Exposing Groups Through Shibboleth

Institutions may want to release group information to Shibboleth Service Providers in a secure way when a user is accessing a site. Here are some ways to do that. You may also want to see the page on Grouper and Shibboleth Integration.

Sending the isMemberOf attribute

Using LDAP's isMemberOf attribute as a source

If you have users' group memberships being sync'ed to user objects in LDAP in an attribute such as isMemberOf and you also have a Shibboleth Identity Provider data connector for your LDAP, then you can very easily and securely release specific group memberships to each Service Provider using attribute filter policies. Your data connector for LDAP and your attribute definition for isMemberOf may look like the following:

```
<resolver:DataConnector id="myLDAP" xsi:type="LDAPDirectory" xmlns="urn:mace:shibboleth:2.0:resolver:dc"
    ldapURL="ldaps://ldap.example.org" baseDN="dc=example,dc=org" principal="cn=directory manager"
    principalCredential="password" poolInitialSize="3" poolMaxIdleSize="3">
    <FilterTemplate>
        <![CDATA[
            (uid=$requestContext.principalName)
        ]]>
    </FilterTemplate>
</resolver:DataConnector>

<resolver:AttributeDefinition id="isMemberOf" xsi:type="Simple" xmlns="urn:mace:shibboleth:2.0:resolver:ad"
    sourceAttributeID="isMemberOf">
    <resolver:Dependency ref="myLDAP" />

    <resolver:AttributeEncoder xsi:type="SAML1String" xmlns="urn:mace:shibboleth:2.0:attribute:encoder"
        name="urn:oid:1.3.6.1.4.1.5923.1.5.1.1" />

    <resolver:AttributeEncoder xsi:type="SAML2String" xmlns="urn:mace:shibboleth:2.0:attribute:encoder"
        name="urn:oid:1.3.6.1.4.1.5923.1.5.1.1" friendlyName="isMemberOf" />
</resolver:AttributeDefinition>
```

Using LDAP's member attribute as a source

If, instead, you have users' group memberships being sync'ed to group objects in LDAP, where the group's name is the **cn** attribute, you can still easily and securely release specific group memberships to each Service Provider using attribute filter policies. Your data connector for LDAP and your attribute definition for isMemberOf may look like the following:

```
    <resolver:DataConnector id="myLDAP" xsi:type="LDAPDirectory" xmlns="urn:mace:shibboleth:2.0:resolver:dc"
        ldapURL="ldaps://ldap.example.org" baseDN="dc=example,dc=org" principal="cn=directory manager"
        principalCredential="password" poolInitialSize="3" poolMaxIdleSize="3"
        maxResultSize="500" mergeResults="true" >
        <FilterTemplate>
            <![CDATA[
                (member=uid=${requestContext.principalName})
            ]]>
        </FilterTemplate>
        <ReturnAttributes>cn</ReturnAttributes>
    </resolver:DataConnector>

    <resolver:AttributeDefinition id="isMemberOf" xsi:type="Simple" xmlns="urn:mace:shibboleth:2.0:resolver:ad"
        sourceAttributeID="cn">
        <resolver:Dependency ref="myLDAP" />

        <resolver:AttributeEncoder xsi:type="SAML1String" xmlns="urn:mace:shibboleth:2.0:attribute:encoder"
            name="urn:oid:1.3.6.1.4.1.5923.1.5.1.1" />

        <resolver:AttributeEncoder xsi:type="SAML2String" xmlns="urn:mace:shibboleth:2.0:attribute:encoder"
            name="urn:oid:1.3.6.1.4.1.5923.1.5.1.1" friendlyName="isMemberOf" />
    </resolver:AttributeDefinition>
```

Filtering the attribute

So, for instance, say if you have a group called institution:dept:allMembers. And say if you want to release this group (but no other groups) to sp.example.org for users that are a member of this group. Your attribute filter policy may look like the following. The PermitValueRule is used to restrict the values that are sent to the Service Provider.

```
        <AttributeFilterPolicy id="isMemberOfRuleExample">
                <PolicyRequirementRule xsi:type="basic:AttributeRequesterString" value="https://sp.example.org
/shibboleth" />

                <AttributeRule attributeID="isMemberOf">
                        <PermitValueRule xsi:type="basic:AttributeValueString" value="institution:dept:
allMembers" />
                </AttributeRule>
        </AttributeFilterPolicy>
```

Another use case may be that you allow a department to create adhoc groups within the folder institution:dept:adhoc and you want to release all those adhoc groups (but no other groups) to sp.example.org. Your attribute filter policy may look like the following:

```
        <AttributeFilterPolicy id="isMemberOfRuleExample2">
                <PolicyRequirementRule xsi:type="basic:AttributeRequesterString" value="https://sp.example.org
/shibboleth" />

                <AttributeRule attributeID="isMemberOf">
                        <PermitValueRule xsi:type="basic:AttributeValueRegex" regex="institution:dept:adhoc:.*"
/>
                </AttributeRule>
        </AttributeFilterPolicy>
```

The difference here is that the PermitValueRule is based on a regular expression match rather than an exact string match.

Sending the entitlement attribute

The isMemberOf attribute requires that an SP have intimate knowledge of the group names used by each of its IdP partners. It seems unlikely that all the partners will conveniently choose exactly the same names for groups of similar purpose. In addition, the SP has to know, for each IdP, what membership in one or more groups implies. Thus an SP working with many IdPs will often ask for an consistent entitlement, which may or may not be derived from group membership at a particular IdP.

Converting isMemberOf to an entitlement

If you have a data connector producing isMemberOf attributes (or cn), as described above, your attribute definition for entitlement may look like the following:

Suppose you want to release the standard library entitlement, based on membership in one or more groups.

```
<resolver:AttributeDefinition id="memberships" xsi:type="Simple"
    xmlns="urn:mace:shibboleth:2.0:resolver:ad"
    sourceAttributeID="isMemberOf">
    <resolver:Dependency ref="myLDAP" />
</resolver:AttributeDefinition>

<resolver:AttributeDefinition id="entitlement_lib" xsi:type="Script"
    xmlns="urn:mace:shibboleth:2.0:resolver:ad">
    <resolver:Dependency ref="memberships" />

   <resolver:AttributeEncoder xsi:type="SAML1String" xmlns="urn:mace:shibboleth:2.0:attribute:encoder"
        name="urn:mace:dir:attribute-def:eduPersonEntitlement" />

    <resolver:AttributeEncoder xsi:type="SAML2String" xmlns="urn:mace:shibboleth:2.0:attribute:encoder"
        name="urn:oid:1.3.6.1.4.1.5923.1.1.1.7" friendlyName="eduPersonEntitlement" />

  <Script>
    <![CDATA[
       importPackage(Packages.edu.internet2.middleware.shibboleth.common.attribute.provider);
       entitlement = new BasicAttribute("entitlement_lib");
       var ngroup = memberships.getValues().size();
       for (var i=0; i<ngroup; i++) {
         var group = memberships.getValues().get(i);
           if (group.equals("uw:student") || group.equals("uw:employee") || group.equals("uw:lib:users") ) {
              entitlement.getValues().add('urn:mace:dir:entitlement:common-lib-terms');
              break;
           }
       }
      ]]>
   </Script>

</resolver:AttributeDefinition>
```

Your filters for this attribute would look similar to those for the raw isMemberOf attribute filters.