# InCommon Shibboleth IdP Training - NameID Configuration

The NameID, short for Name Identifier, is a value that identifies the subject of the SAML assertion. It is similar to an attribute, but it is sent in the `<Subject>` part of the assertion rather than the `<AttributeStatement>`. In most federated use cases, the NameID is not used. By default, Shibboleth populates the NameID with a temporary string called a "transient" identifier, which changes with each new assertion the IdP issues, and most of the time, you won't need to configure this behavior. However, many cloud vendors expect the NameID to be populated with the "username" or "unique ID" of the user, in lieu of looking for a standard attribute such as `eduPersonPrincipalName`. In these cases, you'll need to do some special configuration to populate the NameID with the expected value. Several very large cloud providers require this type of configuration, including:

- Microsoft (Office 365)
- Google Apps
- DocuSign

## NameID Formats

- In addition to the NameID itself, the assertion will also include a "NameID format" identifier. These are a way to tell the service provider what type of NameID is being sent in the assertion.
- In the metadata, SPs typically (but not always) specify what "format" of NameID they prefer. Multiple formats may be specified in the SP's metadata, indicating that any of the listed formats are acceptable.

### Common NameID Formats

These are what you are most likely to encounter in most single sign-on integrations with cloud vendors:

- Persistent (`urn:oasis:names:tc:SAML:2.0:nameid-format:persistent`)
  A permanent identifier; for example, a username, employee ID number, or other unique campus identifier

  > ⚠ The actual usage case for the "persistent" format is to generate privacy-preserving attributes that are unique for each SP and guaranteed not to change. However, many cloud vendors will disregard the spec and ask that you use this format in conjunction with a simple user identifier.

- Transient (`urn:oasis:names:tc:SAML:2.0:nameid-format:transient`)
  A temporary identifier, usually unique to the SP. This is the default setting for Shibboleth. The Shibboleth IdP will include an automatically-generated transient NameID with each assertion it generates, unless a different format is explicitly specified in the SP's metadata or in `relying-party.xml`.
- Email Address (`urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress`)
  An email address (user@host.domain)
- Unspecified or "catch-all" (`urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified`)
  Can be pretty much anything, but in practice, often populated with a simple unqualified user identifier

In practice, an SP typically will request a certain NameID format, either in its metadata (look for one or more `<NameIdFormat>` elements), or out-of-band (e.g. in a single sign-on configuration document).

## Populating the NameID for a specific SP

For most cloud SSO integrations that require a custom NameID, you'll need to populate the NameID with the value of an attribute specified in `attribute-resolver.xml`. Here is a typical recipe. In this example, the SP entity ID is `https://sp.example.com/shibboleth`, the NameID format is `urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified`, and the NameID will be populated with the value of the `googleAppsPrincipal` attribute.

> ⚠ Example only – do not paste these snippets directly into your configuration. See "tasks" section below for the actual exercise.

1. In `saml-nameid.xml`, look for the list of SAML 2 NameID Generators (`<util:list id="shibboleth.SAML2NameIDGenerators">`). Inside this block, add a new `<bean>` that specifies the NameID format, entity ID for the target SP, and attribute that you want to use:

**saml-nameid.xml**

```
<bean parent="shibboleth.SAML2AttributeSourcedGenerator"
      p:format="urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified"
      p:attributeSourceIds="#{ {'googleAppsPrincipal'} }">

      <property name="activationCondition">
          <bean parent="shibboleth.Conditions.RelyingPartyId"
                c:candidate="https://sp.training.incommon.org/shibboleth" />
      </property>

</bean>
```

2. In `attribute-filter.xml`, make sure that you're releasing the attribute to the SP:

**attribute-filter.xml**

```
<AttributeFilterPolicy id="exampleServiceProvider">
    <PolicyRequirementRule xsi:type="Requester" value="https://sp.training.incommon.org/shibboleth" />

    <AttributeRule attributeID="googleAppsPrincipal" permitAny="true" />
</AttributeFilterPolicy>
```

If you release an attribute that has encoders attached to it, the attribute will also be included in the `<AttributeStatement>` of the assertion. Some SPs may balk at this (although, in practice, most won't care). To avoid this side effect, you can add a new attribute definition (in `attribute-resolver.xml`) that does not include any `<AttributeEncoder>` blocks.

3. Sometimes, you'll also need to add an entry to `relying-party.xml` that explicitly specifies the NameID format you want to use. Typically, you'll need to do this if:
    - The SP's metadata does not have a `<NameIDFormat>` section that includes the desired format(s) (in these cases, absent a specific relying party configuration, the IdP will use an automatically-generated transient identifier, which usually is not what you will want).
    - The SP is requesting the `urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified` format (the IdP will ignore this specific format if it sees it in the metadata).

**relying-party.xml**

```
<bean parent="RelyingPartyByName" c:relyingPartyIds="https://sp.training.incommon.org/shibboleth">
    <property name="profileConfigurations">
        <list>
            <bean parent="SAML2.SSO" p:nameIDFormatPrecedence="urn:oasis:names:tc:SAML:1.1:nameid-format:
unspecified" />
        </list>
    </property>
</bean>
```

# Tasks

**Configure your IdP to populate the NameID field with the user's email address (value of the `mail` attribute) for all assertions sent to the classroom SP.**

Before you start, take a look at the classroom SP's metadata. You can download the metadata at https://sp.training.incommon.org/Shibboleth.sso/Metadata, or look in your IdP's cached copy of the classroom metadata aggregate, on your IdP container, in `/opt/shibboleth-idp/metadata/ShibTrain1-metadata.xml`. The SP's metadata should be one of the first two entries at or near the top of the file. Notice that the SP's metadata does not contain any `<NameIDFormat>` tags. This indicates that the SP doesn't have a preference for what format of NameID you send it. In these cases, in its default configuration, the IdP will always send a transient NameID, unless you tell it to do otherwise in `relying-party.xml`. Therefore, you'll need to do the following three things (described in greater detail in the previous section):

1. Edit `saml-nameid.xml` and add an attribute-sourced NameID generator for the classroom SP that uses `mail` as the attribute

> In default IdP configurations, `saml-nameid.xml` is not automatically reloaded when it changes, so you'll need to restart the IdP after editing it.

2. Check `attribute-filter.xml` to ensure you have released the `mail` attribute to the classroom SP
3. Edit `relying-party.xml` and add a relying party override for the classroom SP that specifies an explicit NameID format precedence (`urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress`)

When finished, rebuild and restart your IdP container, and log in to the classroom SP. The landing page should show the email address in the "Name Identifier" section.

If you have problems, see the "troubleshooting" section (below), or ask one of your friendly trainers.

# Troubleshooting

Custom NameID configurations require editing several configuration files, and as such, can sometimes be error-prone and tricky to troubleshoot. If the NameID isn't getting populated the way you expect for a certain SP, the first place to look is `idp-process.log`. For each assertion the IdP generates, a log entry will be created in this file that includes (among other useful information) the NameID that the IdP generated and used in the assertion.

**idp-process.log entry**

```
2018-09-28 17:09:15,937 - INFO [Shibboleth-Audit.SSO:275] - 20180928T170915Z|urn:oasis:names:tc:SAML:2.0:
bindings:HTTP-Redirect|allgfcphaibjgmlcdakgoofmmpmdejnkapmdlall|google.com/a/umbc.edu|http://shibboleth.net/ns
/profiles/saml2/sso/browser|urn:mace:incommon:umbc.edu|urn:oasis:names:tc:SAML:2.0:bindings:HTTP-
POST|_303e1132e75b63f2016267bf1bfa5871|AB12345|urn:oasis:names:tc:SAML:2.0:ac:classes:
PasswordProtectedTransport|googleAppsPrincipal|student1|_3e48179574a18249129ab1f645bb3860
```

In this example, the NameID sent to Google was `student1` (see the second-to-last vertical-bar delimited field). To the left of this field is the list of attributes the IdP released to the SP, which in this case, is just the single attribute `googleAppsPrincipal` (which is what it used to generate the NameID)

If you see a long, random string of characters where you were expecting to see an identifier, that means that the IdP was unable to generate a NameID, so it fell back on using a transient identifier. That's usually due to an error in the configuration. In these cases, it can be helpful to enable debug logging. In `/opt/shibboleth-idp/conf/logback.xml`, change `idp.loglevel.idp` to DEBUG, and restart (or wait for the file to reload). This log level will include useful debugging output related to NameID generation, and you usually can identify the problem by reading through the logs.

# References

- https://wiki.shibboleth.net/confluence/display/CONCEPT/NameIdentifiers
- https://wiki.shibboleth.net/confluence/display/IDP30/NameIDGenerationConfiguration