ProjectAccessController

September 4, 2026

ProjectAccessController

Kind: Controller

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

Owner API for gated/private docs (A3). MAINTAINER/ADMIN only; the service additionally verifies the project belongs to the caller's org and applies the per-mode plan gate (PASSWORD=PRO; EMAIL_DOMAIN/SSO/PRIVATE/audience-rules +
share links=BUSINESS/PRO). Password hashes and share-link tokens never leave the server except the one-time plaintext token at creation.

`ProjectAccessController` provides owner-facing APIs for configuring gated and private documentation access. It restricts operations to `MAINTAINER` and `ADMIN` users, verifies that the requested project belongs to the caller’s organization, and delegates plan eligibility and access-mode validation to the access service.

Sensitive values remain server-side: password hashes are never returned, and share-link tokens are returned only once when a link is created.

Diagram

```
graph LR
  Client[Owner Dashboard / API Client] --> Controller[ProjectAccessController]
  Controller --> Auth[Auth + Role Guard]
  Auth --> OrgCheck[Project Organization Verification]
  OrgCheck --> Service[Project Access Service]
  Service --> PlanGate[Plan / Feature Gate]
  PlanGate --> Modes[Access Modes]

  Modes --> Password[PASSWORD<br/>PRO plan]
  Modes --> Domain[EMAIL_DOMAIN / SSO / PRIVATE<br/>BUSINESS or PRO plan]
  Modes --> Audience[Audience Rules<br/>BUSINESS or PRO plan]
  Modes --> Share[Share Links<br/>BUSINESS or PRO plan]

  Service --> Storage[(Project Access Configuration)]
```

```
  Service --> Response[Sanitized API Response]
  Response --> Client
```

Usage

```
import { Controller, Get, Param, Patch, Body } from '@nestjs/common';
import { ProjectAccessController } from './project-access.controller';

@Controller('projects')
export class OwnerProjectSettingsController {
  constructor(
    private readonly projectAccessController: ProjectAccessController,
  ) {}

  @Patch(':projectId/access')
  async updateAccess(
    @Param('projectId') projectId: string,
    @Body()
    body: {
      mode: 'PASSWORD' | 'EMAIL_DOMAIN' | 'SSO' | 'PRIVATE';
      password?: string;
      allowedDomains?: string[];
    },
  ) {
    // In normal application code, call the underlying access service directly.
    // The controller is invoked by NestJS through its registered routes.
    return this.projectAccessController.updateProjectAccess(
      projectId,
      body,
    );
  }

  @Get(':projectId/access')
  async getAccess(@Param('projectId') projectId: string) {
    return this.projectAccessController.getProjectAccess(projectId);
  }
}
```

AI Coding Instructions

- Keep authorization at the controller boundary: access-management endpoints must require `MAINTAINER` or `ADMIN` membership.
- Always verify project ownership through the caller's organization before reading or mutating access settings.
- Delegate feature eligibility to the plan-gating service; do not hardcode plan checks independently in new endpoints.

- Never expose password hashes or persisted share-link tokens in responses; return a plaintext share token only during creation.
- Validate access-mode-specific fields, such as requiring a password for `PASSWORD` mode and domains for `EMAIL_DOMAIN` mode.

Relationships

- MODULE_DECLARES → `getPolicy`
- MODULE_DECLARES → `updatePolicy`
- MODULE_DECLARES → `listShareLinks`
- MODULE_DECLARES → `createShareLink`
- MODULE_DECLARES → `revokeShareLink`
- DEPENDS_ON → `ProjectAccessService`

Referenced By

- `ReaderAccessModule` (MODULE_DECLARES)