**OIDC app1**, **SAML app2**, **cross-domain**, same Keycloak realm:

**🎬 Scenario: App1 (OIDC), App2 (SAML), cross-domain**

**Environment:**

* App1:
  + OIDC client → https://app1.example.com
* App2:
  + SAML SP → https://app2.example.org
* Both use:
  + Same **Keycloak realm** → https://keycloak.example.net

**1️⃣ User logs into App1 (OIDC flow)**

**A. User accesses App1 → no session → App1 redirects:**

GET https://keycloak.example.net/realms/myrealm/protocol/openid-connect/auth

?client\_id=app1-client

&redirect\_uri=https://app1.example.com/callback

&response\_type=code

&scope=openid profile

**B. At Keycloak:**

* No existing SSO session yet → shows **login page**.
* User logs in → Keycloak:
  + Creates **Keycloak SSO session** (server-side session).
  + Issues **KEYCLOAK\_SESSION** cookie to browser:
  + Set-Cookie: KEYCLOAK\_SESSION=abc123; Domain=keycloak.example.net; Path=/; Secure; HttpOnly

**C. Back to App1:**

* App1 exchanges code → gets tokens → sets its **own session** (usually its own domain-scoped cookie).

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

**A. User accesses App2 → App2 redirects to Keycloak:**

* SAML **AuthnRequest** → Redirect or POST to:

https://keycloak.example.net/realms/myrealm/protocol/saml

**B. At Keycloak — SSO magic**

* Browser sends **KEYCLOAK\_SESSION** cookie (domain = keycloak.example.net).
* Even though App1 and App2 are cross-domain, both use same Keycloak — so browser still sends the SSO cookie!
* Keycloak sees: "User already has an SSO session."

👉 **No login screen** → immediately issues **SAMLResponse**.

**C. Back to App2:**

* Keycloak POSTs back to App2:

<form action="https://app2.example.org/saml/acs" method="POST">

<input type="hidden" name="SAMLResponse" value="...base64 SAML assertion...">

</form>

* App2 establishes its own local session.

**🔍 Visual Flow**

User → App1 → Keycloak (OIDC login) → App1

User → App2 → Keycloak (SAML SSO used) → App2

**🚩 Key Considerations (cross-protocol SSO)**

| **Point** | **Status** |
| --- | --- |
| Same Keycloak realm | ✅ Required (so SSO session is shared) |
| Browser sends KEYCLOAK\_SESSION | ✅ Yes — because both use same Keycloak domain |
| OIDC → SAML SSO? | ✅ Works — Keycloak treats both as relying parties of same SSO session |
| App1 local session vs App2 local session | ❌ Not shared → each app builds its own session after receiving tokens/assertions |
| Cross-domain | ✅ Works fine — because KEYCLOAK\_SESSION is for Keycloak's domain only, not app domains |

**🛠 Keycloak Config Notes**

| **Setting** | **Recommendation** |
| --- | --- |
| SSO Session Idle | Tune this → controls how long SSO session remains valid |
| SSO Session Max | Tune this → max absolute SSO lifetime |
| Client Session Idle | Tune per-client (OIDC / SAML) if needed |
| Client Session Max | Tune per-client (OIDC / SAML) if needed |
| SAML SP | Use POST binding → works more robustly across domains |
| KEYCLOAK\_SESSION Cookie | Make sure not blocked by SameSite restrictions (usually SameSite=None; Secure) |

**Final Summary**

✅ **YES** — this works:

1️⃣ User logs in to App1 (OIDC) → Keycloak SSO session created.

2️⃣ User goes to App2 (SAML) → Keycloak recognizes SSO → no login required → SAMLResponse issued.

**Limitations / Watch Outs**

| **Issue** | **Detail** |
| --- | --- |
| Browser blocks cookies (SameSite) | Make sure KEYCLOAK\_SESSION is SameSite=None; Secure to allow cross-site SSO redirects. |
| Different Keycloak realms | Then **no SSO** — SSO session is realm-specific. |
| Different IdPs (OIDC vs SAML brokered) | Complex if you mix different IdPs — but in your case, same realm, works fine. |

**TL;DR**

👉 **OIDC app + SAML app + cross-domain** — works perfectly **IF:**

✅ same Keycloak realm  
✅ browser sends KEYCLOAK\_SESSION cookie  
✅ you tune SSO session timeouts carefully  
✅ you handle cross-domain cookies properly