AiCostRateLimitGuard

September 4, 2026

AiCostRateLimitGuard

Kind: Service

Source: `atloria-monorepo/apps/api/src/technical-docs/guards/ai-cost-rate-limit.guard.ts`

Rate limit for COST-BEARING AI endpoints (/ask, /mcp, /assist, /rag/index, /enrich). Each request spends real money (embedding + gpt-5.6-terra), and /ask + /mcp are reachable
ANONYMOUSLY on public projects — so unlike the per-user AI guard this one:

- keys by userId when authenticated, else by client IP (anonymous still limited)
- never fails open to unlimited: when Redis is unavailable it falls back to a per-instance in-memory window (weaker, but bounded) instead of waving traffic through.

`AiCostRateLimitGuard` is a NestJS guard that rate-limits **cost-bearing AI endpoints** (e.g. `/ask`, `/mcp`, `/assist`, `/rag/index`, `/enrich`) to control real spend from embeddings and LLM calls. It keys limits by `userId` when authenticated, otherwise by **client IP** so anonymous access on public projects is still bounded. If Redis is unavailable, it **does not fail open**—it falls back to a per-instance in-memory window to keep traffic limited.

Methods

Method	Signature	Returns
`canActivate`	`canActivate(context: ExecutionContext)`	`Promise<boolean>`
`onModuleDestroy`	`onModuleDestroy()`	`unknown`

Dependencies

- `ConfigService`

Where it refuses work

- `AiCostRateLimitGuard` stops the work with `HttpException` when `count > this.limit`.

When something fails

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

Diagram

```
sequenceDiagram
  participant Client
  participant Controller
  participant Guard as AiCostRateLimitGuard
  participant Redis
  participant Memory as InMemoryWindow
  participant AI as AI Provider (Embeddings/LLM)

  Client->>Controller: HTTP request to cost-bearing endpoint
  Controller->>Guard: canActivate(context)

  alt Authenticated
    Guard->>Guard: key = userId
  else Anonymous (public project)
    Guard->>Guard: key = client IP
  end

  Guard->>Redis: increment/check window(key)
  alt Redis available + under limit
    Redis-->>Guard: allow
    Guard-->>Controller: proceed
    Controller->>AI: perform embedding/LLM call
    AI-->>Controller: response
    Controller-->>Client: 200 OK
  else Redis available + over limit
    Redis-->>Guard: deny
    Guard-->>Client: 429 Too Many Requests
  else Redis unavailable
    Guard->>Memory: increment/check window(key)
    alt under limit
      Memory-->>Guard: allow
      Guard-->>Controller: proceed
      Controller->>AI: perform embedding/LLM call
      AI-->>Controller: response
      Controller-->>Client: 200 OK
    else over limit
      Memory-->>Guard: deny
      Guard-->>Client: 429 Too Many Requests
```

```
    end
  end
```

Usage

```
import { Controller, Post, UseGuards, Body } from '@nestjs/common';
import { AiCostRateLimitGuard } from '../technical-docs/guards/ai-cost-rate-limit.guard';

@Controller()
export class AskController {
  // Apply guard to a cost-bearing endpoint
  @Post('/ask')
  @UseGuards(AiCostRateLimitGuard)
  async ask(@Body() body: { prompt: string }) {
    // If this runs, the request has passed rate limiting.
    // Perform your embedding + LLM call here.
    return { answer: `You asked: ${body.prompt}` };
  }
}

// Alternatively, apply at the controller level:
// @UseGuards(AiCostRateLimitGuard)
// @Controller('/mcp')
// export class McpController { ... }
```

AI Coding Instructions

- Preserve the **keying strategy**: use `userId` when authenticated; otherwise fall back to **client IP** so anonymous usage is still rate-limited.
- Never change behavior to **fail open** on Redis errors; keep the **in-memory fallback** bounded to prevent unlimited spend during outages.
- Ensure the guard is applied only to **cost-bearing** routes; don’t blanket-apply to non-AI endpoints unless you intend to throttle them too.
- When integrating behind proxies/CDNs, confirm the **true client IP extraction** is correct (e.g., trusted proxy headers) to avoid mis-keying or over-throttling.

Relationships

- DEPENDS_ON → `configservice`

Referenced By

- `TechnicalDocsModule` (MODULE_PROVIDES)