# ScimAdminService

September 4, 2026

## ScimAdminService

**Kind:** Service

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

Admin-facing SCIM management: token lifecycle + group-mapping CRUD + a sync dashboard. Guarded upstream (JwtAuthGuard + RolesGuard ADMIN + PlanGuard BUSINESS); this service re-verifies the org matches the caller's JWT org so a param can never reach another tenant.

`ScimAdminService` provides tenant-safe administration for SCIM provisioning within an organization. It manages SCIM bearer-token lifecycle, group-mapping CRUD operations, and synchronization status data while re-validating that the requested organization matches the caller’s JWT organization. It is intended to run behind the API’s JWT, admin-role, and Business-plan guards.

### Methods

| Method | Signature | Returns | Description |
|---|---|---|---|
| `listTokens` | `listTokens(orgId: string, user: JwtPayload)` | `unknown` | |
| `createToken` | `createToken(orgId: string, user: JwtPayload, dto: { label?: string })` | `unknown` | Mint a token — the plaintext is returned ONCE and never persisted. |
| `revokeToken` | `revokeToken(orgId: string, user: JwtPayload, tokenId: string)` | `unknown` | |
| `listGroupMappings` | `listGroupMappings(orgId: string, user: JwtPayload)` | `unknown` | |
| `createGroupMapping` | `createGroupMapping(orgId: string, user: JwtPayload, dto: {` | `unknown` | |

| Method | Signature | Returns | Description |
|---|---|---|---|
| | `groupName: string; role?: UserRole; audienceId?: string })` | | |
| `updateGroupMapping` | `updateGroupMapping(orgId: string, user: JwtPayload, id: string, dto: { role?: UserRole | null; audienceId?: string | null })` |
| `deleteGroupMapping` | `deleteGroupMapping(orgId: string, user: JwtPayload, id: string)` | `unknown` | |
| `status` | `status(orgId: string, user: JwtPayload)` | `unknown` | |

## Dependencies

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

## Where it refuses work

- `ScimAdminService` stops the work with `NotFoundException` when `!mapping` — “Group mapping not found”, in 2 places.
- `ScimAdminService` stops the work with `NotFoundException` when `!token` — “SCIM token not found”.
- `ScimAdminService` stops the work with `ForbiddenException` when `!groupName` — “groupName is required”.
- `ScimAdminService` stops the work with `ForbiddenException` when `orgId !== user.organizationId` — “Access denied to this organization”.
- `ScimAdminService` stops the work with `NotFoundException` when `!audience` — “Audience not found in this organization”.
- `ScimAdminService` stops the work with an early return when `token.revokedAt`.

## Diagram

```
sequenceDiagram
  participant Admin as Admin Client
  participant Guard as Auth / Role / Plan Guards
  participant Controller as SCIM Admin Controller
  participant Service as ScimAdminService
  participant DB as Database
```

```
  participant SCIM as SCIM Sync Provider

  Admin->>Guard: Request with JWT and org parameter
  Guard->>Guard: Validate JWT, ADMIN role, BUSINESS plan
  Guard->>Controller: Authorized request
  Controller->>Service: Invoke admin operation(orgId, caller)
  Service->>Service: Verify caller JWT org matches orgId

  alt Token lifecycle
    Service->>DB: List, create, or revoke SCIM token
    DB-->>Service: Token result
  else Group mapping CRUD
    Service->>DB: List, create, update, or delete mapping
    DB-->>Service: Mapping result
  else Sync dashboard
    Service->>DB: Read sync configuration and history
    Service->>SCIM: Read sync/provider status
    SCIM-->>Service: Current sync state
  end

  Service-->>Controller: Tenant-scoped result
  Controller-->>Admin: Response
```

## Usage

```
import { Injectable, ForbiddenException } from '@nestjs/common';
import { ScimAdminService } from './scim-admin.service';

@Injectable()
export class ScimAdminController {
  constructor(private readonly scimAdminService: ScimAdminService) {}

  async createProvisioningToken(
    organizationId: string,
    currentUser: { organizationId: string; id: string },
  ) {
    // Route guards should already enforce authentication, ADMIN role,
    // and BUSINESS plan access.
    if (currentUser.organizationId !== organizationId) {
      throw new ForbiddenException('Organization access denied');
    }

    return this.scimAdminService.createToken(
      organizationId,
      currentUser,
    );
  }

  async addGroupMapping(
    organizationId: string,
```

```
    currentUser: { organizationId: string; id: string },
  ) {
    return this.scimAdminService.createGroupMapping(
      organizationId,
      currentUser,
      {
        sourceGroupId: 'idp-engineering',
        targetGroupId: 'app-engineering',
      },
    );
  }

  async getSyncStatus(
    organizationId: string,
    currentUser: { organizationId: string; id: string },
  ) {
    return this.scimAdminService.status(organizationId, currentUser);
  }
}
```

## AI Coding Instructions

- Preserve tenant isolation on every method: validate that the requested organization ID matches the organization claim from the authenticated caller.
- Keep authorization in the controller/guard layer ( `JwtAuthGuard` , `RolesGuard` with `ADMIN` , and `PlanGuard` with `BUSINESS` ), but do not remove the service-level organization verification.
- Treat SCIM tokens as secrets: return plaintext only at creation time if supported, store only safe token representations, and revoke rather than mutate issued tokens.
- Scope all token, mapping, and sync-status database queries by organization ID; never fetch a record by its ID alone without tenant filtering.
- When changing group mappings, ensure downstream SCIM synchronization/dashboard status remains consistent with the mapping configuration.

## Relationships

- DEPENDS_ON → `PrismaService`
- DEPENDS_ON → `AuditService`

## Referenced By

- `ScimAdminController` (DEPENDS_ON)
- `ScimModule` (MODULE_PROVIDES)
- `ScimModule` (MODULE_EXPORTS)