OrganizationGuard

September 4, 2026

OrganizationGuard

Kind: Service

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

OrganizationGuard — validates an EXPLICIT `organizationId` present in the request (param, query,
or body) against the caller's org. Correct for org-settings-style routes that name an org directly.

⚠️ IT DOES NOT PROTECT resource-scoped routes. When no `organizationId` is present (the case for
every `:projectId` / `:id` / `:versionId` / `:documentId` route) it INTENTIONALLY no-ops and
allows the request — the resource-scoped access is expected to be enforced elsewhere. A 2026-07
audit found ~16 controllers wrongly treated this guard as tenant protection on such routes,
leaving them cross-tenant. For any route keyed by a child id use `ResourceOrgGuard` + `@RequireResourceOwnership(...)`, which resolves the resource → its owning org and 403s a
mismatch. Do NOT add new `:id`-scoped routes relying on this guard alone.

`OrganizationGuard` validates an explicitly supplied `organizationId` from request params, query, or body against the caller’s organization membership. It is intended for organization-scoped routes such as organization settings; when no explicit `organizationId` exists, it intentionally allows the request and does **not** provide resource-level tenant isolation.

Methods

Method	Signature	Returns
`canActivate`	`canActivate(context: ExecutionContext)`	`boolean`

Where it refuses work

- `OrganizationGuard` stops the work with `ForbiddenException` when `!user` — “User not authenticated”.
- `OrganizationGuard` stops the work with `ForbiddenException` when `user.organizationId !== organizationId` — “Access denied: You do not have permission to access this organization”.
- `OrganizationGuard` stops the work with an early return when `!organizationId`.

Diagram

```
sequenceDiagram
  participant Client
  participant Controller
  participant Guard as OrganizationGuard
  participant Request
  participant Auth as Authenticated Caller

  Client->>Controller: Request with organizationId
  Controller->>Guard: canActivate(context)
  Guard->>Request: Read params, query, and body
  Guard->>Auth: Read caller organization context

  alt Explicit organizationId is present
    Guard->>Guard: Compare requested org to caller org
    alt Organization matches
      Guard-->>Controller: Allow request
    else Organization mismatches
      Guard-->>Controller: Deny request (403)
    end
  else No organizationId is present
    Guard-->>Controller: Allow request (no-op)
  end
```

Usage

```
import { Controller, Get, Param, UseGuards } from '@nestjs/common';
import { OrganizationGuard } from '@/auth/guards/organization.guard';

@Controller('organizations')
@UseGuards(OrganizationGuard)
```

```
export class OrganizationSettingsController {
  // Appropriate: organizationId is explicitly present in the route.
  @Get(':organizationId/settings')
  getSettings(@Param('organizationId') organizationId: string) {
    return {
      organizationId,
      message: 'Organization settings',
    };
  }
}
```

AI Coding Instructions

- Use `OrganizationGuard` only when the request explicitly includes `organizationId` in params, query, or body.
- Do not treat this guard as tenant protection for routes keyed only by child resource IDs such as `:projectId`, `:documentId`, `:versionId`, or generic `:id`.
- For resource-scoped routes, use `ResourceOrgGuard` together with `@RequireResourceOwnership(...)` so the resource's owning organization is resolved and verified.
- Preserve the guard's no-op behavior when no explicit `organizationId` is supplied; resource ownership enforcement belongs in the resource guard flow.

Referenced By

- `AuthModule` (MODULE_PROVIDES)
- `AuthModule` (MODULE_EXPORTS)