PlanService

September 4, 2026

PlanService

Kind: Service

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

Feature-gating by plan (P0-a — the first real feature gate in the codebase; until now PRO/BUSINESS gated nothing but monthly credits).

`featureLevel` answers "what tier is this org on RIGHT NOW" with a short in-memory TTL cache: gates must react to upgrades within a minute, and a downgrade must re-lock features without a data migration — which is why gated READS should also compute effective values through this service rather than persisting gate outcomes.

`PlanService` centralizes plan-based feature gating for organizations. It resolves the organization’s current `PlanLevel` through a short-lived in-memory cache, allowing upgrades to take effect within a minute and downgrades to immediately re-lock gated behavior without persisting feature decisions.

Methods

Method	Signature	Returns	Description
`featureLevel`	`featureLevel(organizationId: string)`	`Promise<PlanLevel>`	The org's EFFECTIVE plan RIGHT NOW (FREE when the org is missing — fail closed).
`meetsPlan`	`meetsPlan(actual: PlanLevel, required: PlanLevel)`	`boolean`	Does `actual` satisfy `required`?
`orgMeetsPlan`	`orgMeetsPlan(organizationId: string, required: PlanLevel)`	`Promise<boolean>`	Convenience: does this org

Method	Signature	Returns	Description
			meet the required tier?
`invalidate`	`invalidate(organizationId: string)`	`void`	Test/ops hook: drop a cached entry (e.g.

Dependencies

- `PrismaService`

Where it refuses work

- `PlanService` stops the work with an early return when `hit && hit.expiresAt > Date.now()`.

Diagram

```
sequenceDiagram
  participant Caller as API Service / Resolver
  participant PlanService
  participant Cache as In-Memory TTL Cache
  participant Billing as Billing Plan Source

  Caller->>PlanService: orgMeetsPlan(orgId, requiredPlan)
  PlanService->>PlanService: featureLevel(orgId)

  PlanService->>Cache: Read cached plan level
  alt Cache hit
    Cache-->>PlanService: PlanLevel
  else Cache miss or expired
    PlanService->>Billing: Resolve current organization plan
    Billing-->>PlanService: PlanLevel
    PlanService->>Cache: Store level with short TTL
  end

  PlanService->>PlanService: meetsPlan(currentLevel, requiredPlan)
  PlanService-->>Caller: true / false

  Note over Caller,PlanService: Call invalidate() after plan changes<br/>when immediate cache r
```

Usage

```ts
import { ForbiddenException } from '@nestjs/common';
import { PlanLevel } from './plan.types';
import { PlanService } from './plan.service';

export class ReportsService {
  constructor(private readonly planService: PlanService) {}

  async getAdvancedReport(orgId: string) {
    const hasAccess = await this.planService.orgMeetsPlan(
      orgId,
      PlanLevel.PRO,
    );

    if (!hasAccess) {
      throw new ForbiddenException(
        'Advanced reports require a Pro or Business plan.',
      );
    }

    return this.buildAdvancedReport(orgId);
  }

  async handlePlanUpdated(orgId: string) {
    // Refresh feature-gate decisions immediately after an upgrade/downgrade.
    this.planService.invalidate(orgId);
  }

  private async buildAdvancedReport(orgId: string) {
    return { orgId, reportType: 'advanced' };
  }
}
```

AI Coding Instructions

- Use `orgMeetsPlan()` for organization-aware authorization checks; use `meetsPlan()` only when a current `PlanLevel` is already available.
- Route both gated writes and gated reads through `PlanService` so downgraded organizations lose access without data migrations.
- Do not persist derived feature-access flags; compute effective access from the organization's current plan at request time.
- Call `invalidate()` after billing subscription, upgrade, downgrade, or cancellation events when access must refresh before the TTL expires.
- Keep new plan comparisons aligned with the existing `PlanLevel` ordering and avoid duplicating tier-comparison logic in controllers or resolvers.

Relationships

- DEPENDS_ON → `PrismaService`

Referenced By

- `BillingModule` (MODULE_PROVIDES)
- `BillingModule` (MODULE_EXPORTS)
- `PlanGuard` (DEPENDS_ON)
- `BrandingService` (DEPENDS_ON)
- `DocAutomationService` (DEPENDS_ON)
- `PlatformService` (DEPENDS_ON)
- `PublicProjectController` (DEPENDS_ON)
- `ProjectAccessService` (DEPENDS_ON)
- `ScimAuthGuard` (DEPENDS_ON)