ChatRateLimitGuard

September 4, 2026

ChatRateLimitGuard

Kind: Service

Source: `atloria-monorepo/apps/api/src/support-agent/chat-rate-limit.guard.ts`

Phase 7: In-memory rate limiter for public chat endpoints.

Three independent limits:

- Per IP: 10 messages / minute
- Per project: 1000 messages / hour
- Per session: 50 messages total (TTL: 24 h)

`ChatRateLimitGuard` is a NestJS guard that enforces in-memory rate limits for public chat endpoints to prevent abuse and protect backend resources. It applies three independent limits—per IP (10/min), per project (1000/hour), and per session (50 total with a 24h TTL)—and blocks requests that exceed any threshold.

Methods

Method	Signature	Returns
`canActivate`	`canActivate(context: ExecutionContext)`	`boolean`

Where it refuses work

- `ChatRateLimitGuard` stops the work with `HttpException` when `entry.count >= max`.

Diagram

```
sequenceDiagram
  autonumber
  participant C as Client
  participant G as ChatRateLimitGuard
  participant R as In-memory Rate Store
```

```
  participant H as Controller/Handler

  C->>G: HTTP request (public chat endpoint)
  G->>R: Increment/check counters (IP, project, session)
  alt Any limit exceeded
    R-->>G: Reject (rate limit hit)
    G-->>C: 429 Too Many Requests
  else Within limits
    R-->>G: Allow
    G->>H: canActivate() passes
    H-->>C: Normal response
  end
```

Usage

```
import { Controller, Post, UseGuards, Body } from '@nestjs/common';
import { ChatRateLimitGuard } from './chat-rate-limit.guard';

@Controller('public/chat')
@UseGuards(ChatRateLimitGuard)
export class PublicChatController {
  @Post('message')
  async sendMessage(
    @Body() body: { projectId: string; sessionId: string; message: string },
  ) {
    // If any rate limit is exceeded, the guard will block before reaching here.
    return { ok: true };
  }
}
```

AI Coding Instructions

- Keep limits independent: evaluate IP, project, and session constraints separately and fail fast if any one is exceeded.
- Ensure stable identifiers: extract IP (respecting proxy setup), projectId, and sessionId consistently; mismatched keys will silently weaken or over-tighten limits.
- Remember this is in-memory: it resets on process restart and does not coordinate across multiple instances; avoid assuming global enforcement in distributed deployments.
- Session limit is total with TTL: do not accidentally implement it as “per window” like the IP/project limits; preserve the 24h TTL semantics.
- When integrating new endpoints, apply the guard only to public-facing chat routes and ensure required identifiers (projectId/sessionId) are present before the guard

runs.