ChangelogQueue

September 4, 2026

ChangelogQueue

Kind: Service

Source: `atloria-monorepo/apps/api/src/changelog/changelog-queue.service.ts`

Durable queue for changelog AI-summarize + publish fan-out jobs (mirrors DocsPrQueue). The worker runs in-process — low volume, seconds per job. The processor is registered by ChangelogDraftsService (callback, not DI) so there is no module cycle. If Redis is unavailable, `enqueue` returns false and the caller runs the work inline — behaviour, not availability, is what degrades.

jobIds use '-' separators only: BullMQ rejects ':' in custom job ids.

`ChangelogQueue` provides a durable BullMQ-backed queue for changelog AI summarization and publish fan-out jobs. `ChangelogDraftsService` registers the in-process processor through a callback to avoid a NestJS module cycle; when Redis is unavailable, `enqueue()` returns `false` so callers can run the same work inline.

Methods

Method	Signature	Returns	Description
`registerProcessor`	`registerProcessor(fn: (job: ChangelogJob) => Promise<void>)`	`void`	
`enqueue`	`enqueue(job: ChangelogJob, jobId: string)`	`Promise<boolean>`	Durable enqueue; returns false if the queue is unavailable so the caller can run inline.
`onModuleInit`	`onModuleInit()`	`unknown`	
`onModuleDestroy`	`onModuleDestroy()`	`unknown`	

Dependencies

- `ConfigService`

Where it refuses work

- `ChangelogQueue` stops the work with an early return when `!this.queue`.

When something fails

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

Diagram

```
sequenceDiagram
  participant Drafts as ChangelogDraftsService
  participant Queue as ChangelogQueue
  participant Redis as Redis / BullMQ
  participant Worker as In-process Worker

  Drafts->>Queue: registerProcessor(processJob)
  Queue->>Worker: Create worker with callback

  Drafts->>Queue: enqueue(job payload)
  Queue->>Redis: Add durable job with "-" job ID

  alt Redis available
    Redis-->>Queue: Job accepted
    Queue-->>Drafts: true
    Worker->>Redis: Reserve job
    Worker->>Worker: processJob(payload)
  else Redis unavailable
    Redis-->>Queue: Queue operation fails
    Queue-->>Drafts: false
    Drafts->>Drafts: Run work inline
  end
```

Usage

```
import { Injectable, OnModuleInit } from '@nestjs/common';
import { ChangelogQueue } from './changelog-queue.service';

@Injectable()
export class ChangelogDraftsService implements OnModuleInit {
  constructor(private readonly changelogQueue: ChangelogQueue) {}
```

```
  onModuleInit(): void {
    this.changelogQueue.registerProcessor(async (job) => {
      // Summarize the draft with AI and publish its related updates.
      await this.summarizeAndPublish(job.data.draftId);
    });
  }

  async queuePublish(draftId: string): Promise<void> {
    const queued = await this.changelogQueue.enqueue({
      draftId,
      // Use hyphens only in any custom job ID values.
      jobId: `changelog-publish-${draftId}`,
    });

    if (!queued) {
      // Redis is unavailable: preserve behavior by running inline.
      await this.summarizeAndPublish(draftId);
    }
  }

  private async summarizeAndPublish(draftId: string): Promise<void> {
    // AI summarization and publish fan-out implementation.
  }
}
```

AI Coding Instructions

- Register the processor from `ChangelogDraftsService` via `registerProcessor()` rather than injecting that service into the queue; this prevents a NestJS module dependency cycle.
- Treat `enqueue()` returning `false` as a Redis availability fallback, not a failed business operation—run the equivalent work inline.
- Use hyphens (`-`) rather than colons (`:`) in custom BullMQ job IDs, since BullMQ rejects colon-separated IDs.
- Keep queued jobs small and serializable; pass identifiers such as `draftId` and load additional state inside the processor.
- Preserve lifecycle behavior in `onModuleInit()` and `onModuleDestroy()` so the in-process worker and queue connections start and stop cleanly.

Relationships

- DEPENDS_ON → `configservice`

Referenced By

- `ChangelogDraftsService` (DEPENDS_ON)
- `ChangelogModule` (MODULE_PROVIDES)