StaleAlertService

September 4, 2026

StaleAlertService

Kind: Service

Source: `atloria-monorepo/apps/api/src/technical-docs/stale-alert.service.ts`

`StaleAlertService` handles notifications triggered when an alert transitions into a stale state. It encapsulates the stale-transition notification workflow so other backend components can invoke it without duplicating alert delivery logic.

Methods

Method	Signature	Returns
`notifyStaleTransition`	`notifyStaleTransition(projectId: string, info: StaleTransitionInfo)`	`Promise<void>`

Dependencies

- `PrismaService`
- `EmailService`
- `ConfigService`

Where it refuses work

- `StaleAlertService` stops the work with an early return when `!(await this.pastCooldown(projectId))`.
- `StaleAlertService` stops the work with an early return when `!project`.
- `StaleAlertService` stops the work with an early return when `!to.length`.

When something fails

- `StaleAlertService` handles failure in 1 place: it logs it and continues in all 1.

Diagram

```
sequenceDiagram
  participant Monitor as Alert/Monitoring Flow
  participant Service as StaleAlertService
  participant Notification as Notification Channel

  Monitor->>Service: notifyStaleTransition()
  Service->>Service: Process stale transition
  Service->>Notification: Send stale alert notification
  Notification-->>Service: Delivery result
  Service-->>Monitor: Promise<void>
```

Usage

```
import { Injectable } from '@nestjs/common';
import { StaleAlertService } from './stale-alert.service';

@Injectable()
export class AlertStateService {
  constructor(
    private readonly staleAlertService: StaleAlertService,
  ) {}

  async markAlertAsStale(): Promise<void> {
    // Update the alert state using the application's alert workflow.

    await this.staleAlertService.notifyStaleTransition();
  }
}
```

AI Coding Instructions

- Inject `StaleAlertService` through NestJS dependency injection; do not instantiate it directly.
- Call `notifyStaleTransition()` only after the application has determined that an alert entered the stale state.
- Always `await` the returned promise so notification failures follow the surrounding error-handling strategy.
- Keep stale-state detection outside this service; this service should remain focused on handling the stale-transition notification workflow.
- Register the service in the appropriate NestJS module and ensure its notification dependencies are available in that module.

Relationships

- DEPENDS_ON → `PrismaService`
- DEPENDS_ON → `EmailService`
- DEPENDS_ON → `configservice`

Referenced By

- `TechnicalDocsModule` (MODULE_PROVIDES)
- `TechnicalDocsService` (DEPENDS_ON)