ChangeRequestController

September 4, 2026

ChangeRequestController

Kind: Controller

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

Docs change requests (S2.1), project-scoped. POST creates a 'draft' row and enqueues the
branch-staging job — the response is the draft; the worker flips it to 'open' (with baseCommit/headCommit/diffCache) or 'failed' asynchronously. Poll GET :crId for the outcome.

ProjectOrgGuard (not OrganizationGuard) is the class-level authorizer: every route is keyed
on `:projectId`, and OrganizationGuard no-ops when the request carries no `organizationId`,
so it would let a user from another org act on this project. ProjectOrgGuard loads the project and enforces org ownership on GET and POST alike.

`ChangeRequestController` manages project-scoped documentation change requests. Creating a change request returns a `draft` immediately and queues branch staging asynchronously; clients should poll the detail endpoint until the worker transitions it to `open` or `failed`. All routes use `ProjectOrgGuard` to load the project and enforce organization ownership.

Diagram

```
graph LR
  Client[Client] -->|POST /projects/:projectId/change-requests| Controller[ChangeRequestControl
  Controller --> Guard[ProjectOrgGuard]
  Guard --> Project[Project + organization authorization]
  Controller --> Service[ChangeRequestService]
  Service --> Draft[Create draft change request]
  Service --> Queue[Enqueue branch-staging job]
  Queue --> Worker[Branch-staging worker]
```

```
  Worker -->|success| Open[Status: open<br/>baseCommit, headCommit, diffCache]
  Worker -->|failure| Failed[Status: failed]

  Client -->|GET /projects/:projectId/change-requests/:crId| Controller
  Controller --> Service
  Service --> Draft
  Service --> Open
  Service --> Failed
```

Usage

```
const projectId = 'project_123';

// Create a draft change request. Staging happens asynchronously.
const createResponse = await fetch(
  `/projects/${projectId}/change-requests`,
  {
    method: 'POST',
    headers: {
      Authorization: `Bearer ${accessToken}`,
      'Content-Type': 'application/json',
    },
    body: JSON.stringify({
      title: 'Update API authentication docs',
      description: 'Clarify bearer token requirements.',
    }),
  },
);

const draft = await createResponse.json();
// draft.status is initially "draft"

// Poll until the background branch-staging job completes.
const statusResponse = await fetch(
  `/projects/${projectId}/change-requests/${draft.id}`,
  {
    headers: { Authorization: `Bearer ${accessToken}` },
  },
);

const changeRequest = await statusResponse.json();

if (changeRequest.status === 'open') {
  console.log(changeRequest.baseCommit, changeRequest.headCommit);
  console.log(changeRequest.diffCache);
} else if (changeRequest.status === 'failed') {
  console.error('Branch staging failed');
}
```

AI Coding Instructions

- Keep every route project-scoped through `:projectId`; do not introduce unscoped change-request access paths.
- Use `ProjectOrgGuard` at the controller level rather than `OrganizationGuard`, since project ownership must be validated even when `organizationId` is absent from the request.
- Treat `POST` responses as asynchronous draft creation only; branch commits and diffs are populated later by the staging worker.
- Preserve the `draft` → `open` or `failed` lifecycle, and ensure clients can retrieve the final state through `GET :crId`.
- When adding creation fields or worker behavior, update both the request DTO/service flow and the branch-staging job payload.

Relationships

- MODULE_DECLARES → `open`
- MODULE_DECLARES → `list`
- MODULE_DECLARES → `get`
- MODULE_DECLARES → `close`
- MODULE_DECLARES → `merge`
- MODULE_DECLARES → `previewLink`
- MODULE_DECLARES → `refreshDiff`
- MODULE_DECLARES → `resolve`
- DEPENDS_ON → `ChangeRequestService`
- DEPENDS_ON → `PreviewService`

Referenced By

- `ChangeRequestModule` (MODULE_DECLARES)