When something fails

- `PdfExportLockService` handles failure in 2 places: it logs it and continues in 1, and turns it into a return value in 1.

Diagram

```
sequenceDiagram
  participant Reader as Reader/API Request
  participant Export as PDF Export Handler
  participant Lock as PdfExportLockService
  participant Redis as Redis
  participant Chromium as Chromium Renderer

  Reader->>Export: Request PDF export
  Export->>Lock: acquire()

  alt Redis available and lock acquired
    Lock->>Redis: SET lockKey token NX PX ttl
    Redis-->>Lock: OK
    Lock-->>Export: token
    Export->>Chromium: Render manual PDF
    Chromium-->>Export: PDF output
    Export->>Lock: release()
    Lock->>Redis: Delete lock if token matches
  else Lock already held
    Lock->>Redis: SET lockKey token NX PX ttl
    Redis-->>Lock: Not acquired
    Lock-->>Export: null
    Export-->>Reader: Handle active export / retry policy
  else Redis unavailable
    Lock->>Redis: Attempt lock operation
    Redis-->>Lock: Connection error
    Lock-->>Export: token or fail-open result
    Export->>Chromium: Render without distributed lock
    Chromium-->>Export: PDF output
  end
```

Usage

```
import { Injectable } from '@nestjs/common';
import { PdfExportLockService } from './pdf-export-lock.service';

@Injectable()
export class PdfExportService {
  constructor(
    private readonly pdfExportLockService: PdfExportLockService,
  ) {}
```

```
  async exportManual(manualId: string): Promise<Buffer> {
    const lockToken = await this.pdfExportLockService.acquire();

    // A null token may mean another export holds the lock. Apply the
    // application's retry/conflict behavior before starting a duplicate render.
    if (!lockToken) {
      throw new Error(`A PDF export is already in progress for ${manualId}`);
    }

    try {
      return await this.renderWithChromium(manualId);
    } finally {
      await this.pdfExportLockService.release();
    }
  }

  private async renderWithChromium(manualId: string): Promise<Buffer> {
    // Load the manual and render it with Chromium.
    return Buffer.from(`PDF for ${manualId}`);
  }
}
```

AI Coding Instructions

- Always call `release()` in a `finally` block after a successful `acquire()` attempt to avoid retaining the short-lived Redis lock unnecessarily.
- Preserve the fail-open policy: Redis connection or command failures must not block PDF exports; duplicate renders are acceptable, failed downloads are not.
- Treat a normal lock-contention result separately from Redis failures so callers can apply an appropriate “export already running” or retry behavior.
- Keep lock TTLs shorter than the expected render duration only when renewal is supported; otherwise ensure the configured TTL safely covers Chromium render time.
- Let NestJS invoke `onModuleDestroy()` during shutdown so the lazily created Redis client is cleaned up correctly.

Relationships

- DEPENDS_ON → `configservice`

Referenced By

- `DocVersionModule` (MODULE_PROVIDES)

- `DocVersionExportService` (DEPENDS_ON)