DocsConfigController

September 4, 2026

DocsConfigController

Kind: Controller

Source: `atloria-monorepo/apps/api/src/change-request/docs-config.controller.ts`

Per-project docs config (the S2 review gate). Sibling of ChangeRequestController with the
SAME class-level authorizer: every route is keyed on `:projectId`, and OrganizationGuard
no-ops when the request carries no `organizationId`, so ProjectOrgGuard (which loads the
project and enforces org ownership) is what protects GET and PATCH alike.

GET is open to any project member (the editor reads it to decide whether saving must go
through a change request); PATCH is MAINTAINER/ADMIN only.

`DocsConfigController` manages per-project documentation configuration, including the S2 review gate that determines whether edits must flow through a change request. It exposes read access to project members and restricts updates to `MAINTAINER` or `ADMIN` members. Project ownership and organization access are enforced through `ProjectOrgGuard`.

Diagram

```
graph LR
  Editor[Project member / editor] -->|GET /projects/:projectId/docs-config| Controller[DocsConf
  Maintainer[Maintainer or admin] -->|PATCH /projects/:projectId/docs-config| Controller

  Controller --> ProjectGuard[ProjectOrgGuard]
  ProjectGuard --> Project[Load project and verify organization ownership]

  Controller --> Service[Docs config service]
  Service --> Config[(Per-project docs config)]
```

```
  Config --> Gate{S2 review gate enabled?}
  Gate -->|Yes| ChangeRequest[Create or update through change request]
  Gate -->|No| DirectEdit[Allow direct documentation save]
```

Usage

```
const projectId = "project_123";
const baseUrl = "https://api.example.com";
const headers = {
  Authorization: `Bearer ${accessToken}`,
  "Content-Type": "application/json",
};

// Read the project's documentation configuration.
// Available to members of the project.
const currentConfigResponse = await fetch(
  `${baseUrl}/projects/${projectId}/docs-config`,
  { headers },
);

const currentConfig = await currentConfigResponse.json();

// Update the configuration.
// Requires a MAINTAINER or ADMIN role in the project.
const updateResponse = await fetch(
  `${baseUrl}/projects/${projectId}/docs-config`,
  {
    method: "PATCH",
    headers,
    body: JSON.stringify({
      ...currentConfig,
      // Apply supported docs-config fields here.
      // Example: requireChangeRequestForDocs: true,
    }),
  },
);

if (!updateResponse.ok) {
  throw new Error(`Unable to update docs config: ${updateResponse.status}`);
}

const updatedConfig = await updateResponse.json();
```

AI Coding Instructions

- Keep every route scoped by `:projectId`; this controller relies on project context rather than a request-level `organizationId`.

- Preserve `ProjectOrgGuard` on both read and update paths so the project is loaded and organization ownership is verified consistently.
- Allow project members to read the config, but enforce `MAINTAINER` or `ADMIN` authorization for `PATCH` operations.
- Treat the docs configuration as the source of truth for editor save behavior: when the S2 gate is enabled, integrations should create a change request instead of directly saving documentation.
- When adding configuration fields, update the corresponding DTO, validation rules, service persistence, and editor-side config consumption together.

Relationships

- MODULE_DECLARES → `get`
- MODULE_DECLARES → `set`
- DEPENDS_ON → `DocsConfigService`

Referenced By

- `ChangeRequestModule` (MODULE_DECLARES)