CustomNameIDGenerationConfiguration

Overview

An identifier that is neither "transient" nor "persistent" format is of a more general category referred to for convenience in this documentation as a "custom" identifier. Plugins are included for generating a custom identifier based on an attribute produced by the attribute resolution process. In most cases, this should be sufficient. In unusual scenarios, you may also build your own generator plugin, but this should rarely be necessary.

By default, support for custom identifiers is included in *saml-nameid.xml* but is only partially configured and commented out (because the specifics are a local matter). In the process outlined below, you will create/uncomment one or more copies of the custom generator bean(s) appropriately, and ensure the underlying source attribute(s) are released to the applicable relying parties.

There are no properties relevant to the use of custom identifiers in *saml-nameid.properties*, so do not be distracted by that content; those settings apply only to the use of "transient" and "persistent" format identifiers.

General Procedure

To support a custom identifier Format, take the following steps:

1. Determine the Format constant you need to support (note that most of the constants used in SAML contain a "1.1" in the URN and apply to both SAML versions).
2. Choose one or more attributes to use to supply the identifier's value (e.g. "mail"). The first value found (in order listed) will be used.
3. Configure your attribute filter policy to release the attribute(s) chosen to the relevant SP(s).
4. Uncomment or create a generator bean in *saml-nameid.xml* for SAML 1 and/or SAML 2 as required.

   SAML 2.0 E-Mail Format Example

   ```
   <util:list id="shibboleth.SAML2NameIDGenerators">
           <ref bean="shibboleth.SAML2TransientGenerator" />

           <!-- SAML 2 EXAMPLE -->
       <bean parent="shibboleth.SAML2AttributeSourcedGenerator"
                   p:omitQualifiers="true"
                   p:format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress"
                   p:attributeSourceIds="#{ {'mail', 'othermail'} }" />
   </util:list>
   ```

   The `format` property is self-explanatory and must contain a single value per bean. The `attributeSourceIds` property is a list of attribute IDs to source the value from (the first value found will be used), and you can embed a comma-delimited list of quoted IDs inside the double braces (see SpringConfiguration). Most custom formats in SAML should generally leave out the `NameQualifier` and `SPNameQualifier` attributes and there's a property per above that needs to be set to suppress them.
5. Lastly, trigger the appropriate Format to be selected by manipulating the selection process described in the NameIDGenerationConfiguration topic. If you control the SP's metadata (which is very common when this use case arises), the best way to do this is by inserting a `<NameIDFormat>` element into the metadata. If this isn't possible, a `nameIDFormatPrecedence` profile configuration property can be used in a relying party override definition.

The process above requires that you explicitly release the attribute to use as a source in your filter policy. If you wish, you can avoid this step by using a generator property (`useUnfilteredAttributes`) that allows an unreleased attribute to be used as a source, but note that doing so creates a sort of "pseudo-policy" exposing information to an SP outside of the normal filtering process.

Dealing with Conflicting Requirements

Ideally, you should try and avoid scenarios in which you must generate a different identifier of the same Format for two different SPs. In other words, a given Format should contain the same data regardless of the SP involved. This makes it possible to maintain a simple configuration of any number of generators as above, because the Format chosen will drive the approach used.

If you must deviate from that rule, you can accomodate that by attaching an `activationCondition` property to a generator bean that triggers based on a particular relying party. In that fashion, you can include multiple generators for a given Format, but limit their use to specific SPs.

Example of a Generator for a specific SP

```
<bean parent="shibboleth.SAML2AttributeSourcedGenerator"
        p:omitQualifiers="true"
        p:format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress"
        p:attributeSourceIds="#{ {'mail', 'othermail'} }">

        <property name="activationCondition">
                <bean parent="shibboleth.Conditions.RelyingPartyId" c:candidate="https://sp.example.com
/shibboleth" />
        </property>
</bean>
```

Dealing with "Unspecified"

Bear in mind that many vendors make the poor decision to use a Format called `urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified`, which makes it very difficult to interoperably deal with them. In addition to being, well, unspecified, each vendor may ascribe a totally different set of requirements to that format, making it hard to properly generate without a lot of special rules and configuration.

Shibboleth does not honor this Format when it is encountered in a `<NameIDPolicy>` element in a request or in an SP's metadata and it is ignored. As a result, if you must support this Format, you must trigger its use with a `nameIDFormatPrecedence` profile configuration property:

Example Relying Party override specifying a NameID Format

```
<bean parent="RelyingPartyByName" c:relyingPartyIds="https://scroogle.com">
    <property name="profileConfigurations">
        <list>
            <bean parent="SAML2.SSO" p:encryptAssertions="false"
                p:nameIDFormatPrecedence="urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified" />
        </list>
    </property>
</bean>
```

We strongly urge deployers to avoid the use of this Format when possible. Note that in many cases when vendors claim to "require" its use, what they really mean (aside from "we're not interesting in supporting SAML properly") is that they don't **care** what Format you use.

You should **always** start by trying to use a standard Format of some sort that fits the data you need to send. If no standard Format exists for the data, use the URI name of the underlying SAML Attribute as the Format. For example, passing an employee ID should be done using a Format of `urn:oid:2.16.840.1.113730.3.1.3`