**🔍 Your Architecture**

| **Component** | **URL / Domain** |
| --- | --- |
| App1 (OIDC) | https://healthgateway1.bc.ca |
| App2 (SAML) | https://accessmyhealth.vch.ca |
| Keycloak | https://bcportalsauth.bc.ca (realm: shared realm) |

**🎬 Expected SSO Behavior**

**1️⃣ App1: User logs in (OIDC flow)**

* User visits https://healthgateway1.bc.ca
* App1 redirects to:

https://bcportalsauth.bc.ca/realms/myrealm/protocol/openid-connect/auth

* User logs in → Keycloak:
  + Sets **KEYCLOAK\_SESSION** cookie:
* Set-Cookie: KEYCLOAK\_SESSION=abc123;
* Domain=bcportalsauth.bc.ca;
* Path=/;
* Secure;
* SameSite=None;
* User redirected back to App1 → App1 establishes its own local session.

**2️⃣ User later goes to App2 (SAML flow)**

* User visits https://accessmyhealth.vch.ca
* App2 sends **SAML AuthnRequest** to:

https://bcportalsauth.bc.ca/realms/myrealm/protocol/saml

* Browser → **will send KEYCLOAK\_SESSION cookie** to bcportalsauth.bc.ca!
* Keycloak sees active SSO session → **no login screen** → sends SAMLResponse to App2 → App2 establishes its own session.

**🚩 Key: Why SSO Works Here**

| **Factor** | **Why it works** |
| --- | --- |
| Different app domains (bc.ca, vch.ca) | OK — apps do *not* share sessions — that’s expected. |
| Keycloak domain is bcportalsauth.bc.ca | Browser will send KEYCLOAK\_SESSION cookie when redirected to Keycloak. |
| SSO happens **at Keycloak** | Not between apps → each app talks to Keycloak → Keycloak enforces SSO. |
| Cookie domain is correct | Must be Domain=bcportalsauth.bc.ca (not bc.ca or .ca — must be exact FQDN). |
| SameSite=None; Secure required | Needed because **cross-domain POST redirects** are used in SAML flow, and modern browsers block cookies otherwise. |

**🛠 Specific Tuning Needed**

| **Setting** | **Why** |
| --- | --- |
| KEYCLOAK\_SESSION cookie → SameSite=None; Secure | Required because App1 and App2 are cross-domain. |
| SSO Session Idle / Max | Tune this for how long SSO should last — e.g. 2 hours idle, 8 hours max. |
| SAML SP binding | Use **POST** binding (more reliable than Redirect) for App2. |
| App1 OIDC client settings | Standard — nothing special. |
| App2 SAML client settings | Ensure correct ACS URL and binding. |

**🚩 How KEYCLOAK\_SESSION cookie behaves:**

| **Request** | **Does browser send KEYCLOAK\_SESSION?** |
| --- | --- |
| App1 → App1 | ❌ No — App1’s own domain, no Keycloak cookies sent. |
| App2 → App2 | ❌ No — App2’s own domain. |
| App1 → redirect to Keycloak → Keycloak | ✅ Yes — KEYCLOAK\_SESSION sent → SSO works. |
| App2 → redirect to Keycloak → Keycloak | ✅ Yes — KEYCLOAK\_SESSION sent → SSO works. |

**🚩 Common Pitfall: SameSite Cookie**

If Keycloak’s **KEYCLOAK\_SESSION** cookie does not have:

SameSite=None; Secure

→ Browser will block it in cross-domain redirect → SSO fails → login page shown again at App2.

**Final Summary for Your Case**

✅ **OIDC App1 + SAML App2 → SSO will work perfectly**, even cross-domain, because:

* Same Keycloak realm.
* Browser sends KEYCLOAK\_SESSION to Keycloak regardless of app domain.
* Keycloak does *protocol-agnostic SSO*: OIDC + SAML → no problem.

**🚩 Checklist for Success (Actionable for your team)**

| **Checklist Item** | **Status** |
| --- | --- |
| Keycloak **KEYCLOAK\_SESSION** cookie: SameSite=None; Secure | MUST have — check Keycloak version & reverse proxy config (Nginx, Apache) |
| Keycloak realm → same realm used by App1 and App2 | ✅ Yes |
| App1 OIDC flow → works already | ✅ Yes — sounds like in place |
| App2 SAML SP → configured with POST binding | Strongly recommended |
| Tune SSO session timeout | Configure in Keycloak realm settings → SSO Session Idle / Max settings |
| Optional → add "SSO test page" in App2 to verify silent SAML login | Useful for QA testing |

**🚀 Summary**

👉 In your exact architecture:

https://healthgateway1.bc.ca (OIDC app)

https://accessmyhealth.vch.ca (SAML app)

https://bcportalsauth.bc.ca (Keycloak)

✅ **Cross-protocol SSO will work**  
✅ **Cross-domain SSO will work**  
✅ **Browser sends SSO cookie to Keycloak** → no login screen needed when user switches app.