SsoController

September 4, 2026

SsoController

Kind: Controller

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

SSO handoff + discovery (A1). Public by design:

- POST /auth/sso/exchange redeems a one-time code (minted after a validated SAML assertion) for the standard token pair — single use, so a leaked URL can't be replayed.
- GET /auth/sso/discover maps an email domain to an SSO-enabled org so the login page can offer "Continue with SSO" from just an email.

`SsoController` exposes public SSO handoff endpoints for the authentication flow. It redeems validated, one-time SAML exchange codes for the standard token pair and discovers whether an email domain belongs to an SSO-enabled organization so clients can offer a “Continue with SSO” option.

Diagram

```
graph LR
  Client[Login client] -->|GET /auth/sso/discover?email=user@company.com| SsoController
  SsoController --> DiscoveryService[SSO discovery service]
  DiscoveryService --> Organization[Organization SSO configuration]
  Organization --> DiscoveryService
  DiscoveryService --> Client

  SamlCallback[Validated SAML assertion] -->|Mints one-time code| ExchangeCode[One-time SSO cod
  Client -->|POST /auth/sso/exchange| SsoController
  SsoController --> ExchangeService[SSO exchange service]
  ExchangeService -->|Consumes code once| ExchangeCode
  ExchangeService --> TokenPair[Access and refresh tokens]
  TokenPair --> Client
```

Usage

```
// Discover whether the user's organization supports SSO.
const discoveryResponse = await fetch(
  `/auth/sso/discover?email=${encodeURIComponent("user@acme.com")}`,
);

const discovery = await discoveryResponse.json();

if (discovery.ssoEnabled) {
  // Redirect or present the organization's SSO login option.
  window.location.assign(discovery.loginUrl);
}

// After the SAML callback redirects the client with a one-time code,
// exchange it immediately for the standard application token pair.
const exchangeResponse = await fetch("/auth/sso/exchange", {
  method: "POST",
  headers: {
    "Content-Type": "application/json",
  },
  body: JSON.stringify({
    code: new URLSearchParams(window.location.search).get("code"),
  }),
});

if (!exchangeResponse.ok) {
  throw new Error("SSO code exchange failed or was already used.");
}

const { accessToken, refreshToken } = await exchangeResponse.json();
```

AI Coding Instructions

- Keep both endpoints public: SSO discovery and code redemption must be reachable before a user has an application session.
- Treat exchange codes as short-lived, single-use credentials; consume them atomically and never allow a successful code to be redeemed twice.
- Do not issue tokens directly from unvalidated SAML data in this controller; rely on the SAML validation and exchange services for assertion verification and token minting.
- Preserve the standard authentication token response shape so SSO clients integrate identically to password or other login methods.
- Avoid leaking organization or SSO configuration details during discovery beyond what the login client needs to offer the SSO flow.

Relationships

- MODULE_DECLARES → `exchange`
- MODULE_DECLARES → `discover`
- DEPENDS_ON → `SamlService`
- DEPENDS_ON → `SsoCodeService`

Referenced By

- `SamlModule` (MODULE_DECLARES)