ChangelogController

September 4, 2026

ChangelogController

Kind: Controller

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

Owner surface for the B5 changelog: drafts inbox (list/edit/publish/discard), manual entries, and the auto-publish + subscriber-count settings.

`ChangelogController` exposes the backend owner surface for managing the B5 changelog. It supports reviewing and editing generated drafts, publishing or discarding drafts, creating manual changelog entries, and updating auto-publish and subscriber-count settings.

Diagram

```
graph LR
  Admin[Owner/Admin Client] --> Controller[ChangelogController]

  Controller --> Drafts[Draft Inbox]
  Controller --> Entries[Manual Changelog Entries]
  Controller --> Settings[Changelog Settings]

  Drafts --> Edit[Edit Draft]
  Drafts --> Publish[Publish Draft]
  Drafts --> Discard[Discard Draft]

  Publish --> Subscribers[Subscriber Notifications]
  Settings --> AutoPublish[Auto-publish Rules]
  Settings --> SubscriberCount[Subscriber-count Settings]
```

Usage

```
const baseUrl = 'https://api.example.com/changelog';
const headers = {
  Authorization: `Bearer ${ownerAccessToken}`,
```

```
  'Content-Type': 'application/json',
};

// Load changelog drafts awaiting review.
const drafts = await fetch(`${baseUrl}/drafts`, { headers }).then((response) =>
  response.json(),
);

// Edit and publish a selected draft.
const draftId = drafts[0].id;

await fetch(`${baseUrl}/drafts/${draftId}`, {
  method: 'PATCH',
  headers,
  body: JSON.stringify({
    title: 'Improved workspace navigation',
    content: 'Navigation is now faster and easier to use.',
  }),
});

await fetch(`${baseUrl}/drafts/${draftId}/publish`, {
  method: 'POST',
  headers,
});

// Update auto-publish settings.
await fetch(`${baseUrl}/settings`, {
  method: 'PATCH',
  headers,
  body: JSON.stringify({
    autoPublishEnabled: true,
    subscriberCountEnabled: true,
  }),
});
```

AI Coding Instructions

- Keep controller methods thin: validate request DTOs, enforce authorization, and delegate business logic to the changelog service layer.
- Preserve the draft lifecycle explicitly—editing, publishing, and discarding should only operate on valid draft states.
- Treat publishing as a side-effecting operation; ensure notification and subscriber-count updates are coordinated and safe to retry.
- Use DTO validation and consistent response shapes for manual entries, draft updates, and settings changes.

- Protect all owner-facing routes with the appropriate NestJS authentication and role/permission guards.

Relationships

- MODULE_DECLARES → `list`
- MODULE_DECLARES → `get`
- MODULE_DECLARES → `create`
- MODULE_DECLARES → `update`
- MODULE_DECLARES → `publish`
- MODULE_DECLARES → `discard`
- MODULE_DECLARES → `getConfig`
- MODULE_DECLARES → `setConfig`
- DEPENDS_ON → `ChangelogDraftsService`

Referenced By

- `ChangelogModule` (MODULE_DECLARES)