InvitationService

September 4, 2026

InvitationService

Kind: Service

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

Invitation-token flow — the ONLY path into an existing organization.
(Open registration always creates a fresh org; see AuthService.register.)

`InvitationService` manages the invitation-token workflow, which is the only supported way for a user to join an existing organization. It creates invitation records, resolves invitations by token, and accepts valid tokens to add the authenticated user to the invited organization. Open registration is intentionally separate and always creates a new organization.

Methods

Method	Signature	Returns	Description
`create`	`create(organizationId: string, dto: CreateInvitationDto, user: JwtPayload)`	`unknown`	Create an invitation and email the accept link to the invitee.
`getByToken`	`getByToken(token: string)`	`unknown`	Public info about an invitation (for the accept page).
`accept`	`accept(token: string, dto: AcceptInvitationDto)`	`unknown`	Accept an invitation.

Dependencies

- `PrismaService`
- `EmailService`
- `ConfigService`
- `AuditService` *(optional)*

Where it refuses work

- `InvitationService` stops the work with `NotFoundException` when `!invitation` — “Invitation not found”, in 2 places.
- `InvitationService` stops the work with `ForbiddenException` when `organizationId !== user.organizationId` — “Access denied to this organization”.
- `InvitationService` stops the work with `ForbiddenException` when `!allowedRoles.includes(user.role as UserRole)` — “Insufficient permissions to invite members”.
- `InvitationService` stops the work with `ForbiddenException` when `RANK[dto.role as UserRole] > RANK[user.role as UserRole]` — “You cannot invite a member with a role higher than your own”.
- `InvitationService` stops the work with `ConflictException` when `existingUser && existingUser.organizationId === organizationId` — “User with this email already exists in the organization”.
- `InvitationService` stops the work with `NotFoundException` when `!organization` — “Organization not found”.

Diagram

```
sequenceDiagram
  participant Admin as Organization Admin
  participant API as Invitation Controller
  participant Service as InvitationService
  participant DB as Database
  participant User as Invited User

  Admin->>API: Create invitation
  API->>Service: create(invitation details)
  Service->>DB: Store invitation and token
  DB-->>Service: Invitation record
  Service-->>API: Invitation token/link

  User->>API: Open invitation link
  API->>Service: getByToken(token)
  Service->>DB: Find invitation by token
  DB-->>Service: Invitation details
  Service-->>API: Invitation details

  User->>API: Accept invitation
  API->>Service: accept(token, user)
  Service->>DB: Validate token and add user to organization
```

```
  DB-->>Service: Updated membership/invitation
  Service-->>API: Acceptance result
```

Usage

```
import { Injectable } from '@nestjs/common';
import { InvitationService } from './invitation.service';

@Injectable()
export class OrganizationAdminService {
  constructor(private readonly invitationService: InvitationService) {}

  async inviteMember(organizationId: string, email: string) {
    const invitation = await this.invitationService.create({
      organizationId,
      email,
    });

    return {
      token: invitation.token,
      invitationUrl: `https://app.example.com/invitations/${invitation.token}`,
    };
  }

  async acceptInvitation(token: string, userId: string) {
    // Optionally display invitation details before acceptance.
    await this.invitationService.getByToken(token);

    return this.invitationService.accept(token, userId);
  }
}
```

AI Coding Instructions

- Keep existing-organization membership changes inside the invitation flow; do not add users to an existing organization through open registration.
- Validate invitation tokens through `getByToken()` before presenting invitation details or attempting acceptance.
- Ensure `accept()` is called with the authenticated user identity, not a user ID supplied directly by an untrusted client.
- Preserve token validation rules such as expiration, prior acceptance, revoked invitations, and organization membership checks when modifying the service.
- Keep invitation creation and acceptance transactional where membership updates and invitation state changes must succeed together.

Relationships

- DEPENDS_ON → `PrismaService`
- DEPENDS_ON → `EmailService`
- DEPENDS_ON → `configservice`
- DEPENDS_ON → `AuditService`

Referenced By

- `InvitationController` (DEPENDS_ON)
- `InvitationModule` (MODULE_PROVIDES)
- `InvitationModule` (MODULE_EXPORTS)
- `OrganizationController` (DEPENDS_ON)