ResolverScriptAttributeDefinitionExamples

Script Attribute Definition Examples

The following examples are simply that, examples. They do not illustrate all possible configuration properties or features. Refer to script attribute definition for this information.

Generate Unique Opaque Identifier

Contributed by: Lukas Haemmerle, SWITCH, Switzerland

This example generates an unique, opaque, identifier based off of an identifier from another attribute. This is approach is useful when an opaque identifier is necessary but is not available in the user data store.

```
<resolver:AttributeDefinition xsi:type="Script" xmlns="urn:mace:shibboleth:2.0:resolver:ad"
                              id="swissEduPersonUniqueID"
                              sourceAttributeID="uidNumber">

    <!-- Dependency that provides the source attribute. -->
    <resolver:Dependency ref="myLDAP" />
    <resolver:Dependency ref="uidNumber" />

    <!-- SAML 1 and 2 encoders for the attribute. -->
    <resolver:AttributeEncoder xsi:type="SAML1String" xmlns="urn:mace:shibboleth:2.0:attribute:encoder"
                               name="urn:mace:switch.ch:attribute-def:swissEduPersonUniqueID" />
    <resolver:AttributeEncoder xsi:type="SAML2String" xmlns="urn:mace:shibboleth:2.0:attribute:encoder"
                               name="urn:oid:2.16.756.1.2.5.1.1.1"
                               friendlyName="swissEduPersonUniqueID" />

    <!-- The script, wrapped in a CDATA section so that special XML characters don't need to be removed -->
    <Script><![CDATA[
        // Import Shibboleth attribute provider
        importPackage(Packages.edu.internet2.middleware.shibboleth.common.attribute.provider);

        // Import Apache commons codecs
        importPackage(Packages.org.apache.commons.codec.digest);

        // Get the unique value
        uniqueValue = uidNumber.getValues().get(0) + "some#salt#value#12345679";

        // Create md5 value
        localpart = DigestUtils.md5Hex(uniqueValue);

        // Get attribute to add iff we have to
                 if (null == swissEduPersonUniqueID) {
                swissEduPersonUniqueID = new BasicAttribute("swissEduPersonUniqueID");
                }

        // Prepend unique and pseudo-random localpart to domain name
        swissEduPersonUniqueID.getValues().add(localpart + "@switch.ch");
    ]]></Script>
</resolver:AttributeDefinition>
```

Generate Affiliation based on Groups

Contributed by: Halm Reusser, SWITCH, Switzerland

This example demonstrates the generation of values for the `eduPersonAffiliation` attribute based on the group membership attribute `memberOf`.

Group membership in many LDAP directories is carried in the `memberOf` attribute. In Novell's eDirectory it is carried in the `groupMembership` attribute.

```
<resolver:AttributeDefinition xsi:type="Script" xmlns="urn:mace:shibboleth:2.0:resolver:ad"
                              id="eduPersonAffiliation"
                              sourceAttributeID="eduPersonAffiliation">

    <!-- Dependency that provides the source attribute. -->
    <resolver:Dependency ref="myLDAP" />

    <!-- SAML 1 and 2 encoders for the attribute. -->
    <resolver:AttributeEncoder xsi:type="SAML1String" xmlns="urn:mace:shibboleth:2.0:attribute:encoder"
                               name="urn:mace:dir:attribute-def:eduPersonAffiliation" />
    <resolver:AttributeEncoder xsi:type="SAML2String" xmlns="urn:mace:shibboleth:2.0:attribute:encoder"
                               name="urn:oid:1.3.6.1.4.1.5923.1.1.1.1"
                               friendlyName="eduPersonAffiliation" />

    <!-- The script, wrapped in a CDATA section so that special XML characters don't need to be removed -->
    <Script><![CDATA[
        importPackage(Packages.edu.internet2.middleware.shibboleth.common.attribute.provider);

        // Create attribute to be returned from definition, but iff we have to
                if (null == eduPersonAffiliation) {
                eduPersonAffiliation = new BasicAttribute("eduPersonAffiliation");
                }

        // Add at least one value
        eduPersonAffiliation.getValues().add("affiliate");

        // If the user has group membership
        if (typeof memberOf != "undefined" && memberOf != null ){
            // The go through each group membership and add the appropriate affiliation
            // The IdP will remove duplicate values so we don't need to worry about that here
            for ( i = 0; memberOf != null && i < memberOf.getValues().size(); i++ ){
                value = memberOf.getValues().get(i);

                if (value.indexOf("OU=Students") > -1){
                    eduPersonAffiliation.getValues().add("student");
                }

                if (value.indexOf("OU=Teachers") > -1){
                    eduPersonAffiliation.getValues().add("faculty");
                    eduPersonAffiliation.getValues().add("staff");
                }

                if (value.indexOf("OU=Staff") > -1){
                    eduPersonAffiliation.getValues().add("staff");
                }
            }
        }
    ]]></Script>
</resolver:AttributeDefinition>
```

Generate eduPersonAffiliation based on recursive group membership in Active Directory

Contributed by Vladimir Mencl, Tuakiri Federation, New Zealand

This example expands on the previous example by Halm Reusser. Some Active Directory (AD) servers use recursive group membership. E.g. an CN=All-Staff group would only contain other groups (per organizational unit) which then contain the users. This example defines a second LDAP connector that looks up the transitive closure of the user's group membership (using the AD LDAP_MATCHING_RULE_IN_CHAIN extension filter).

> As the groups used for the eduPersonAffiliation attribute use recursive membership, for this attribute specifically, it was necessary to define a second LDAP connector that looks up all the recursive group membership for a user.
>
> We obtain the list of all groups a user is *recursively* a member of by using the Microsoft the AD extension LDAP_MATCHING_RULE_IN_CHAIN (1.2.840.113556.1.4.1941) in the search filter to do the recursive search. Further documentation on this is available at http://msdn.microsoft.com/en-us/library/windows/desktop/aa746475

We define a second LDAP connector (called `groupLDAP` here) in `attribute-resolver.xml` that does a recursive lookup on a user DN. This connector depends on the `distinguishedName` attribute provided by the main LDAP connector (`myLDAP`) – to have the user's full DN to search the groups for – and is in turn used by the `memberOf` attribute that is later used in the `eduPersonAffiliation` attribute.

Specifically, this connector:

- Uses a search base in the groups OU: `OU=Groups and Resources,DC=EXAMPLE,DC=ORG`
- Increases `maxResultSize` from 1 to 1000 (no way to say unlimited, this is a reasonable value)
    - Increases `searchTimeLimit` from 3000ms to 10,000ms
- Has `mergeResults="true"` (to flatten the results - return a single object with multiple values in the `distinguishedName` attribute)
- Uses a `FilterTemplate` that searches based on the DN returned by the main LDAP conenctor
    - We cannot use `"dn"` directly (not seen as an attribute), but we can use `"distinguishedName"` (visible in AD as an attribute on all objects)
- Requests only the `distinguishedName` attribute on the result objects (not to download the complete group objets) – this is done via a (space-separated) `<ReturnAttributes>` element (to come after `FilterTemplate`)
    - Again, as we cannot see `“dn”`, we request `distinguishedName` as the group name. All of the group names are then flattened into a single multivalued attribute (used by the `memberOf` definition).

```
    <!-- get the user's DN from the main LDAP connector (myLDAP) for searching the groups the user is in -->
    <resolver:AttributeDefinition id="distinguishedName" xsi:type="ad:Simple"
              sourceAttributeID="distinguishedName">
        <resolver:Dependency ref="myLDAP" />
        <!-- no encoder needed -->
    </resolver:AttributeDefinition>

    <!-- search for all groups the user is recursively in - and flatten the distinguishedName(s) of all the
groups into a single multivalued attribute -->
    <resolver:DataConnector id="groupLDAP" xsi:type="dc:LDAPDirectory"
        ldapURL="ldaps://ad.example.org"
        baseDN="OU=Groups and Resources,DC=EXAMPLE,DC=ORG"
        principal="CN=Binder,DC=EXAMPLE,DC=ORG"
        principalCredential="PASSWORD-HERE"

        maxResultSize="1000"
        mergeResults="true"
        searchTimeLimit="PT10.000S"

        >
        <resolver:Dependency ref="distinguishedName" />
        <dc:FilterTemplate>
            <![CDATA[
                (member:1.2.840.113556.1.4.1941:=${distinguishedName.get(0)})
            ]]>
        </dc:FilterTemplate>
        <dc:ReturnAttributes>distinguishedName</dc:ReturnAttributes>
        <dc:LDAPProperty name="java.naming.referral" value="follow"/>
    </resolver:DataConnector>

    <!-- define the memberOf attribute based on the distinguishedName attribute returned by the groupLDAP
connector - names of all groups the user is in -->
    <resolver:AttributeDefinition id="memberOf" xsi:type="ad:Simple"
              sourceAttributeID="distinguishedName">
        <resolver:Dependency ref="groupLDAP" />
        <!-- no encoder needed -->
    </resolver:AttributeDefinition>

    <!-- define eduPersonAffiliation same as in the previous example - but put a dependency on the memberOf
attribute defined through the groupLDAP connector -->
    <resolver:AttributeDefinition id="eduPersonAffiliation" xsi:type="ad:Script">
        <resolver:Dependency ref="memberOf" />
        <resolver:DisplayName xml:lang="en">Affiliation type</resolver:DisplayName>
        <resolver:AttributeEncoder xsi:type="enc:SAML1String" name="urn:mace:dir:attribute-def:
eduPersonAffiliation" />
        <resolver:AttributeEncoder xsi:type="enc:SAML2String" name="urn:oid:1.3.6.1.4.1.5923.1.1.1.1"
friendlyName="eduPersonAffiliation" />
        <ad:Script>
        <![CDATA[
                importPackage(Packages.edu.internet2.middleware.shibboleth.common.attribute.provider);
                if (eduPersonAffiliation == null) {
                        eduPersonAffiliation = new BasicAttribute("eduPersonAffiliation");
                }
                is_Staff = memberOf != null && memberOf.getValues().contains("CN=All-Staff,OU=Groups and
Resources,DC=EXAMPLE,DC=ORG");

                if (is_Staff) { eduPersonAffiliation.getValues().add("staff"); };
                if (is_Staff) { eduPersonAffiliation.getValues().add("member"); };
        ]]>
        </ad:Script>
    </resolver:AttributeDefinition>
```

Add common-lib-terms to all Staff and Students

Contributed by: Halm Reusser, SWITCH, Switzerland

This example adds the common-lib-terms URN to the `eduPersonEntitlement` attribute for a principal with affiliation `staff` or `student` while keeping any entitlement values retrieved from the directory. If your organization also assigns the affiliation value 'faculty' extend the script to add it for these users as well.

```
<resolver:AttributeDefinition xsi:type="Script" xmlns="urn:mace:shibboleth:2.0:resolver:ad"
                              id="eduPersonEntitlement"
                              sourceAttributeID="eduPersonEntitlement">

    <!-- Dependency that provides the source attribute. -->
    <resolver:Dependency ref="myLDAP" />
    <resolver:Dependency ref="eduPersonAffiliation" />

    <!-- SAML 1 and 2 encoders for the attribute. -->
    <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" />

    <!-- The script, wrapped in a CDATA section so that special XML characters don't need to be removed -->
    <Script><![CDATA[
        importPackage(Packages.edu.internet2.middleware.shibboleth.common.attribute.provider);

        // Create attribute to be returned from definition
        if (eduPersonEntitlement == null) eduPersonEntitlement = new BasicAttribute("eduPersonEntitlement");

        if (eduPersonAffiliation.getValues().contains("staff") ||
                eduPersonAffiliation.getValues().contains("student")) {
            eduPersonEntitlement.getValues().add("urn:mace:dir:entitlement:common-lib-terms");
        }
    ]]></Script>
</resolver:AttributeDefinition>
```

Generate eduPersonEntitlement based on the age of the subject

Contributed by: Peter Schober, ACOnet, Austria.

German-language page only at this time, but feel free to ask about this on the users mailing list if configuration and/or code are not sufficiently clear. Uses s chacDateOfBirth and a Script-type AttributeDefinition to calculate whether the subject satisfies the business criteria for carrying a specific eduPersonEntitlement value, which include an age limit, *without* leaking the age (or birth date) of the subject to the relying party.

- https://wiki.univie.ac.at/display/federation/IDP+3+USI+Wien+eduPersonEntitlement (for IDPv3; the old version for IDPv2 is still available, though)