ContentTasksController

September 4, 2026

ContentTasksController

Kind: Controller

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

Analytics action queue — project-scoped ContentTask CRUD. Every route rides `:projectId` and is org-verified fail-closed via ResourceOrgGuard(kind: 'project'); the service additionally filters every query by projectId. Members are scoped to the caller's own organization (resolved from the verified JWT, not the path).

`ContentTasksController` exposes project-scoped CRUD endpoints for analytics action queue `ContentTask` records. Every route is protected by `ResourceOrgGuard(kind: 'project')`, which verifies project organization access fail-closed, while the underlying service applies `projectId` filtering to every query. Member-related operations resolve organization scope from the verified JWT rather than trusting organization identifiers in request paths.

Diagram

```
graph LR
  Client[Authenticated Client] --> Controller[ContentTasksController]
  Controller --> Guard[ResourceOrgGuard<br/>kind: project]
  Guard --> JWT[Verified JWT Organization]
  Guard -->|Authorized project access| Service[ContentTasksService]
  Service -->|Filter by projectId| Database[(ContentTask Records)]

  JWT -->|Organization scope| Service
  Guard -->|Unauthorized| Denied[403 Forbidden]
```

Usage

```
// Example request flow for a project-scoped content task.
//
// POST /projects/proj_123/content-tasks
```

```
// Authorization: Bearer <verified-jwt>

const response = await fetch(
  `${API_URL}/projects/proj_123/content-tasks`,
  {
    method: 'POST',
    headers: {
      Authorization: `Bearer ${accessToken}`,
      'Content-Type': 'application/json',
    },
    body: JSON.stringify({
      title: 'Review landing page copy',
      status: 'pending',
      priority: 'high',
    }),
  },
);

if (!response.ok) {
  throw new Error(`Unable to create content task: ${response.statusText}`);
}

const contentTask = await response.json();

// Fetch tasks for the same verified project scope.
const tasksResponse = await fetch(
  `${API_URL}/projects/proj_123/content-tasks`,
  {
    headers: {
      Authorization: `Bearer ${accessToken}`,
    },
  },
);

const tasks = await tasksResponse.json();
console.log(contentTask, tasks);
```

AI Coding Instructions

- Keep all controller routes nested beneath `:projectId`; treat the project path parameter as the resource scope for every CRUD operation.
- Preserve `ResourceOrgGuard(kind: 'project')` on project-scoped endpoints. Do not replace verified authorization with client-provided organization IDs.
- Ensure service methods filter reads, updates, and deletes by `projectId`, not only by the content task ID.
- Resolve member or organization context from the verified JWT/request context, never from an unverified route parameter or request body.

- Return standard NestJS DTO-based validation errors and authorization responses; do not leak whether tasks exist outside the caller’s organization.

Relationships

- MODULE_DECLARES → `list`
- MODULE_DECLARES → `members`
- MODULE_DECLARES → `create`
- MODULE_DECLARES → `update`
- MODULE_DECLARES → `remove`
- DEPENDS_ON → `ContentTasksService`

Referenced By

- `ContentTasksModule` (MODULE_DECLARES)