PlanGuard

September 4, 2026

PlanGuard

Kind: Service

Source: `atloria-monorepo/apps/api/src/billing/plan.guard.ts`

Plan-tier gate (P0-a). Reads the `@RequiresPlan('PRO'|'BUSINESS')` metadata and verifies the CALLER'S org (from the JWT — never from params/body, per the org-guard lessons) is on that tier or higher via `PlanService.featureLevel`.

No metadata → pass (safe to register broadly). Failure → 402 with a typed payload (`code: 'PLAN_REQUIRED'`) so the web app can show an upgrade CTA.

`PlanGuard` is a NestJS authorization guard that enforces feature access based on the authenticated caller’s organization plan. It reads `@RequiresPlan('PRO' | 'BUSINESS')` metadata and compares it against `PlanService.featureLevel`; routes without plan metadata are allowed through. When the organization does not meet the required tier, it returns a typed `402 Payment Required` response with `code: 'PLAN_REQUIRED'` for upgrade handling in the web app.

Methods

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

Dependencies

- `Reflector`
- `PlanService`

Where it refuses work

- `PlanGuard` stops the work with `ForbiddenException` when `!user?.organizationId` — “User not authenticated”.
- `PlanGuard` stops the work with an early return when `!required`.
- `PlanGuard` stops the work with an early return when `this.planService.meetsPlan(actual, required)`.

Diagram

```
sequenceDiagram
  participant Client
  participant Guard as PlanGuard
  participant Reflector
  participant JWT as Authenticated Request
  participant PlanService
  participant Controller

  Client->>Guard: Request protected endpoint
  Guard->>Reflector: Read @RequiresPlan metadata

  alt No required plan metadata
    Guard->>Controller: Allow request
  else Plan requirement exists
    Guard->>JWT: Read caller org from JWT
    Guard->>PlanService: featureLevel(callerOrgId)
    PlanService-->>Guard: Current organization plan level

    alt Plan meets or exceeds requirement
      Guard->>Controller: Allow request
    else Plan is insufficient
      Guard-->>Client: 402 Payment Required<br/>{ code: "PLAN_REQUIRED" }
    end
  end
```

Usage

```
import { Controller, Get, UseGuards } from '@nestjs/common';
import { PlanGuard } from './plan.guard';
import { RequiresPlan } from './requires-plan.decorator';

@Controller('analytics')
@UseGuards(PlanGuard)
export class AnalyticsController {
  @Get('dashboard')
  @RequiresPlan('PRO')
  getDashboard() {
    return { data: 'Pro analytics dashboard' };
```

```
  }

  @Get('advanced-export')
  @RequiresPlan('BUSINESS')
  exportAdvancedReport() {
    return { data: 'Business-only export' };
  }

  @Get('status')
  getStatus() {
    // No @RequiresPlan metadata: PlanGuard allows this route.
    return { status: 'ok' };
  }
}
```

AI Coding Instructions

- Always derive the organization identity from the authenticated JWT/request context; never trust organization IDs from route parameters, query strings, or request bodies.
- Use `@RequiresPlan('PRO')` or `@RequiresPlan('BUSINESS')` on routes requiring paid-plan features, while registering `PlanGuard` broadly is safe for unannotated routes.
- Compare plan access through `PlanService.featureLevel` rather than hard-coding plan ordering in controllers or feature handlers.
- Preserve the typed insufficient-plan response: return `402 Payment Required` with `{ code: 'PLAN_REQUIRED' }` so clients can consistently show upgrade CTAs.
- Keep this guard focused on plan authorization; authentication and organization-membership validation should remain the responsibility of their dedicated guards.

Relationships

- DEPENDS_ON → `reflector`
- DEPENDS_ON → `PlanService`

Referenced By

- `BillingModule` (MODULE_PROVIDES)
- `BillingModule` (MODULE_EXPORTS)