ToursController

September 4, 2026

ToursController

Kind: Controller

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

Member-gated orchestration CRUD (D2): owners define tour/checklist/ announcement/nps experiences, target them, publish/pause them and read the per-experience funnel. Every route is :projectId-scoped and org-verified by ResourceOrgGuard (fail-closed) + a service-level re-check.

`ToursController` exposes project-scoped endpoints for managing member-gated experiences, including tours, checklists, announcements, and NPS flows. It coordinates creation, targeting, publication state, and funnel analytics while enforcing organization access through `ResourceOrgGuard` and service-level authorization re-checks.

Diagram

```
graph LR
  Client[Authenticated member] --> Controller[ToursController]
  Controller --> Guard[ResourceOrgGuard]
  Guard -->|Verified project organization| Controller
  Controller --> Service[ToursService]
  Service --> Auth[Service-level org re-check]
  Service --> Store[(Experience data)]
  Service --> Funnel[Funnel analytics]

  Controller --> CRUD[Create / Read / Update / Delete]
  Controller --> State[Publish / Pause]
  Controller --> Targeting[Audience targeting]
  Controller --> Analytics[Per-experience funnel]
```

Usage

```ts
// Example API client usage for a project-scoped tour experience.
const projectId = "proj_123";
const token = process.env.API_TOKEN!;

const response = await fetch(
  `https://api.example.com/projects/${projectId}/tours`,
  {
    method: "POST",
    headers: {
      Authorization: `Bearer ${token}`,
      "Content-Type": "application/json",
    },
    body: JSON.stringify({
      type: "tour",
      name: "Welcome tour",
      target: {
        segmentIds: ["new-members"],
      },
      steps: [
        {
          title: "Welcome",
          content: "Start here to learn the product.",
          target: "[data-tour='dashboard']",
        },
      ],
    }),
  },
);

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

const tour = await response.json();

// Publish the created experience when it is ready.
await fetch(
  `https://api.example.com/projects/${projectId}/tours/${tour.id}/publish`,
  {
    method: "POST",
    headers: { Authorization: `Bearer ${token}` },
  },
);
```

AI Coding Instructions

- Keep every route scoped by `:projectId`; do not add unscoped experience endpoints.

- Preserve `ResourceOrgGuard` on protected routes and retain the service-level organization verification as defense in depth.
- Route mutations through the service layer rather than placing targeting, publishing, or analytics logic in the controller.
- Validate experience type and payload shape consistently for tours, checklists, announcements, and NPS experiences.
- Treat publish/pause actions as state transitions and ensure funnel queries remain scoped to both the project and experience.

Relationships

- MODULE_DECLARES → `create`
- MODULE_DECLARES → `list`
- MODULE_DECLARES → `get`
- MODULE_DECLARES → `update`
- MODULE_DECLARES → `publish`
- MODULE_DECLARES → `pause`
- MODULE_DECLARES → `archive`
- MODULE_DECLARES → `remove`
- MODULE_DECLARES → `funnel`
- MODULE_DECLARES → `validate`
- DEPENDS_ON → `ToursService`
- DEPENDS_ON → `ToursHealingService`

Referenced By

- `ToursModule` (MODULE_DECLARES)