AuditService

September 4, 2026

AuditService

Kind: Service

Source: `atloria-monorepo/apps/api/src/audit/audit.service.ts`

`AuditService` centralizes audit-log persistence, search, aggregation, and export behavior for the API. It records application events, exposes query and facet operations for audit views, and generates CSV output for reporting or compliance workflows. As a NestJS service, it is typically injected into controllers or domain services that need to capture auditable actions.

Methods

Method	Signature	Returns	Description
`record`	`record(event: AuditEventInput)`	`Promise<void>`	Record an audit event.
`query`	`query(opts: {		

```
organizationId: string;
projectId?: string;
docVersionId?: string;
actorId?: string;
action?: string;
entityType?: string;
startDate?: string;
endDate?: string;
limit?: number;
offset?: number;
```

}) | unknown | `Query audit trail with filters.` | | facets | facets(organizationId: string) | unknown | `Distinct actions + actors for an organization — feeds the filter dropdowns of the org-wide audit view.` | | exportCSV | exportCSV(opts: { organizationId: string; projectId?: string }) | Promise` | Export audit trail as CSV. |

Dependencies

- `PrismaService`

When something fails

- `AuditService` handles failure in 1 place: it logs it and continues in all 1.

Diagram

```
sequenceDiagram
    participant Client
    participant Controller
    participant AuditService
    participant AuditStore as Audit Repository/Store

    Client->>Controller: Request that performs an auditable action
    Controller->>AuditService: record(auditEvent)
    AuditService->>AuditStore: Persist audit record
    AuditStore-->>AuditService: Record saved
    AuditService-->>Controller: Promise<void>

    Client->>Controller: Request audit history/export
    Controller->>AuditService: query(filters) / facets(filters)
    AuditService->>AuditStore: Search and aggregate records
    AuditStore-->>AuditService: Results/facets
    AuditService-->>Controller: Audit data

    Client->>Controller: Export audit CSV
    Controller->>AuditService: exportCSV(filters)
    AuditService->>AuditStore: Retrieve matching records
    AuditStore-->>AuditService: Audit records
    AuditService-->>Controller: CSV string
```

Usage

```
import { Injectable } from '@nestjs/common';
import { AuditService } from './audit/audit.service';

@Injectable()
export class UserService {
  constructor(private readonly auditService: AuditService) {}

  async deactivateUser(actorId: string, userId: string) {
    // Perform the domain operation first.
    // await this.usersRepository.deactivate(userId);
```

```
    await this.auditService.record({
      actorId,
      action: 'user.deactivated',
      resourceType: 'user',
      resourceId: userId,
      timestamp: new Date(),
    });
  }

  async getAuditLog(filters: Record<string, unknown>) {
    return this.auditService.query(filters);
  }

  async exportAuditLog(filters: Record<string, unknown>): Promise<string> {
    return this.auditService.exportCSV(filters);
  }
}
```

AI Coding Instructions

- Inject `AuditService` through NestJS dependency injection; do not instantiate it directly.
- Record audit events after successful domain mutations so logs do not describe failed operations.
- Include consistent actor, action, resource, and timestamp metadata when calling `record()`.
- Reuse the same filter shape across `query()`, `facets()`, and `exportCSV()` to keep UI results and exports aligned.
- Treat CSV output as response content: set appropriate download headers and avoid exposing audit fields that callers are not authorized to access.

Relationships

- DEPENDS_ON → `PrismaService`

Referenced By

- `AuditController` (DEPENDS_ON)
- `AuditModule` (MODULE_PROVIDES)
- `AuditModule` (MODULE_EXPORTS)
- `AuthService` (DEPENDS_ON)
- `SamlController` (DEPENDS_ON)

- `BrandingService` (DEPENDS_ON)
- `DocVersionService` (DEPENDS_ON)
- `DocVersionRetentionService` (DEPENDS_ON)
- `ScheduledPublishService` (DEPENDS_ON)
- `InvitationService` (DEPENDS_ON)
- `PlatformService` (DEPENDS_ON)
- `ProjectService` (DEPENDS_ON)
- `ProjectAccessService` (DEPENDS_ON)
- `ScimAdminService` (DEPENDS_ON)
- `ScimService` (DEPENDS_ON)