Attribute Resolution

SWITCH
Serving Swiss Universities

Notes:

Terms: Attribute

- A piece of information about a user. Each attribute has a unique ID and has zero of more values.
- Shibboleth attributes are protocol-agnostic data structures.

Notes:

Terms: SAML Attribute

- An attribute that is represented in SAML notation.
- Shibboleth transforms attributes into SAML attributes by a process known as encoding.

Notes:

Terms: Data Connector

- A plugin that creates *multiple* attributes from information in data sources like LDAP and databases.
- Shibboleth currently supports static, LDAP, relational database, computed, and stored ID data connectors.

Notes:

Terms: Attribute Definition

- A plugin that creates a *single* attribute by transforming other attributes and state information.
- Shibboleth currently supports simple, scoping, regex, mapping, template, scripting, principal name, and principal authentication method attribute definitions.

Notes:

Terms: Attribute Encoder

- A plugin that converts an attribute into a protocol specific form, like a SAML attribute.
- Attribute encoders are associated with an attribute through the attribute's attribute definition.

Notes:

Terms: Principal Connector

- A plugin that converts a name identifier, provided by a relying party, into the internally used userid.

Notes:

Terms: Attribute Resolver

- A subsystem in Shibboleth responsible for fetching, transforming, and associating encoders with attributes.
- Only attributes produced by attribute definitions leave the resolver and are available to other parts of the system.

Notes:

A bit of logging configuration

- Edit logging.xml
- Turn the logging level of each currently defined logger to WARN
- Add a new logger:

```
<logger name="edu.internet2.middleware.shibboleth.common.attribute">
   <level value="DEBUG" />
</logger>
```


Notes:

Attribute Goals

- Define a simple attribute with a static value.
- Gather user information from an LDAP directory
- Create attribute definition that release some information with simple values and other information with scoped values

Notes:

Data Connector: Configuration

- Data connectors are configured in attribute-resolver.xml
- `<DataConnector>` defines a data connector
- Every data connector has a `id` attribute that uniquely identifies it.
- Every data connector has a `xsi:type` attribute that defines the type of the handler.
- Each type has its own set of configuration options.

 https://spaces.internet2.edu/display/SHIB2/IdPAddAttribute

Notes:

Data Connector: Configuration

- Some connectors will need information collected by another plugin in order to work. This is represented by a
  `<resolver:Dependency ref="NAME" />`
- The dependency is declared before any other configuration elements.
- The value of the `ref` attribute is the ID of the plugin upon which the connector depends.

Notes:

Data Connector: Static

- Static data connector adds attributes to every resolved account.
- Type attribute value:
  `Static`
- Configuration attributes:
  none

 https://spaces.internet2.edu/display/SHIB2/ResolverStaticDataConnector

Notes:

Data Connector: Static

- The produced attributes are defined by:
  `<Attribute id="ATTRIBUTE_ID">`
- Values are added by:
  `<Value>VALUE</Value>`
- An attribute may have more than one value.

Notes:

Data Connector: Static

- Create an attribute ‘eduPersonAffiliation’ that has one value ‘member’

```
<resolver:DataConnector id="staticEPA”
    xsi:type="Static”
    xmlns="urn:mace:shibboleth:2.0:resolver:dc">

    <Attribute id="eduPersonAffiliation">
        <Value>member</Value>
    </Attribute>

</resolver:DataConnector>
```


Notes:

Data Connector Resolution

- Restart the IdP and login again
- Do you see anything in your log file about the static data connector being invoked?
- The IdP only invokes a data connector if another an attribute definition or another invoked data connector depends on it.

Notes:

Attribute Definition: Configuration

- Attribute definitions are configured in attribute-resolver.xml
- `<AttributeDefinition>` defines a definition
- Every definition has a `id` attribute that uniquely identifies it.
- Every definition has a `xsi:type` attribute that defines the type of the handler.
- Each type has its own set of configuration options.

 https://spaces.internet2.edu/display/SHIB2/IdPAddAttribute

Notes:

Attribute Definition: Configuration

- Most definitions will need information collected by another plugin in order to work. This is represented by a
  `<resolver:Dependency ref="NAME" />`
- The dependency is declared before any other configuration elements.
- The value of the `ref` attribute is the ID of the plugin upon which the definition depends.

Notes:

Attribute Definition: Simple

- Attribute definition that simply releases an attribute from the resolver.
- Type attribute value:
  `Simple`
- Configuration attributes:
  `sourceAttributeID` - the name of the attribute, provided the dependencies, that will provide the values for this attribute

 https://spaces.internet2.edu/display/SHIB2/ResolverSimpleAttributeDefinition

Notes:

Attribute Definition: ePA

- Putting it all together we define an attribute definition for eduPersonAffiliation as follows:

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

    <resolver:Dependency ref="staticEPA" />

</resolver:AttributeDefinition>
```

Notes:

Attribute Definition: Testing

- Restart the IdP
- Watch the logs using
  tail -f /opt/shibboleth-idp/logs/idp-process.log
- Log in to https://sp#.example.com/cgi-bin/attribute-viewer

Notes:

Attribute Encoders: Configuration

- Attribute encoders are configured as children of an attribute definition.
- `<AttributeEncoder>` defines an encoder
- Every definition has a `xsi:type` attribute that defines the type of the handler.
- Each type has its own set of configuration options.

 https://spaces.internet2.edu/display/SHIB2/IdPAddAttribute

Notes:

Attribute Encoder: Basic SAML 1

- A SAML 1 encoder always looks like this:

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

- Only the name changes

 https://spaces.internet2.edu/display/SHIB2/SAML1StringAttributeEncoder

Notes:

Attribute Encoder: Basic SAML 2

- A SAML 2 encoder always looks like this:

```
<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" />
```

- Only the name and friendly name changes

 https://spaces.internet2.edu/display/SHIB2/SAML2StringAttributeEncoder

Notes:

Attribute Encoder: Configuration

- Add SAML 1 and SAML 2 attribute encoders to your `eduPersonAffiliation`

- eduPersonAffiliation:
  `urn:mace:dir:attribute-def:eduPersonAffiliation`
  `urn:oid:1.3.6.1.4.1.5923.1.1.1.1`

Notes:

Attribute Goals

- Define a simple attribute with a static value.
- Gather user information from an LDAP directory
- Create attribute definition that release some information with simple values and other information with scoped values

Notes:

Data Connector: LDAP

- Data connector that pulls user information from LDAP
- Type attribute value:
  `LDAPDirectory`
- Configuration Attributes:
  `ldapURL` - ldap server connection URL
  `baseDN` - search filter base DN
  `principal` - DN of user to connect as
  `credential` - principal's password

 https://spaces.internet2.edu/display/SHIB2/ResolverLDAPDataConnector

Notes:

Data Connector: LDAP

- Lastly the LDAP data connector contains a child element `<FilterTemplate>`
- The template is used to construct the query filter, for now we'll use
  `(uid=$requestContext.principalName)`

Notes:

Data Connector: LDAP

- If you put it all together you should get:

```
<resolver:DataConnector id=”localLDAP”
    xsi:type="LDAPDirectory”
    xmlns="urn:mace:shibboleth:2.0:resolver:dc”
    ldapURL="ldap://127.0.0.1:10389”
    baseDN="ou=people,dc=example,dc=org”
    principal="uid=admin,ou=system”
    principalCredential=”password”>
    <FilterTemplate>
        (uid=$requestContext.principalName)
    </FilterTemplate>
</resolver:DataConnector>
```


Notes:

Attribute Definition: ePA

- Add the LDAP data connector as a dependency to your `eduPersonAffiliation` attribute definition.
- Run another test
- Note how the LDAP’s values are added to the value from the static data connector?

Notes:

Attribute Definition: ePPA

- Create a simple attribute definition, called `eduPersonPrimaryAffiliation` that has a `sourceAttributeID` of `eduPersonPrimaryAffiliation` and depends `localLDAP`

- Add attribute SAML1/2 string encoders:
  `urn:mace:dir:attribute-def:eduPersonPrimaryAffiliation`
  `urn:oid:1.3.6.1.4.1.5923.1.1.1.5`

Notes:

Attribute Scoping

- Some attribute values may have Scopes
- Scopes provide a domain within which an attribute value is valid
- Example:
  Georgetown University has a main campus, a law school, and a medical school. A professor at the law school may not have the same rights as a professor at the medical school.

Notes:

Attribute Definition: Scoped

- An attribute definition that adds a static scope
- Type attribute value: `Scoped`
- Configuration Attributes:
  - `sourceAttributeID` - ID of the attribute whose values will be scoped
  - `scope` - scope added to the attribute values

 https://spaces.internet2.edu/display/SHIB2/ResolverScopedAttributeDefinition

Notes:

Attribute Definition: Scoped

- Create an attribute definition for `eduPersonScopedAffiliation`.

```
<resolver:AttributeDefinition id=”eduPersonScopedAffiliation"
xsi:type=”Scoped”
xmlns="urn:mace:shibboleth:2.0:resolver:ad"
   sourceAttributeID=”eduPersonAffiliation”
scope=“example.org”>

     <resolver:Dependency ref=”localLDAP”/>

</resolver:AttributeDefinition>
```

Notes:

Attribute Definition: Prescoped

- Prescoped attribute values already contain the scope within the datasource
- Type attribute value: `Prescoped`
- Configuration Attributes:
  - `sourceAttributeID` - ID of the attribute with prescoped values
  - `scopeDelimiter` - the scope delimiter used in the attributes values (default: @)

 https://spaces.internet2.edu/display/SHIB2/ResolverPrescopedAttributeDefinition

Notes:

Attribute Definition: Prescoped

- Create an attribute definition that operates on the prescoped eduPersonPrincipalName attribute

```
<resolver:AttributeDefinition id=”eduPersonPrincipalName"
xsi:type=”Prescoped”
xmlns="urn:mace:shibboleth:2.0:resolver:ad"
   sourceAttributeID=”eduPersonPrincipalName”>

     <resolver:Dependency ref=”localLDAP" />

</resolver:AttributeDefinition>
```

Notes:

Attribute Encoders: Scoped

- An attributes scope may be written into a SAML message in two ways:
  - As an attribute on the SAML `<AttributeValue Scope="…">`
  - Using inline `value@scope` notation
- Notation used may be controlled by the `scopeType` attribute on the encoder. Values: `attribute`, `inline`

Notes:

Attribute Encoders: Scoped

- SAML 1 Scoped Value Encoder

```
<resolver:AttributeEncoder xsi:type="SAML1ScopedString"
xmlns="urn:mace:shibboleth:2.0:attribute:encoder"
name="urn:mace:dir:attribute-def:eduPersonPrincipalName" />
```

- SAML 2 Scoped Valued Encoder

```
<resolver:AttributeEncoder xsi:type="SAML2ScopedString"
xmlns="urn:mace:shibboleth:2.0:attribute:encoder"
name="urn:oid:1.3.6.1.4.1.5923.1.1.1.6"
    friendlyName="eduPersonPrincipalName" />
```

Notes:

More about Dependencies

- Any resolver plugin may have any number of dependencies.
- If more than one dependency provides the same attribute the dependant plugin operates on the effective union of values
- Attribute definitions may be marked with a `dependencyOnly="true"` attribute. This ensures the value is never released outside the resolver (and speeds up filtering a bit).

Notes:

Data Connector Failover

- Data connectors may define failover connectors such that if the data connector fails the failover connector is invoked.
- If more than one failover connector is defined they are tried in order until one succeeds.
- They are defined using:

```
<resolver:FailoverDataConnector ref="CONNECTOR_ID_1" />
```

Notes: