PlatformAdminGuard

September 4, 2026

PlatformAdminGuard

Kind: Service

Source: `atloria-monorepo/apps/api/src/auth/guards/platform-admin.guard.ts`

Gates the cross-org /platform/* console. Runs AFTER JwtAuthGuard, so `request.user` is a validated JwtPayload.

SECURITY: re-verifies `isPlatformAdmin` against the DB on EVERY request rather than trusting the (long-lived) JWT `platformAdmin` claim. This mirrors JwtStrategy's per-request isActive check — revoking platform-admin (or deactivating the account) must take effect within one request, not at token expiry. Fail-closed: any missing user / inactive / non-admin ⇒ 403.

`PlatformAdminGuard` protects cross-organization `/platform/*` console endpoints after `JwtAuthGuard` has validated the JWT. On every request, it re-checks the authenticated user's active status and `isPlatformAdmin` role in the database, ensuring revoked administrator access takes effect immediately and failing closed with `403` for invalid users.

Methods

Method	Signature	Returns
`canActivate`	`canActivate(context: ExecutionContext)`	`Promise<boolean>`

Dependencies

- `PrismaService`

Where it refuses work

- `PlatformAdminGuard` stops the work with `ForbiddenException` when `!jwt?.sub` — “Not authenticated”.

- `PlatformAdminGuard` stops the work with `ForbiddenException` when `!user?.isActive || !user.isPlatformAdmin` — “Platform administrator access is required.”.

Diagram

```
sequenceDiagram
    participant Client
    participant JwtAuthGuard
    participant PlatformAdminGuard
    participant Database
    participant PlatformController

    Client->>JwtAuthGuard: Request to /platform/*
    JwtAuthGuard->>JwtAuthGuard: Validate JWT and attach request.user

    JwtAuthGuard->>PlatformAdminGuard: canActivate(context)
    PlatformAdminGuard->>PlatformAdminGuard: Read validated request.user
    PlatformAdminGuard->>Database: Load current user status and isPlatformAdmin

    alt User is active and platform admin
        Database-->>PlatformAdminGuard: Active platform-admin user
        PlatformAdminGuard-->>PlatformController: Allow request
        PlatformController-->>Client: Protected response
    else Missing user, inactive user, or non-admin
        Database-->>PlatformAdminGuard: Invalid or insufficient access
        PlatformAdminGuard-->>Client: 403 Forbidden
    end
```

Usage

```
import { Controller, Get, UseGuards } from '@nestjs/common';
import { JwtAuthGuard } from '@/auth/guards/jwt-auth.guard';
import { PlatformAdminGuard } from '@/auth/guards/platform-admin.guard';

@Controller('platform')
@UseGuards(JwtAuthGuard, PlatformAdminGuard)
export class PlatformController {
  @Get('organizations')
  async listOrganizations() {
    // Only active users currently marked as platform admins can reach this.
    return [];
  }
}
```

AI Coding Instructions

- Apply `PlatformAdminGuard` after `JwtAuthGuard`; it expects `request.user` to contain a validated JWT payload.
- Do not trust the JWT `platformAdmin` claim as the authorization source—always use the database-backed check performed by this guard.
- Preserve fail-closed behavior: missing users, inactive accounts, unavailable user records, and non-admin users must deny access with `403`.
- Use this guard only for cross-organization platform-console routes, typically under the `/platform/*` route prefix.
- When changing user-role or account-status persistence, ensure the guard's database lookup still reflects revocations on the very next request.

Relationships

- DEPENDS_ON → `PrismaService`

Referenced By

- `PlatformModule` (MODULE_PROVIDES)