SsoCodeService

September 4, 2026

SsoCodeService

Kind: Service

Source: `atloria-monorepo/apps/api/src/auth/saml/sso-code.service.ts`

One-time SSO handoff codes (A1).

SECURITY: a SAML assertion is validated on the API, but the browser learns the result only by exchanging a short-lived, single-use code — we NEVER put a JWT (or the refresh token) in a redirect URL, where it would leak via history, referrer headers and server logs. The already-minted token pair is stored in Redis under 32 bytes of CSPRNG entropy with a 60s TTL, and the exchange deletes the key atomically (GETDEL) so a code can be redeemed exactly once.

`SsoCodeService` provides short-lived, single-use handoff codes for completing SAML SSO authentication safely. After the API validates a SAML assertion and mints an authentication token pair, this service stores the result in Redis and returns an opaque code that the browser can exchange once—preventing JWTs or refresh tokens from appearing in redirect URLs, history, referrer headers, or logs.

Methods

Method	Signature	Returns	Description
`mint`	`mint(payload: AuthResponseDto)`	`Promise<string>`	Store an authenticated token pair under a fresh code and return the code.
`exchange`	`exchange(code: string)`	`Promise<AuthResponseDto | null>`	

Dependencies

- `RedisService`

Where it refuses work

- `SsoCodeService` stops the work with `Error` when `!ok` — “Could not persist SSO handoff code (Redis unavailable).”.
- `SsoCodeService` stops the work with an early return when `!code || typeof code !== 'string'`.
- `SsoCodeService` stops the work with an early return when `!raw`.

When something fails

- `SsoCodeService` handles failure in 2 places: it turns it into a return value in all 2.

Diagram

```
sequenceDiagram
    participant Browser
    participant SAML as SAML Callback Controller
    participant SsoCodeService
    participant Redis
    participant Auth as Auth Controller

    SAML->>SsoCodeService: mint(authResponse)
    SsoCodeService->>Redis: SET code -> auth response (TTL: 60s)
    Redis-->>SsoCodeService: OK
    SsoCodeService-->>SAML: opaque one-time code
    SAML-->>Browser: Redirect with ?code=...

    Browser->>Auth: POST /auth/sso/exchange { code }
    Auth->>SsoCodeService: exchange(code)
    SsoCodeService->>Redis: GETDEL code
    Redis-->>SsoCodeService: stored auth response or null
    SsoCodeService-->>Auth: AuthResponseDto | null
    Auth-->>Browser: Token response or invalid-code error
```

Usage

```
import { Controller, Get, Post, Body, Res } from '@nestjs/common';
import type { Response } from 'express';
import { SsoCodeService } from './sso-code.service';
import { AuthResponseDto } from '../dto/auth-response.dto';

@Controller('auth/sso')
export class SsoController {
  constructor(private readonly ssoCodeService: SsoCodeService) {}
```

```
  @Get('callback')
  async samlCallback(@Res() response: Response) {
    // The SAML assertion has already been validated and tokens minted.
    const authResponse: AuthResponseDto = {
      accessToken: 'minted-access-token',
      refreshToken: 'minted-refresh-token',
    };

    const code = await this.ssoCodeService.mint(authResponse);

    // Redirect only with the opaque, short-lived code.
    return response.redirect(
      `${process.env.WEB_APP_URL}/sso/complete?code=${encodeURIComponent(code)}`,
    );
  }

  @Post('exchange')
  async exchangeCode(@Body('code') code: string) {
    const authResponse = await this.ssoCodeService.exchange(code);

    if (!authResponse) {
      return { error: 'SSO code is invalid, expired, or already redeemed.' };
    }

    return authResponse;
  }
}
```

AI Coding Instructions

- Keep access and refresh tokens out of redirect URLs; redirect only with the opaque code returned by `mint()`.
- Treat SSO codes as secrets: generate them with cryptographically secure randomness, avoid logging them, and preserve the short TTL.
- Use Redis atomic `GETDEL` semantics during `exchange()` so concurrent requests cannot redeem the same code twice.
- Handle a `null` exchange result as an invalid, expired, or already-used code; do not distinguish these cases to clients.
- Ensure the SAML callback validates the assertion and mints the token pair before calling `mint()`, while the frontend exchanges the code immediately after redirect.

Relationships

- DEPENDS_ON → `RedisService`

Referenced By

- `SamlController` (DEPENDS_ON)
- `SamlModule` (MODULE_PROVIDES)
- `SsoController` (DEPENDS_ON)