ToursHealingService

September 4, 2026

ToursHealingService

Kind: Service

Source: `atloria-monorepo/apps/api/src/tours/tours-healing.service.ts`

ToursHealingService (D2 self-healing).

Live tours rot when the customer ships UI changes: selectors stop resolving and the player degrades to "element not found". This service closes the loop:

1. NIGHTLY CRON — for every live, self-heal-OPTED-IN tour experience (selfHealEnabled defaults to false), ask the screenshot-worker to replay the flow's selectors against the live app (`POST screenshot/flow/validate`).
2. WRITE-BACK — healed selectors are CAS-written into CapturedFlow.steps: the update only lands if steps are UNCHANGED since our read (a Prisma Json-equality guard), so an owner editing steps concurrently always wins.
3. FLAGGING — un-healable steps (element truly gone) are recorded on TourExperience.healing and emitted as a `tour_healing` event on the interaction_events spine.
4. SPIKES — the public events endpoint counts live `step_element_missing` reports per experience/hour; crossing the threshold fires an immediate out-of-band validation (debounced, opt-in only) instead of waiting for the nightly sweep.

The worker call mirrors CaptureService.callWorker (fire-and-forget friendly, no auth header, SCREENSHOT_WORKER_URL env).

`ToursHealingService` keeps live, opted-in tour experiences healthy when UI changes cause captured selectors to stop resolving. It runs nightly validation through the screenshot worker, CAS-writes healed selectors only when flow steps have not changed, records unhealable steps, and triggers debounced validation when missing-element events spike.

Methods

Method	Signature	Returns	Description
`nightlySweep`	`nightlySweep()`	`Promise<void>`	Validate every opted-in live tour once a night.
`noteElementMissing`	`noteElementMissing(experienceId: string)`	`Promise<void>`	Count a live step_element_missing report; when one experience crosses N misses within the hour

Method	Signature	Returns	Description
			(TOUR_MISS_SPIKE_THRESHOLD, default 10), fire an out-of-band v…
`validateById`	`validateById(projectId: string, experienceId: string, trigger: 'manual')`	`Promise<Record<string, unknown>>`	Owner-triggered validation (POST :id/validate).
`validateExperience`	`validateExperience(exp: TourExperienceRow, trigger: 'cron'	'spike'	'manual')`

Dependencies

- `PrismaService`
- `RedisService`
- `EventsService`

Where it refuses work

- `ToursHealingService` stops the work with an early return when `acquired !== 'OK'`.
- `ToursHealingService` stops the work with an early return when `experiences.length === 0`.
- `ToursHealingService` stops the work with an early return when `!client`.
- `ToursHealingService` stops the work with an early return when `count < this.spikeThreshold()`.
- `ToursHealingService` stops the work with an early return when `debounced !== 'OK'`.
- `ToursHealingService` stops the work with an early return when `!exp`.

When something fails

- `ToursHealingService` handles failure in 3 places: it logs it and continues in 1, turns it into a return value in 1, and discards it silently in 1. A failure discarded silently leaves no trace for whoever debugs this later.

Diagram

```
sequenceDiagram
  participant Cron as Nightly Cron / Event Endpoint
  participant Service as ToursHealingService
  participant DB as Prisma Database
  participant Worker as Screenshot Worker
  participant Events as interaction_events

  Cron->>Service: nightlySweep() / noteElementMissing()
  Service->>DB: Find live experiences with selfHealEnabled=true

  alt Nightly sweep
    Service->>Worker: POST screenshot/flow/validate
    Worker-->>Service: Validation result / healed selectors
  else Missing-element spike
    Service->>DB: Count step_element_missing events by hour
    Service->>Worker: POST screenshot/flow/validate (debounced)
    Worker-->>Service: Validation result / healed selectors
  end
```

```
  Service->>DB: CAS update CapturedFlow.steps<br/>(only if original JSON still matches)

  alt Steps cannot be healed
    Service->>DB: Update TourExperience.healing
    Service->>Events: Emit tour_healing event
  end
```

Usage

```
import { Injectable } from '@nestjs/common';
import { ToursHealingService } from './tours-healing.service';

@Injectable()
export class ToursMaintenanceJob {
  constructor(
    private readonly toursHealingService: ToursHealingService,
  ) {}

  // Invoke from the scheduled nightly job.
  async runNightlyHealing(): Promise<void> {
    await this.toursHealingService.nightlySweep();
  }

  // Invoke after receiving a public step_element_missing event.
  async handleMissingElement(experienceId: string): Promise<void> {
    await this.toursHealingService.noteElementMissing(experienceId);
  }

  // Manually validate a specific tour experience.
  async validateExperience(experienceId: string) {
    return this.toursHealingService.validateExperience(experienceId);
  }
}
```

AI Coding Instructions

- Only process live experiences where `selfHealEnabled` is explicitly enabled; the default behavior must remain opt-out.
- Preserve the compare-and-swap JSON equality guard when writing `CapturedFlow.steps`; never overwrite steps edited concurrently by an owner.
- Route validation through the screenshot worker using `SCREENSHOT_WORKER_URL` and the `POST screenshot/flow/validate` contract, consistent with `CaptureService.callWorker`.
- Record genuinely unhealable selectors in `TourExperience.healing` and emit a `tour_healing` event on the `interaction_events` spine.
- Keep spike-triggered validation debounced and scoped to per-experience, per-hour `step_element_missing` counts to avoid repeated worker calls.

Relationships

- DEPENDS_ON → `PrismaService`
- DEPENDS_ON → `RedisService`
- DEPENDS_ON → `EventsService`

Referenced By

- `PublicToursController` (DEPENDS_ON)
- `ToursController` (DEPENDS_ON)
- `ToursModule` (MODULE_PROVIDES)