Shibboleth SP 3

Basic upstream documentation – start here!

- https://wiki.shibboleth.net/confluence/display/SP3/GettingStarted

More content will be provided in this wiki as needed:

Metadata configuration

> All examples below reference the eduID.at Metadata Signing Key in their configuration and will only work once the certificate has been copied to (by default) `/etc/shibboleth`. Do not skip this step!

> **Necessary config amendment**
>
> Some of the configuration examples below make use of an XML namespace prefix (`md:`) that needs to be declared in your `shibboleth2.xml`. Simply add the following line with the XML namespace declaration
>
> `xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"`
>
> to the root element (`SPConfig`) of your `shibboleth2.xml`, so that the beginning of the file looks like the example below. Note the **addition of the third line**, you don't need to change anything else:
>
> **SPConfig element in shibboleth2.xml**
>
> ```
> <SPConfig xmlns="urn:mace:shibboleth:3.0:native:sp:config"
>     xmlns:conf="urn:mace:shibboleth:3.0:native:sp:config"
>     xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
>     clockSkew="180">
> ```

eduID.at-registered Service Providers have a choice of two metadata resources – one including only ACOnet-registered IDPs and one also including IDPs known via Interfederation – depending on whether the Service Provider itself is accessible via Interfederation. In any case Service Providers should have proper access control rules in place, metadata filtering/limiting is not a substitute for that.

If in doubt, use the Interfederation-enabled metadata, so this doesn't have to be changed later on, once you decide to participate in Interfederation.

Service Providers only providing services to subjects associated with ACOnet participants can use this limited Metadata document, which only contains entities registered with ACOnet.

All IDPs registered with ACOnet

```
<MetadataProvider type="XML" url="https://eduid.at/md/aconet-registered.xml"
          backingFilePath="aconet-metadata.xml" reloadInterval="7200">
    <MetadataFilter type="RequireValidUntil" maxValidityInterval="2419200"/>
    <MetadataFilter type="Signature" certificate="aconet-metadata-signing.crt" verifyBackup="false"/>
    <MetadataFilter type="EntityRole">
        <RetainedRole>md:IDPSSODescriptor</RetainedRole>
        <RetainedRole>md:AttributeAuthorityDescriptor</RetainedRole>
    </MetadataFilter>
    <DiscoveryFilter type="Exclude" matcher="EntityAttributes" trimTags="true"
      attributeName="http://macedir.org/entity-category"
      attributeNameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
      attributeValue="http://refeds.org/category/hide-from-discovery" />
</MetadataProvider>
```

Service Providers not intenting to limit their potential audience to subjects from ACOnet participants will want to make use of the Interfederation-enabled Metadata document, which contains all eduID.at member institutions *as well as* any SAML entities known via Interfederation agreements (such as eduGAIN). Of course consuming Interfederation-enabled metadata only makes sense if your Service Provider is also exposed to Interfederation-enabled IDPs in other Federations.

All IDPs registered with ACOnet plus Interfederation IDPs

```
<MetadataProvider type="XML" url="https://eduid.at/md/aconet-interfed.xml"
          backingFilePath="aconet-metadata.xml" reloadInterval="7200">
    <MetadataFilter type="RequireValidUntil" maxValidityInterval="2419200"/>
    <MetadataFilter type="Signature" certificate="aconet-metadata-signing.crt" verifyBackup="false"/>
    <MetadataFilter type="EntityRole">
        <RetainedRole>md:IDPSSODescriptor</RetainedRole>
        <RetainedRole>md:AttributeAuthorityDescriptor</RetainedRole>
    </MetadataFilter>
    <DiscoveryFilter type="Exclude" matcher="EntityAttributes" trimTags="true"
      attributeName="http://macedir.org/entity-category"
      attributeNameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
      attributeValue="http://refeds.org/category/hide-from-discovery" />
</MetadataProvider>
```

Metadata filter examples

The Shibboleth SP software has powerful Metadata filtering capabilities which allow to restrict entities known to the SP via SAML metadata to only those matching certain criteria. Below are a couple of examples that are useful in specific situations. For convenience ("copy+paste") these are all self-contained but can be composed and arranged as specified in the documentation.

Campus-internal SPs (`Include` selected IDPs to remove all *others*)

Many institutions use Shibboleth and SAML also for internal "campus federation", i.e. with Service Providers *not* exposed to the eduID.at Federation but only known to the institutions' own IDP. Still these SPs will need trustworthy and current SAML metadata for the institutional IDP. The easiest and most secure way of achieving that is by pointing such SPs to the eduID.at federation Metadata *while also* adding a metadata `Include` filter, which effectively removes all *other* IDPs. E.g. only limiting an SP to the Vienna University SAML IDPs (production and test instances):

```
<MetadataProvider type="XML" url="https://eduid.at/md/aconet-registered.xml"
          validate="true" backingFilePath="aconet-metadata.xml" reloadInterval="14400">
    <MetadataFilter type="RequireValidUntil" maxValidityInterval="2419200"/>
    <MetadataFilter type="Signature" certificate="aconet-metadata-signing.crt" verifyBackup="false"/>
    <MetadataFilter type="Include">
        <Include>https://weblogin.univie.ac.at/shibboleth</Include>
        <Include>https://weblogin-test.univie.ac.at/shibboleth</Include>
    </MetadataFilter>
</MetadataProvider>
```

Of course the eduID.at certificate for signature validation needs to be downloaded/configured on these SPs just like on any other SP (i.e., those that are registered with the eduID.at federation) even though the SP itself is *not* exposed to the federation. That also means the institutional IDP needs a way to register those SPs, but usually a local XML file with SAML metadata managed directly at the IDP is sufficient for this.

Prevent all access from OpenIDP (`Exclude` selected IDPs)

If deployers of a Service Provider are certain they don't have a current (or future) use for identities provided by e.g. the ACOnet OpenIDP they can filter it out at the metadata level, preventing any logins from that IDP wholesale:

```
<MetadataProvider type="XML" url="https://eduid.at/md/aconet-registered.xml"
          validate="true" backingFilePath="aconet-metadata.xml" reloadInterval="14400">
    <MetadataFilter type="RequireValidUntil" maxValidityInterval="2419200"/>
    <MetadataFilter type="Signature" certificate="aconet-metadata-signing.crt" verifyBackup="false"/>
    <MetadataFilter type="Exclude">
        <Exclude>https://openidp.aco.net/saml</Exclude>
    </MetadataFilter>
</MetadataProvider>
```

Note that if you're using the Shibboleth EDS Discovery Service you could chose to only hide an entity in the discovery service, but keep it's metadata available to the Shibboleth SP. This allows continued use of the "hidden" IDP (e.g. for testing purposes) without showing the IDP publicly in the EDS interface. (Of course you'd need to provide other methods or documentation for the ones supposed to use such "hidden" IDPs.)

Hide IDPs that have asked not to be shown (Exclude IDP from Discovery)

Here's a combined example of how to hide two specific IDPs by name from the EDS, as well as hiding all IDPs tagged with the REFEDS Hide-from-Discovery category:

```
<MetadataProvider type="XML" url="https://eduid.at/md/aconet-registered.xml" ...>
    <MetadataFilter ...>
    <DiscoveryFilter type="Exclude" matcher="Name" Name="https://idp.example.org/shibboleth"/>
    <DiscoveryFilter type="Exclude" matcher="Name" Name="https://another.idp.example.org/shibboleth"/>
    <DiscoveryFilter type="Exclude" matcher="EntityAttributes"
      attributeName="http://macedir.org/entity-category"
      attributeNameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
      attributeValue="http://refeds.org/category/hide-from-discovery" />
</MetadataProvider>
```

> Note that a `DiscoveryFilter` will *not* prevent use of matching IDPs with the local SP! The only result of a discovery filter is filtering the output of the Shibboleth SP's "DiscoFeed" JSON resource that is used by the Shibboleth Embedded Discovery Service (but can be used by others as well, of course). If you want to prevent the SP from federating with certain IDPs (or groups of IDPs) – as some kind of initial coarse access control – use a `MetadataFilter` instead (like in the examples above) and/or properly implement authorization within the protected resource (webserver ACLs, application).

Attribute Mapping

Persistent NameIDs

By default the Shibboleth SP software will map persistent NameIDs to the internal attribute "persistent-id", whether they come from a `NameID` from the `Subject` element of the SAML Assertion, or from a NameID as value of the `eduPersonTargetedID` SAML Attribute. That's generally fine but sometimes an IDP may send a persistent NameID (ideally the exact some one) in both places at the same time. If that happens the Shibboleth SP will create a *multi-valued* internal "persistent-id" attribute, with values separated by a semicolon (";") – same as the SP does for all multi-valued attributes. There are several ways to deal with that case:

One way to deal with that is with additional code in your application: You'd split the attribute value on the semicolon and only use one of the values within your application. But then why use powerful middleware such as the Shibboleth SP if you still have to deal with these details within your application code? So let's use the Shibboleth SP software to deal with this in a better way that never creates multi-valued internal "persistent-id" attributes and never requires application code to get back a *single* identifier value.

A better way is to change the SP's attribute map and policy, to avoid those duplicated multi-valued "persistent-id" attributes. To do that first remove the following unneeded mapping from your `attribute-map.xml`:

First, remove (if it exists) the following rule for the internal "targeted-id" attribute from your `attribute-policy.xml` as we're about to change its content further below:

attribute-policy.xml: REMOVE all of this

```
<afp:AttributeRule attributeID="targeted-id">
    <afp:PermitValueRuleReference ref="ScopingRules"/>
</afp:AttributeRule>
```

Then also remove (if it exists) the corresponding rule for the SAML1 "targeted-id" attribute from your `attribute-map.xml` (you should never receive something like that anyway):

attribute-map.xml: REMOVE all of this

```
<Attribute name="urn:mace:dir:attribute-def:eduPersonTargetedID" id="targeted-id">
    <AttributeDecoder xsi:type="ScopedAttributeDecoder"/>
    <!-- <AttributeDecoder xsi:type="NameIDFromScopedAttributeDecoder"
formatter="$NameQualifier!$SPNameQualifier!$Name" defaultQualifiers="true"/> -->
</Attribute>
```

Next, change (only) the `id` XML attribute in the following defintion from "`persistent-id`" to "`targeted-id`", so that:

attribute-map.xml: CHANGE this section FROM...

```
<Attribute name="urn:oid:1.3.6.1.4.1.5923.1.1.1.10" id="persistent-id">
    <AttributeDecoder xsi:type="NameIDAttributeDecoder" formatter="$NameQualifier!$SPNameQualifier!$Name"
defaultQualifiers="true"/>
</Attribute>
```

becomes:

attribute-map.xml: CHANGE this section INTO

```
<Attribute name="urn:oid:1.3.6.1.4.1.5923.1.1.1.10" id="targeted-id">
    <AttributeDecoder xsi:type="NameIDAttributeDecoder" formatter="$NameQualifier!$SPNameQualifier!$Name"
defaultQualifiers="true"/>
</Attribute>
```

Finally, restart the Shibboleth SP software (e.g. `service shibd restart`) to activate the changed configuration. With this in place the following now happens:

- (Proper) Persistent NameIDs sent in the `Subject` element of the SAML Assertion will be mapped to the internal "persistent-id" attribute
- Persistent NameIDs sent as `eduPersonTargetedID` SAML Attribute values will be mapped to the internal "targeted-id" attribute
- IDPs sending *both* will therefore no longer create multi-valued "persistent-id" internal attributes!

To further isolate your application from all those different identifier attributes and NameIDs that are common use the Shibboleth SP's `REMOTE_USER` precedence list: Any (internal, after mapping) attribute names listed there are tried *in order* to populate the `REMOTE_USER` special CGI variable, using the first attribute that is not empty (i.e., that has a value). So all that's needed to support any kind of legacy (`eduPersonTargetedID`), past/present (persistent NameID) or present/future (pairwise-id, subject-id) identifiers is to make sure they're all listed in the Shibboleth SP's `REMOTE_USER` precedence list, e.g. by adding `targeted-id` (as re-defined above) to the end of the list:

For eduPersonTargetedID (legacy) support add it at the end of the REMOTE_USER list

```
<ApplicationDefaults entityID="..."
    REMOTE_USER="eppn subject-id pairwise-id persistent-id targeted-id"
    ... >
```

With that in place an application can simply just check for `REMOTE_USER` and will get the value of the first non-empty attribute listed there (in the order given), but will never receive more than one value.