AgentAnalyticsService

September 4, 2026

AgentAnalyticsService

Kind: Service

Source: `atloria-monorepo/apps/api/src/events/agent-analytics.service.ts`

`AgentAnalyticsService` is a NestJS backend service that provides aggregated analytics for agent-related activity. It exposes traffic and action metrics for use by API controllers, dashboards, reporting jobs, or event-monitoring workflows.

Methods

Method	Signature	Returns	Description
`agentTraffic`	`agentTraffic(projectId: string, organizationId: string, days: unknown)`	`Promise<AgentTraffic>`	Agent traffic for one project over a rolling window (org-scoped).
`agentActions`	`agentActions(projectId: string, organizationId: string, days: unknown)`	`Promise<AgentActions>`	Agent ACTIONS for one project over a rolling window (org-scoped) — what agents DO with the live API via `call_operation` (C4).

Dependencies

- `PrismaService`

Where it refuses work

- `AgentAnalyticsService` stops the work with `ForbiddenException` when `!project` — “Project not found in your organization.”, in 2 places.

Diagram

```
sequenceDiagram
  participant Consumer as Controller / Job
  participant Service as AgentAnalyticsService
  participant Events as Event Data Source

  Consumer->>Service: agentTraffic()
  Service->>Events: Query traffic-related events
  Events-->>Service: Traffic metrics
  Service-->>Consumer: Promise<AgentTraffic>

  Consumer->>Service: agentActions()
  Service->>Events: Query agent action events
  Events-->>Service: Action metrics
  Service-->>Consumer: Promise<AgentActions>
```

Usage

```
import { Controller, Get } from '@nestjs/common';
import { AgentAnalyticsService } from './agent-analytics.service';

@Controller('analytics/agents')
export class AgentAnalyticsController {
  constructor(
    private readonly agentAnalyticsService: AgentAnalyticsService,
  ) {}

  @Get('traffic')
  async getTraffic() {
    return this.agentAnalyticsService.agentTraffic();
  }

  @Get('actions')
  async getActions() {
    return this.agentAnalyticsService.agentActions();
  }
}
```

AI Coding Instructions

- Keep analytics retrieval logic in `AgentAnalyticsService`; controllers should only delegate requests and return results.
- Preserve the `Promise<AgentTraffic>` and `Promise<AgentActions>` return contracts when changing metric calculations.

- Ensure event queries and aggregations consistently use the same time ranges, filters, and agent identifiers across both methods.
- Avoid exposing raw event records when callers only require aggregated analytics data.
- Add or update tests when changing event classifications, aggregation rules, or analytics response shapes.

Relationships

- DEPENDS_ON → `PrismaService`

Referenced By

- `AgentAnalyticsController` (DEPENDS_ON)
- `EventsModule` (MODULE_PROVIDES)