ReaderTokenService

September 4, 2026

ReaderTokenService

Kind: Service

Source: `atloria-monorepo/apps/api/src/reader-access/reader-token.service.ts`

ReaderTokenService — mint/verify the shared reader token (docs-reader aud).

Signed with the API JWT secret but a SEPARATE audience so a reader token can never satisfy a member route (JwtAuthGuard) and vice-versa: the audience option makes verifyAsync reject a member JWT (which carries no `docs-reader` aud) and makes JwtStrategy reject this one (its exp/sub differ and it has an aud the member path never sets). Revocation is short expiry + `ver` bump — there is no deny-list.

`ReaderTokenService` mints and verifies short-lived JWTs used for shared documentation reader access. Tokens are signed with the API JWT secret but use the dedicated `docs-reader` audience, ensuring they cannot authenticate member-only routes; revocation is handled through token expiry and a version (`ver`) bump rather than a deny-list.

Methods

Method	Signature	Returns	Description
`mint`	`mint(input: MintReaderTokenInput)`	`Promise<string>`	
`verify`	`verify(token: string)`	`Promise<ReaderClaims | null>`	
`mintUnion`	`mintUnion(existingToken: string | undefined, base: MintReaderTokenInput)`	`Promise<string>`	
`currentTokenVersion`	`currentTokenVersion(projectId: string)`	`Promise<number>`	The project's current tokenVersion (default 1 when no policy row exists).

Dependencies

- `JwtService`
- `PrismaService`

Where it refuses work

- `ReaderTokenService` stops the work with `BadRequestException` when `capTtl <= 0` — “This access grant has already expired.”.
- `ReaderTokenService` stops the work with an early return when `!token || typeof token !== 'string'`.
- `ReaderTokenService` stops the work with an early return when `!claims?.projectId`.

When something fails

- `ReaderTokenService` handles failure in 1 place: it turns it into a return value in all 1.

Diagram

```
sequenceDiagram
    participant Client
    participant ReaderTokenService
    participant JWT as JwtService
    participant Config as Token Version Store
    participant ReaderRoute

    Client->>ReaderTokenService: mint() / mintUnion()
    ReaderTokenService->>Config: currentTokenVersion()
    Config-->>ReaderTokenService: ver
    ReaderTokenService->>JWT: signAsync(claims, audience: "docs-reader")
    JWT-->>ReaderTokenService: signed reader token
    ReaderTokenService-->>Client: token

    Client->>ReaderRoute: Request with reader token
    ReaderRoute->>ReaderTokenService: verify(token)
    ReaderTokenService->>JWT: verifyAsync(token, audience: "docs-reader")
    JWT-->>ReaderTokenService: ReaderClaims
    ReaderTokenService->>Config: Compare claims.ver with current version
    Config-->>ReaderTokenService: current ver
    ReaderTokenService-->>ReaderRoute: ReaderClaims or null
```

Usage

```
import { Injectable, UnauthorizedException } from '@nestjs/common';
import { ReaderTokenService } from './reader-token.service';
```

```
@Injectable()
export class ReaderAccessService {
  constructor(private readonly readerTokenService: ReaderTokenService) {}

  async createSharedReaderToken(): Promise<{ token: string }> {
    const token = await this.readerTokenService.mint();

    return { token };
  }

  async validateSharedReaderToken(token: string) {
    const claims = await this.readerTokenService.verify(token);

    if (!claims) {
      throw new UnauthorizedException('Invalid or expired reader token');
    }

    return claims;
  }

  async revokeExistingReaderTokens(): Promise<number> {
    // Update the backing token-version value through the owning configuration
    // or settings service. Tokens minted with an older `ver` will fail verify().
    return this.readerTokenService.currentTokenVersion();
  }
}
```

AI Coding Instructions

- Always use the `docs-reader` audience when minting or verifying reader tokens; do not reuse member JWT validation settings.
- Use `mint()` for standard shared-reader access and `mintUnion()` only where union-specific reader claims are required.
- Treat a `null` result from `verify()` as an invalid, expired, incorrectly-audienced, or revoked token and deny access.
- Do not add a token deny-list unless requirements explicitly change; revocation is intentionally implemented with short expiration and the `ver` claim.
- Keep reader-token authentication isolated from `JwtAuthGuard` and member JWT strategies so neither token type can satisfy the other's routes.

Relationships

- DEPENDS_ON → `jwtservice`
- DEPENDS_ON → `PrismaService`

Referenced By

- `SamlController` (DEPENDS_ON)
- `ContentVisibilityService` (DEPENDS_ON)
- `DocsAccessService` (DEPENDS_ON)
- `ReaderAccessModule` (MODULE_PROVIDES)
- `ReaderAccessModule` (MODULE_EXPORTS)
- `ReaderClaimsExchangeService` (DEPENDS_ON)
- `ReaderMagicLinkService` (DEPENDS_ON)
- `ToursService` (DEPENDS_ON)