DeskService

September 4, 2026

DeskService

Kind: Service

Source: `atloria-monorepo/apps/api/src/issues/desk.service.ts`

`DeskService` is the backend application service for managing issue desk threads, including listing threads, retrieving thread details, calculating counts, and handling workflow transitions. It also coordinates agent replies and duplicate-marking actions so API controllers can expose a consistent support-desk workflow.

Methods

Method	Signature	Returns	Description
`listThreads`	`listThreads(projectId: string, user: JwtPayload, bucket: DeskBucket, limit: unknown)`	`unknown`	List a bucket's threads (lazy snooze wake applied first)
`getCounts`	`getCounts(projectId: string, user: JwtPayload)`	`unknown`	Bucket counts for the desk sidebar (lazy snooze wake applied first)
`getThread`	`getThread(projectId: string, issueId: string, user: JwtPayload)`	`unknown`	One thread with its full timeline (comments + activities)
`transition`	`transition(projectId: string, issueId: string, dto: DeskTransitionDto, user: JwtPayload)`	`unknown`	Queue transitions — status changes route through IssuesService.update (activities + emails)
`reply`	`reply(projectId: string, issueId: string, dto: DeskReplyDto, user: JwtPayload)`	`unknown`	Agent reply.
`markDuplicate`	`markDuplicate(projectId: string, issueId: string, dto:`	`unknown`	REVERSIBLE merge (deliberately not Zendesk's

Method	Signature	Returns	Description
	`DeskDuplicateDto, user: JwtPayload)`		lossy one).

Dependencies

- `PrismaService`
- `IssuesService`

Where it refuses work

- `DeskService` stops the work with `NotFoundException` when `!issue` — “Issue not found”, in 4 places.
- `DeskService` stops the work with `NotFoundException` when `!losing` — “Issue not found”.
- `DeskService` stops the work with `NotFoundException` when `!winning`.
- `DeskService` stops the work with `BadRequestException` when `winning.id === losing.id` — “Cannot merge a ticket into itself”.
- `DeskService` stops the work with `BadRequestException` when `Number.isNaN(wakeAt.getTime()) || wakeAt.getTime() <= Date.now()` — “snoozedUntil must be a valid future timestamp”.
- `DeskService` stops the work with `NotFoundException` when `!project` — “Project not found”.

Diagram

```
sequenceDiagram
    participant Client
    participant Controller as Desk Controller
    participant Service as DeskService
    participant Store as Issue/Thread Repository

    Client->>Controller: Request desk threads or action
    Controller->>Service: listThreads(), getThread(), or getCounts()
    Service->>Store: Query threads and workflow data
    Store-->>Service: Thread records / counts
    Service-->>Controller: Desk response
    Controller-->>Client: JSON response

    Client->>Controller: transition(), reply(), or markDuplicate()
    Controller->>Service: Execute desk action
    Service->>Store: Validate and persist changes
```

```
    Store-->>Service: Updated thread
    Service-->>Controller: Updated result
    Controller-->>Client: Action response
```

Usage

```
import { Injectable } from '@nestjs/common';
import { DeskService } from './desk.service';

@Injectable()
export class DeskController {
  constructor(private readonly deskService: DeskService) {}

  async getInbox() {
    const [threads, counts] = await Promise.all([
      this.deskService.listThreads(),
      this.deskService.getCounts(),
    ]);

    return { threads, counts };
  }

  async respondToThread(threadId: string, message: string) {
    return this.deskService.reply(threadId, {
      body: message,
    });
  }

  async closeThread(threadId: string) {
    return this.deskService.transition(threadId, {
      status: 'closed',
    });
  }
}
```

AI Coding Instructions

- Keep desk workflow logic inside `DeskService`; controllers should validate request DTOs and delegate actions rather than update thread data directly.
- Use `getThread()` before performing reply, transition, or duplicate actions when the operation depends on the current thread state.
- Preserve valid workflow transitions when extending `transition()`; do not allow arbitrary status changes without authorization and state validation.
- Ensure `reply()` persists the message and updates any related thread metadata, such as last activity timestamps or assignment state.

- When using `markDuplicate()`, maintain a clear reference to the canonical thread and avoid deleting historical duplicate-thread context.

Relationships

- DEPENDS_ON → `PrismaService`
- DEPENDS_ON → `IssuesService`

Referenced By

- `DeskController` (DEPENDS_ON)
- `IssuesModule` (MODULE_PROVIDES)