OrganizationService

September 4, 2026

OrganizationService

Kind: Service

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

`OrganizationService` encapsulates organization-level backend operations, including retrieving and updating organization data, managing members and settings, and maintaining user audience assignments. It is intended to be consumed by NestJS controllers or other application services that need organization-scoped data access and mutation workflows.

Methods

Method	Signature	Returns	Description
`list`	`list(user: JwtPayload)`	`Promise<Organization[]>`	
`get`	`get(id: string, user: JwtPayload)`	`Promise<Organization>`	
`update`	`update(id: string, dto: UpdateOrganizationDto, user: JwtPayload)`	`Promise<Organization>`	
`getMembers`	`getMembers(id: string, user: JwtPayload)`	`Promise<any[]>`	
`getSettings`	`getSettings(id: string, user: JwtPayload)`	`Promise<Record<string, any>>`	
`updateSettings`	`updateSettings(id: string, dto: UpdateOrgSettingsDto, user: JwtPayload)`	`Promise<Organization>`	
`getUserAudiences`	`getUserAudiences(organizationId: string, userId: string, currentUser: JwtPayload)`	`Promise<{ userId: string; audienceIds: string[] }>`	Get a user's audience memberships
`updateUserAudiences`	`updateUserAudiences(organizationId: string, userId: string, dto: UpdateUserAudiencesDto, currentUser: JwtPayload)`	`Promise<{ userId: string; audienceIds: string[] }>`	Update a user's audience memberships Only admins and

Method	Signature	Returns	Description
			maintainers can update user audiences

Dependencies

- `PrismaService`

Where it refuses work

- `OrganizationService` stops the work with `ForbiddenException` when `id !== user.organizationId` — “Access denied to this organization”, in 5 places.
- `OrganizationService` stops the work with `NotFoundException` when `!org` — “Organization not found”, in 2 places.
- `OrganizationService` stops the work with `ForbiddenException` when `organizationId !== currentUser.organizationId` — “Access denied to this organization”, in 2 places.
- `OrganizationService` stops the work with `NotFoundException` when `!targetUser` — “User not found”, in 2 places.
- `OrganizationService` stops the work with `ForbiddenException` when `targetUser.organizationId !== organizationId` — “User does not belong to this organization”, in 2 places.
- `OrganizationService` stops the work with `ForbiddenException` when `!allowedRoles.includes(user.role as UserRole)` — “Only administrators can update organization details”.

Diagram

```
sequenceDiagram
    participant Controller as NestJS Controller
    participant Service as OrganizationService
    participant Store as Organization Data Layer

    Controller->>Service: get() / list()
    Service->>Store: Fetch organization records
    Store-->>Service: Organization data
    Service-->>Controller: Organization or Organization[]

    Controller->>Service: updateSettings()
    Service->>Store: Persist settings changes
    Store-->>Service: Updated organization
    Service-->>Controller: Organization

    Controller->>Service: updateUserAudiences()
    Service->>Store: Save audience assignments
```

```
    Store-->>Service: Updated user audiences
    Service-->>Controller: { userId, audienceIds }
```

Usage

```
import { Injectable } from '@nestjs/common';
import { OrganizationService } from './organization.service';

@Injectable()
export class OrganizationAdminService {
  constructor(
    private readonly organizationService: OrganizationService,
  ) {}

  async updateOrganizationConfiguration() {
    const organization = await this.organizationService.get();

    const updatedOrganization = await this.organizationService.update();

    const settings = await this.organizationService.getSettings();

    await this.organizationService.updateSettings();

    const members = await this.organizationService.getMembers();

    const audiences = await this.organizationService.getUserAudiences();

    return {
      organization,
      updatedOrganization,
      settings,
      memberCount: members.length,
      audiences,
    };
  }
}
```

AI Coding Instructions

- Inject `OrganizationService` through NestJS dependency injection; do not instantiate it directly.
- Use `get()` for the current organization context and `list()` only when the caller genuinely needs all accessible organizations.
- Keep organization settings access centralized through `getSettings()` and `updateSettings()` rather than modifying settings through unrelated services.
- Preserve the `{ userId, audienceIds }` response shape when integrating `getUserAudiences()` and `updateUserAudiences()` with controllers or clients.

- Add authorization and organization-context validation at the controller or guard layer before calling mutation methods such as `update()`, `updateSettings()`, or `updateUserAudiences()`.

Relationships

- DEPENDS_ON → `PrismaService`

Referenced By

- `OrganizationController` (DEPENDS_ON)
- `OrganizationModule` (MODULE_PROVIDES)
- `OrganizationModule` (MODULE_EXPORTS)