ReaderLogRetentionService

September 4, 2026

ReaderLogRetentionService

Kind: Service

Source: `atloria-monorepo/apps/api/src/reader-account/reader-log-retention.service.ts`

C3 log-retention sweep: ApiCallLog is per-reader attributed (PII-adjacent: reader ids + concrete request paths), so rows are hard-deleted after API_CALL_LOG_RETENTION_DAYS (default 90, floor 7). Applies to ALL sources (playground / public-mcp / owner-mcp) — the C4 analytics windows are 30d, so a 90d floor keeps every consumer whole.

Multi-replica single-fire mirrors ScheduledPublishService: a dedicated lazy ioredis client + SET NX PX lock that is never released (TTL expiry is the release), fail-closed on Redis errors (skip, next day retries).

`ReaderLogRetentionService` performs a daily retention sweep for `ApiCallLog` records, permanently deleting logs older than `API_CALL_LOG_RETENTION_DAYS` (default: 90 days; minimum: 7 days). It applies uniformly to playground, public MCP, and owner MCP traffic, using a Redis NX/PX lock so only one API replica runs the sweep while Redis failures safely defer cleanup until the next scheduled attempt.

Methods

Method	Signature	Returns	Description
`sweep`	`sweep()`	`Promise<void>`	
`deleteExpired`	`deleteExpired(now: Date)`	`Promise<number>`	Exposed for tests + manual ops runs.
`retentionDays`	`retentionDays()`	`number`	
`onModuleDestroy`	`onModuleDestroy()`	`void`	

Dependencies

- `ConfigService`
- `PrismaService`

Where it refuses work

- `ReaderLogRetentionService` stops the work with an early return when `process.env.API_CALL_LOG_RETENTION_DISABLED === 'true'`.
- `ReaderLogRetentionService` stops the work with an early return when `!(await this.acquireLock())`.
- `ReaderLogRetentionService` stops the work with an early return when `!Number.isFinite(raw) || raw <= 0`.

When something fails

- `ReaderLogRetentionService` handles failure in 2 places: it logs it and continues in 1, and turns it into a return value in 1.

Diagram

```
sequenceDiagram
  participant Scheduler as Nest Scheduler
  participant Service as ReaderLogRetentionService
  participant Redis as Redis
  participant DB as Database

  Scheduler->>Service: sweep()
  Service->>Redis: SET retention-lock token NX PX ttl

  alt Lock acquired
    Redis-->>Service: OK
    Service->>Service: retentionDays()
    Service->>DB: Delete ApiCallLog rows older than cutoff
    DB-->>Service: Deleted row count
    Service-->>Scheduler: Sweep complete
  else Lock unavailable or Redis error
    Redis-->>Service: Lock denied / error
    Service-->>Scheduler: Skip sweep; retry next day
  end

  Note over Service,Redis: Lock is never explicitly released;<br/>PX TTL provides release.
```

Usage

```
import { Injectable } from '@nestjs/common';
import { ReaderLogRetentionService } from './reader-log-retention.service';

@Injectable()
export class MaintenanceService {
  constructor(
    private readonly readerLogRetentionService: ReaderLogRetentionService,
  ) {}

  async runReaderLogCleanup(): Promise<void> {
    // Normally invoked by the service's scheduled job.
    // Calling sweep() preserves the distributed-lock behavior.
    await this.readerLogRetentionService.sweep();
  }

  async previewRetentionConfiguration(): Promise<void> {
    const days = this.readerLogRetentionService.retentionDays();

    console.log(`Reader API logs are retained for ${days} days.`);
  }
}
```

AI Coding Instructions

- Call `sweep()` for scheduled cleanup flows; do not call `deleteExpired()` directly unless intentionally bypassing the distributed-lock orchestration.
- Preserve the Redis `SET NX PX` single-fire pattern across replicas, and do not manually release the lock—the TTL is the lock release mechanism.
- Treat Redis failures as fail-closed: skip deletion when Redis is unavailable so another scheduled run can retry safely.
- Keep `API_CALL_LOG_RETENTION_DAYS` at or above the enforced 7-day floor; the default 90-day period supports downstream 30-day analytics windows.
- Ensure retention changes continue to apply to every `ApiCallLog` source, including playground, public MCP, and owner MCP traffic.

Relationships

- DEPENDS_ON → `configservice`
- DEPENDS_ON → `PrismaService`

Referenced By

- `ReaderAccountModule` (MODULE_PROVIDES)