ScimService

September 4, 2026

ScimService

Kind: Service

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

SCIM provisioning logic. Every method takes an explicit `orgId` that the caller (controller) sourced from ScimAuthGuard's token — never from the request body. All 404/409 paths throw SCIM-shaped HttpExceptions.

`ScimService` implements SCIM provisioning operations for users and groups, including listing, retrieval, creation, replacement, patching, and user deactivation. Controllers pass an explicit `orgId` obtained from `ScimAuthGuard`, ensuring tenant scope is never derived from request payloads. Missing resources and conflicting operations are returned as SCIM-shaped `HttpException` responses.

Methods

Method	Signature	Returns	Description
`listUsers`	`listUsers(orgId: string, opts: { filter?: string; startIndex?: number; count?: number })`	`Promise<Record<string, unknown>>`	
`getUser`	`getUser(orgId: string, id: string)`	`Promise<ScimUserResource>`	
`createUser`	`createUser(orgId: string, resource: Partial<ScimUserResource>)`	`Promise<ScimUserResource>`	
`replaceUser`	`replaceUser(orgId: string, id: string, resource: Partial<ScimUserResource>)`	`Promise<ScimUserResource>`	PUT — full replace of the mutable attributes.

Method	Signature	Returns	Description
`patchUser`	`patchUser(orgId: string, id: string, body: { Operations?: ScimPatchOp[]; operations?: ScimPatchOp[] })`	`Promise<ScimUserResource>`	PATCH — RFC 7644, tolerant of BOTH Okta and Entra operation shapes.
`deactivateUser`	`deactivateUser(orgId: string, id: string)`	`Promise<void>`	DELETE → soft-deactivate.
`listGroups`	`listGroups(orgId: string)`	`Promise<Record<string, unknown>>`	
`getGroup`	`getGroup(orgId: string, id: string)`	`Promise<Record<string, unknown>>`	
`createGroup`	`createGroup(orgId: string, resource: { displayName?: string })`	`Promise<Record<string, unknown>>`	
`patchGroup`	`patchGroup(orgId: string, id: string, body: { Operations?: ScimPatchOp[]; operations?: ScimPatchOp[] })`	`Promise<Record<string, unknown>>`	Group PATCH: membership add/remove propagates the group's mapped role and audience onto the member users (that's what a group mapping is FOR).
`resolveDefaultRole`	`resolveDefaultRole(orgId: string)`	`Promise<UserRole>`	Role for a freshly-provisioned user: reuse SAML defaultRole if configured.

Dependencies

- `PrismaService`
- `AuthService`
- `ConfigService`
- `AuditService` *(optional)*

Where it refuses work

- `ScimService` stops the work with an early return when `!mapping`, in 2 places.
- `ScimService` stops the work with an early return when `!input.userName`.
- `ScimService` stops the work with an early return when `existing.organizationId !== orgId`.
- `ScimService` stops the work with an early return when `existing.scimExternalId`.
- `ScimService` stops the work with an early return when `!user.isActive`.
- `ScimService` stops the work with an early return when `!groupName`.

Diagram

```
sequenceDiagram
  participant IdP as Identity Provider
  participant Controller as SCIM Controller
  participant Guard as ScimAuthGuard
  participant Service as ScimService
  participant DB as Application Database

  IdP->>Controller: POST /scim/v2/Users
  Controller->>Guard: Validate SCIM bearer token
  Guard-->>Controller: Authenticated orgId
  Controller->>Service: createUser(orgId, scimPayload)
  Service->>DB: Create user within org scope
  DB-->>Service: User record
  Service-->>Controller: SCIM User resource
  Controller-->>IdP: 201 Created

  Note over Service: Not found and conflict cases throw<br/>SCIM-shaped HttpExceptions
```

Usage

```
import { Injectable } from '@nestjs/common';
import { ScimService } from './scim.service';
```

```
@Injectable()
export class ScimUsersController {
  constructor(private readonly scimService: ScimService) {}

  async createUserForOrganization(
    orgId: string,
    payload: {
      userName: string;
      name?: { givenName?: string; familyName?: string };
      emails?: Array<{ value: string; primary?: boolean }>;
      active?: boolean;
    },
  ) {
    // orgId must come from ScimAuthGuard/token context, never request body.
    return this.scimService.createUser(orgId, payload);
  }

  async deactivateUserForOrganization(orgId: string, userId: string) {
    await this.scimService.deactivateUser(orgId, userId);
  }
}
```

AI Coding Instructions

- Always pass the authenticated `orgId` from `ScimAuthGuard` or trusted request context; never accept organization scope from a SCIM request body.
- Preserve SCIM response shapes for user and group resources, including SCIM-compatible error responses for `404` and `409` cases.
- Scope every database lookup and mutation by `orgId` to prevent cross-organization user or group access.
- Use `replaceUser` for full resource replacement and `patchUser` / `patchGroup` for SCIM Patch operations; do not treat PATCH as an unrestricted partial database update.
- Keep controller responsibilities limited to authentication, request parsing, and HTTP translation; place provisioning and SCIM-specific business logic in `ScimService`.

Relationships

- DEPENDS_ON → `PrismaService`
- DEPENDS_ON → `AuthService`
- DEPENDS_ON → `configservice`
- DEPENDS_ON → `AuditService`

Referenced By

- `ScimGroupsController` (DEPENDS_ON)
- `ScimUsersController` (DEPENDS_ON)
- `ScimModule` (MODULE_PROVIDES)
- `ScimModule` (MODULE_EXPORTS)