ProjectAccessService

September 4, 2026

ProjectAccessService

Kind: Service

Source: `atloria-monorepo/apps/api/src/reader-access/project-access.service.ts`

ProjectAccessService — owner-side management of the A3 access policy and share links (MAINTAINER/ADMIN, plan-gated). Org ownership is verified from the caller's JWT org (never params) per the cross-tenant lessons. Changing the mode or the password BUMPS tokenVersion, which instantly revokes every outstanding reader token (short-expiry + ver, no deny-list).

`ProjectAccessService` manages owner-side A3 reader access policies and share links for projects. It verifies organization ownership exclusively from the caller’s JWT context, enforces MAINTAINER/ADMIN and plan-gated capabilities, and revokes active reader tokens by incrementing `tokenVersion` whenever the access mode or password changes.

Methods

Method	Signature	Returns
`getPolicy`	`getPolicy(projectId: string, user: JwtPayload)`	`unknown`
`updatePolicy`	`updatePolicy(projectId: string, dto: UpdateAccessPolicyDto, user: JwtPayload)`	`unknown`
`listShareLinks`	`listShareLinks(projectId: string, user: JwtPayload)`	`unknown`
`createShareLink`	`createShareLink(projectId: string, dto: CreateShareLinkDto, user: JwtPayload)`	`unknown`
`revokeShareLink`	`revokeShareLink(projectId: string, linkId: string, user: JwtPayload)`	`unknown`

Dependencies

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

Where it refuses work

- `ProjectAccessService` stops the work with `NotFoundException` when `!project` — “Project not found”.
- `ProjectAccessService` stops the work with `ForbiddenException` when `project.organizationId !== user.organizationId` — “Access denied to this project”.
- `ProjectAccessService` stops the work with `BadRequestException` when `nextMode === 'PASSWORD' && !existing?.passwordHash && !dto.password` — “Set a password before switching to password protection.”.
- `ProjectAccessService` stops the work with `BadRequestException` when `Number.isNaN(d.getTime())` — “Invalid expiry date.”.
- `ProjectAccessService` stops the work with `NotFoundException` when `!link` — “Share link not found”.
- `ProjectAccessService` stops the work with an early return when `this.plans.meetsPlan(actual, required)`.

Diagram

```
sequenceDiagram
    participant Client as Owner Client
    participant Controller as Project Access Controller
    participant Service as ProjectAccessService
    participant JWT as JWT Organization Context
    participant DB as Database
    participant Reader as Reader Token Validator

    Client->>Controller: Update policy / create share link
    Controller->>Service: request + caller JWT context
    Service->>JWT: Read caller organization identity
    Service->>DB: Verify project belongs to JWT org
    DB-->>Service: Ownership and current policy
    Service->>Service: Check role and plan entitlement

    alt Mode or password changed
        Service->>DB: Update policy and increment tokenVersion
        DB-->>Service: Updated policy
        Note over Reader: Existing tokens fail version validation
```

```
    else Share link operation
        Service->>DB: Create, list, or revoke share link
        DB-->>Service: Share link result
    end

    Service-->>Controller: Policy or share-link response
    Controller-->>Client: Authorized result
```

Usage

```
import { ProjectAccessService } from './project-access.service';

@Injectable()
export class ProjectAccessController {
  constructor(
    private readonly projectAccessService: ProjectAccessService,
  ) {}

  @Patch(':projectId/access-policy')
  async updateAccessPolicy(
    @Param('projectId') projectId: string,
    @Body()
    body: {
      mode: 'PRIVATE' | 'LINK' | 'PASSWORD';
      password?: string;
    },
    @Req() request: AuthenticatedRequest,
  ) {
    // The service must derive organization ownership from request.user,
    // not from an organization ID supplied in route params or body data.
    return this.projectAccessService.updatePolicy(
      projectId,
      body,
      request.user,
    );
  }

  @Post(':projectId/share-links')
  async createShareLink(
    @Param('projectId') projectId: string,
    @Req() request: AuthenticatedRequest,
  ) {
    return this.projectAccessService.createShareLink(
      projectId,
      request.user,
    );
  }
}
```

AI Coding Instructions

- Always derive the organization identity from the authenticated JWT caller context; never trust organization IDs provided through request parameters or request bodies.
- Require the appropriate owner role (`MAINTAINER` or `ADMIN`) and validate plan entitlements before changing policies or managing share links.
- Increment `tokenVersion` whenever an access mode or password changes so all previously issued reader tokens are immediately invalidated.
- Keep reader tokens short-lived and validate their embedded version against the current policy version; do not introduce a token deny-list unless the authentication design changes.
- Treat share-link creation and revocation as project-scoped operations and verify project ownership before reading or mutating records.

Relationships

- DEPENDS_ON → `PrismaService`
- DEPENDS_ON → `PlanService`
- DEPENDS_ON → `DocsAccessService`
- DEPENDS_ON → `AuditService`

Referenced By

- `ProjectAccessController` (DEPENDS_ON)
- `ReaderAccessModule` (MODULE_PROVIDES)