Configure Shibboleth IdP to work with Amazon Web Services

1. Add the AWS relying party to **relying-party.xml** on your Shibboleth IdP (under the default relying party)

```
<rp:RelyingParty id="urn:amazon:webservices"
            provider="https://<yourIdP>/idp/shibboleth"
            defaultSigningCredentialRef="IdPCredential">
    <rp:ProfileConfiguration
                    xsi:type="saml:SAML2SSOProfile"
                    includeAttributeStatement="true"
                    assertionLifetime="PT5M" assertionProxyCount="0"
                    signResponses="never" signAssertions="always"
                    encryptAssertions="never" encryptNameIds="never"
                    includeConditionsNotBefore="true"
                    maximumSPSessionLifetime="PT1H" />
</rp:RelyingParty>
```

2. Add an extra metadata provider to your **relying-party.xml**

```
<metadata:MetadataProvider id="AWS" xsi:type="metadata:FileBackedHTTPMetadataProvider"
                      metadataURL="https://signin.aws.amazon.com/static/saml-metadata.xml"
                      backingFile="/path/to/shibboleth-idp/metadata/aws.xml" />
```

3. Ensure you have unsolicited login setup in your **handler.xml**

```
<ph:ProfileHandler xsi:type="ph:SAML2SSO"
     inboundBinding="urn:mace:shibboleth:2.0:profiles:AuthnRequest"
     outboundBindingEnumeration="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST-SimpleSign
                                 urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST
                                 urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Artifact">
     <ph:RequestPath>/SAML2/Unsolicited/SSO</ph:RequestPath>
 </ph:ProfileHandler>
```

4. Ensure you have unsolicited login setup in your **internal.xml**

(Underneath urn:mace:shibboleth:1.0:profiles:AuthnRequest)

```
<entry>
  <key>
    <value>urn:mace:shibboleth:2.0:profiles:AuthnRequest</value>
  </key>
  <bean id="shibboleth.UnsolicitedSSODecoder"
   class="edu.internet2.middleware.shibboleth.idp.profile.saml2.UnsolicitedSSODecoder">
    <constructor-arg ref="shibboleth.IdGenerator"/>
  </bean>
</entry>
```

5. Add an awsRoleSessionName attribute into **attribute-resolver.xml**

```
<resolver:AttributeDefinition id="awsRoleSessionName" xsi:type="ad:Simple"
sourceAttributeID="mail">
    <resolver:Dependency ref="mail"/>
    <resolver:AttributeEncoder
      xsi:type="enc:SAML2String"
      name="https://aws.amazon.com/SAML/Attributes/RoleSessionName"
      friendlyName="RoleSessionName" />
</resolver:AttributeDefinition>
```

This will be your session name/username on the AWS console (here, we've just used email address for simplicity).

6. Add a awsRoles attribute to **attribute-resolver.xml**

```
<resolver:AttributeDefinition id="awsRoles" xsi:type="ad:Mapped"
sourceAttributeID="memberOf">
  <resolver:Dependency ref="myLDAP"/>
  <resolver:AttributeEncoder
    xsi:type="enc:SAML2String"
    name="https://aws.amazon.com/SAML/Attributes/Role" friendlyName="Role" />
  <ad:ValueMap>
    <ad:ReturnValue>arn:aws:iam::AWSAccountID:saml-
provider/Shibboleth,arn:aws:iam::AWSAccountID:role/$1</ad:ReturnValue>
    <ad:SourceValue>CN=AWS_([^,]*),.*</ad:SourceValue>
  </ad:ValueMap>
</resolver:AttributeDefinition>
```

(Replacing the two instances of AWSAccountID with your account ID)

The example above finds all groups which the user is a member of, in the AD, whose name starts CN=AWS_. It takes the end of the CN as the source value, and ignores the rest of the DN, then maps this onto the return value string.

e.g. **CN=AWS_Admin,AWS_Groups,Web,Example,Org** returns **Admin** as the source value and outputs:

`arn:aws:iam::AWSAccountID:saml-provider/Shibboleth,arn:aws:iam::AWSAccountID:role/admin`

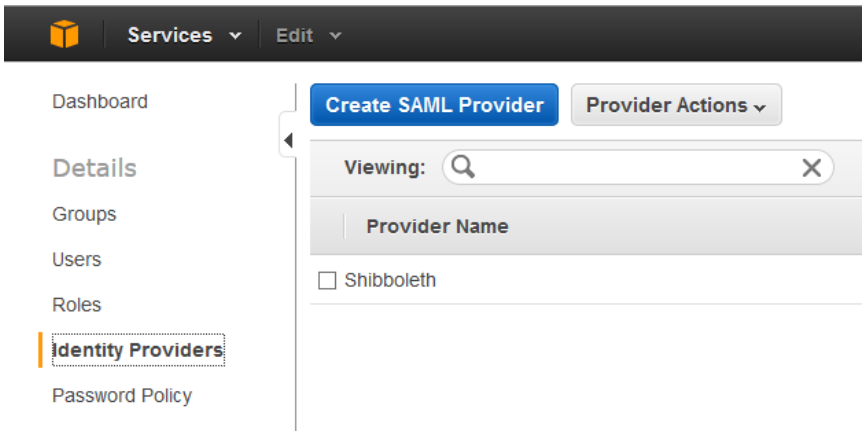
as the attribute value.

These role values need to match up exactly with the roles you'll define in Step 9 and the name after saml-provider/ (in this example "Shibboleth") needs to match the provider name you'll define in Step 8.

7. Release the amazon attributes to urn:amazon:webservices in **attribute-filter.xml**

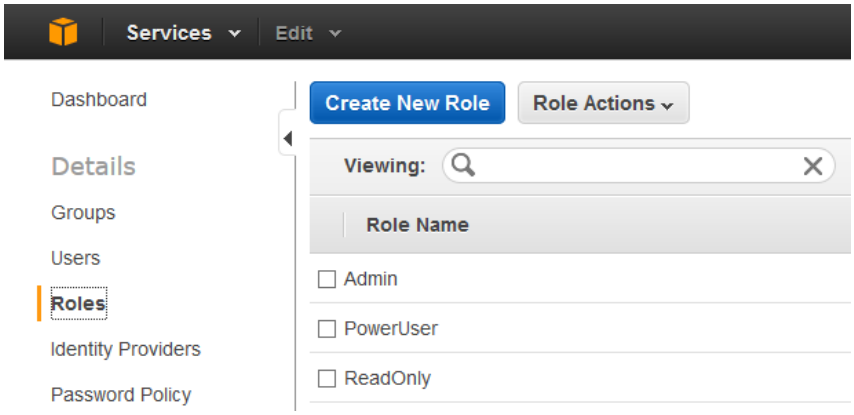
```
<afp:AttributeFilterPolicy>
  <afp:PolicyRequirementRule xsi:type="basic: AttributeRequesterString"
      value="urn:amazon:webservices" />
    <afp:AttributeRule attributeID="awsRoles">
      <afp:PermitValueRule xsi:type="basic:ANY"/>
    </afp:AttributeRule>
    <afp:AttributeRule attributeID="awsRoleSessionName">
      <afp:PermitValueRule xsi:type="basic:ANY"/>
    </afp:AttributeRule>
</afp:AttributeFilterPolicy>
```

8. Upload your **idp-metadata.xml** to the AWS Identity Providers, from your dashboard



Note: the provider name must match exactly the provider element of the attribute you defined in Step 6. So, if you added: `arn:aws:iam::AWSAccountID:saml-provider/Shibboleth` then your provider name in AWS must be `Shibboleth` (this is not the same as your IdPs entity ID, this remains unchanged).

9. Add roles matching the role attribute Source Values that will be generated from the code in Step 5



e.g. **`CN=AWS_Admin,AWS_Groups,Web,Example,Org`** returns **`Admin`** as the source value so the role would simply be called **`Admin`**

10. Browse to your IdP's unsolicited login URL, then (after logging in) select a role from those listed:

https://*<yourIdP>*/idp/profile/SAML2/Unsolicited/SSO?providerId=urn:amazon:webservices