PlatformService

September 4, 2026

PlatformService

Kind: Service

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

`PlatformService` is a NestJS backend service responsible for platform-level organization administration. It supports listing and retrieving organizations, updating organization plans, and managing organization SSO configuration—including repair and disable operations.

Methods

Method	Signature	Returns	Description
`listOrgs`	`listOrgs(params: { query?: string; plan?: string; page?: number; pageSize?: number })`	`unknown`	
`getOrg`	`getOrg(orgId: string)`	`unknown`	
`setPlan`	`setPlan(orgId: string, dto: SetPlanDto, actor: PlatformActor)`	`unknown`	
`getOrgSso`	`getOrgSso(orgId: string)`	`unknown`	Support view: the self-serve config PLUS a cert fingerprint (never the raw cert).
`repairSso`	`repairSso(orgId: string, dto: RepairSsoDto, actor: PlatformActor)`	`unknown`	
`disableSso`	`disableSso(orgId: string, reason: string, actor: PlatformActor)`	`unknown`	

Dependencies

- `PrismaService`
- `PlanService`
- `SamlService`
- `AuditService`

Where it refuses work

- `PlatformService` stops the work with `NotFoundException` when `!o` — “Organization not found”.
- `PlatformService` stops the work with `BadRequestException` when `!dto.plan && !dto.override` — “Provide `plan` and/or `override`.”.
- `PlatformService` stops the work with `NotFoundException` when `!before` — “Organization not found”.
- `PlatformService` stops the work with an early return when `!certBase64`.

When something fails

- `PlatformService` handles failure in 1 place: it turns it into a return value in all 1.

Diagram

```
sequenceDiagram
  participant Controller as Platform Controller
  participant Service as PlatformService
  participant Org as Organization Store
  participant SSO as SSO Provider

  Controller->>Service: listOrgs()
  Service->>Org: Query organizations
  Org-->>Service: Organization list
  Service-->>Controller: Organizations

  Controller->>Service: setPlan(orgId, plan)
  Service->>Org: Update subscription plan
  Org-->>Service: Updated organization
  Service-->>Controller: Updated organization

  Controller->>Service: repairSso(orgId)
  Service->>SSO: Validate or repair SSO setup
  SSO-->>Service: SSO status
  Service-->>Controller: Repaired SSO configuration
```

Usage

```
import { Injectable } from '@nestjs/common';
import { PlatformService } from './platform.service';

@Injectable()
export class PlatformAdminService {
  constructor(private readonly platformService: PlatformService) {}

  async updateOrganizationPlan(orgId: string) {
    const organization = await this.platformService.getOrg(orgId);

    await this.platformService.setPlan(orgId, {
      plan: 'enterprise',
    });

    const sso = await this.platformService.getOrgSso(orgId);

    return {
      organization,
      sso,
    };
  }

  async disableOrganizationSso(orgId: string) {
    return this.platformService.disableSso(orgId);
  }
}
```

AI Coding Instructions

- Keep platform-level organization operations inside `PlatformService`; controllers should validate input and delegate business logic to the service.
- Use `getOrg()` before destructive or configuration-changing operations such as plan updates, SSO repair, or SSO disablement.
- Treat SSO repair and disable actions as state-changing administrative operations; preserve existing authorization and audit-log integrations.
- Ensure plan values and SSO configuration payloads are validated through DTOs or schemas before calling service methods.
- Handle missing organizations and invalid SSO states consistently with the API's existing NestJS exception patterns.

Relationships

- DEPENDS_ON → `PrismaService`
- DEPENDS_ON → `PlanService`
- DEPENDS_ON → `SamlService`
- DEPENDS_ON → `AuditService`

Referenced By

- `PlatformController` (DEPENDS_ON)
- `PlatformModule` (MODULE_PROVIDES)