IdP Attribute Resolution

Migrating configuration to version 3

SWITCHaai Team
aai@switch.ch

Attribute processing in IdP version 3

1. Resolution
2. Filtering
3. Encoding

Compatibility with version 2

- Same XML syntax as v2, should be "nearly 100% compatible"
  Any regression should be reported as a bug
- Some deprecated elements are ignored
- **Exception**: scripted attribute definitions
  Deprecated interfaces may require complex scripts to be adapted

Why upgrade your configuration?

- No warning for using legacy configuration mode
- Delete ignored elements
- Grouping attribute definitions in separate files

⇒ Less misleading, smaller files, clearer

Deprecated items

- Principal connectors
- NameID encoders
- Transient identifier attribute definitions
- Persistent identifier data connectors and attribute definitions

All replaced by NameID generation/comsumption, see next presentation

New features in version 3

- Property replacement: `%{my.property}`
  - Move passwords in a dedicated file
  - Extract duplicated data like URLs
- Can split configuration into several files
- External Spring configuration for data connectors
- Activation conditions on attribute encoders, attribute definitions and data connectors

New features example

```
<resolver:DataConnector id="myStoredId"
  xsi:type="dc:StoredId"
  generatedAttributeID="persistentID"
  sourceAttributeID="%{idp.persistentId.sourceAttribute}"
  salt="%{idp.persistentId.salt}"
  queryTimeout="0">
  <resolver:Dependency ref="%{idp.persistentId.sourceAttribute}" />
  <dc:BeanManagedConnection>
    shibboleth.PostgreSQLDataSource
  </dc:BeanManagedConnection>
</resolver:DataConnector>
```

```
<resolver:DataConnector id="myLDAP" xsi:type="dc:LDAPDirectory"
  ldapURL="%{idp.attribute.resolver.LDAP.ldapURL}"
  baseDN="%{idp.attribute.resolver.LDAP.baseDN}"
  principal="%{idp.attribute.resolver.LDAP.bindDN}"
  principalCredential="%{idp.attribute.resolver.LDAP.bindDNCredential}"
  useStartTLS="%{idp.attribute.resolver.LDAP.useStartTLS:true}">
  <dc:FilterTemplate>%{idp.attribute.resolver.LDAP.searchFilter}</dc:Fi
  <!-- actual values come from conf/ldap.properties -->
</resolver:DataConnector>
```


Hands-on 1

Add a new local attribute definition and release it to the attribute viewer.

```
UZH SAP user ID
SAP User ID used internally by University of Zürich
SAML1 Name: urn:mace:unizh.ch:uzhSapUserId
SAML2 Name: urn:oid:1.3.6.1.4.1.11817.1.1.2.13
Friendly name: uzhSAPUserId
Format: SAP-<username>
```

source: Resource Registry (https://rr.aai.switch.ch/list_attributes.php?sort=name)

Hands-on 1 solution

attribute-resolver-local.xml

New file with:

```
<resolver:AttributeDefinition id="uzhSAPUserId" xsi:type="ad:Template">
  <resolver:Dependency ref="uid" />
  <resolver:AttributeEncoder xsi:type="enc:SAML1String"
    name="urn:mace:unizh.ch:uzhSapUserId" encodeType="false" />
  <resolver:AttributeEncoder xsi:type="enc:SAML2String"
    name="urn:oid:1.3.6.1.4.1.11817.1.1.2.13"
    friendlyName="uzhSAPUserId" encodeType="false" />
  <ad:Template>SAP-${uid}</ad:Template>
  <ad:SourceAttribute>uid</ad:SourceAttribute>
</resolver:AttributeDefinition>
```


Hands-on 1 solution

attribute-filter-local.xml

New file with:

```
<afp:AttributeFilterPolicy id="uzhSAPUserId">
  <afp:PolicyRequirementRule xsi:type="basic:AttributeRequesterString"
    value="https://attribute-viewer.aai.switch.ch/shibboleth" />
  <afp:AttributeRule attributeID="uzhSAPUserId">
      <afp:PermitValueRule xsi:type="basic:ANY" />
  </afp:AttributeRule>
</afp:AttributeFilterPolicy>
```

Doable in Resource Registry too

Hands-on 1 solution

services.xml

Loads both new files:

```
<util:list id="shibboleth.AttributeResolverResources">
  <value>%{idp.home}/conf/attribute-resolver-switchaai-core.xml</value>
  <value>%{idp.home}/conf/attribute-resolver-connectors.xml</value>
  <value>%{idp.home}/conf/attribute-resolver-other.xml</value>
  <value>%{idp.home}/conf/attribute-resolver-local.xml</value>
</util:list>
```

```
<util:list id="shibboleth.AttributeFilterResources">
  <ref bean="FileBackedSWITCHaaiAttributeFilter"/>
  <value>%{idp.home}/conf/attribute-filter-local.xml</value>
</util:list>
```


ScriptedAttribute differences

- IdP API for scripts has changed
  - output attribute variable already created
  - `setValues()` removed
  - and more, see ScriptedAttributeDefinition (https://wiki.shibboleth.net/confluence/display/IDP30/ScriptedAttributeDefinition for details)
- Alternatives: mapped or template attribute definitions
- JavaScript engine change between Java 7 (Rhino) and 8 (Nashorn) ⇒ even *more* things to adapt

Mapped attribute definition

- Many-to-many mapping from source attributes values
- Each input value is checked against every mapping
  if match → input replaced by return value
- Supports regular expressions
- One or more mapping statements: `ValueMap`
  - one `ReturnValue`
  - one or more `SourceValue`
- Zero or one `DefaultValue`, fixed or pass-through

Mapped attribute example

```
<resolver:AttributeDefinition id="eduPersonAffiliation"
  xsi:type="ad:Mapped" sourceAttributeID="eduPersonAffiliation">
  <!-- common attributes omitted -->
  <ad:DefaultValue passThru="true"/>
  <ad:ValueMap>
    <ad:ReturnValue>member</ad:ReturnValue>
    <ad:SourceValue>staff|student|faculty|employee</ad:SourceValue>
  </ad:ValueMap>
  <ad:ValueMap>
    <ad:ReturnValue>$1</ad:ReturnValue>
    <ad:SourceValue>(staff|student|faculty|employee)</ad:SourceValue>
  </ad:ValueMap>
</resolver:AttributeDefinition>
```

- student → student, member
- alumni → alumni

Template attribute definition

- Combines values of multiple input attributes
- Velocity template
- **All** input attributes must contain the **same number** of values
- Zero or one `Template`
- One or more `SourceAttribute`

Template attribute example

```
<resolver:AttributeDefinition id="template" xsi:type="ad:Template">
  <resolver:Dependency ref="OtherAttribute" />
  <resolver:Dependency ref="myLdap" />
  <!-- common attributes omitted -->
  <ad:Template>${attrFromLdap}::${OtherAttr}</ad:Template>
  <ad:SourceAttribute>attrFromLdap</ad:SourceAttribute>
  <ad:SourceAttribute>OtherAttr</ad:SourceAttribute>
</resolver:AttributeDefinition>
```

- ldap1, other1 → ldap1::other1
- ldap1, ldap2, other1 → error!

Hands-on 2

Add new `eduPersonEntitlement` values of the form `https://example.org/groups/<objectClass>` using the LDAP `objectClass` attribute, *without* using script attributes

Hands-on 2 solution

Create a new template attribute definition (without encoders) to generate the new values:

```
<resolver:AttributeDefinition id="eduPersonEntitlement.groups"
  xsi:type="ad:Template" sourceAttributeID="objectClass" dependencyOnly
  <resolver:Dependency ref="myLDAP" />
  <ad:Template>https://example.org/groups/${objectClass}</ad:Template>
  <ad:SourceAttribute>objectClass</ad:SourceAttribute>
</resolver:AttributeDefinition>
```

Add that new attribute to the definition of `eduPersonEntitlement`:

```
<resolver:AttributeDefinition id="eduPersonEntitlement" xsi:type="ad:Si
  <resolver:Dependency ref="eduPersonEntitlement.common-lib-terms" />
  <resolver:Dependency ref="eduPersonEntitlement.groups" />
  <!-- ... rest of original definition -->
</resolver:AttributeDefinition>
```

References

- Shibboleth wiki: AttributeResolverConfiguration
  (https://wiki.shibboleth.net/confluence/display/IDP30/AttributeResolverConfiguration and its child pages)
- Shibboleth wiki: AttributeFilterConfiguration
  (https://wiki.shibboleth.net/confluence/display/IDP30/AttributeFilterConfiguration)
- Shibboleth wiki: AttributeDefinitionConfiguration
  (https://wiki.shibboleth.net/confluence/display/IDP30/AttributeDefinitionConfiguration and its child pages)
- Rhino Migration Guide
  (https://wiki.openjdk.java.net/display/Nashorn/Rhino+Migration+Guide)