属性・NameID

SPに対してどのような属性が送出されるか確認する方法

個別のページに移動

attribute-resolver.xmlやattribute-filter.xml等の設定を行ったあと、SPに対してどのような属性が送出されるか確認するためにはShibboleth IdP付属のaacli.shコマンドを利用することができます。

利用方法、及び出力結果の例は以下の通りです。

```
$ /opt/shibboleth-idp/bin/aacli.sh --principal="ユーザ名" --requester="属性送出を確認したいSPのentityID"

{
"requester": "SPのentityID",
"principal": "ユーザ名",
"attributes": [


  {
    "name": "eduPersonEntitlement",
    "values": [
              "StringAttributeValue{value=XXXXXXXXXXXXXXXXXX}"          ]
  },

  {
    "name": "eduPersonTargetedID",
    "values": [
              "XMLObjectAttributeValue{value=org.opensaml.saml.saml2.core.impl.NameIDImpl@b8728d3}"          ]
  },

  {
    "name": "displayName",
    "values": [
              "StringAttributeValue{value=XXXXXXXXXXXXXXXXXX}"          ]
  },

  {
    "name": "eduPersonPrincipalName",
    "values": [
              "ScopedStringAttributeValue{value=ユーザ名, scope=***.ac.jp}"          ]
  }

]
}
```

aacli.shコマンドの詳細は --help または -h オプションで確認するか、 https://wiki.shibboleth.net/confluence/display/SHIB2/AACLI をご参照ください。
Shibboleth WikiのIdentity Provider 3のスペースには詳細情報はありませんがShibboleth IdP 3でも従来通り使用できます。(Shibboleth Wiki: UpgradingFromV2の"Initial Testing"の項の記述参照)
実行で問題があるようでしたら トラブルシューティング の情報もご参照ください。(2016年9月現在はIdP v2のトラブルシューティングしか掲載されていません)

カスタマイズしたNameIDが送信されることを確認するためには、オプション --saml2 もしくは --saml1 を付けて実行してください。JSON形式でなくNameIDを含めた実際のアサーションのXML形式で出力されるようになります。

同様に、Encoderをカスタマイズして SAML 2.0でのみ、もしくはSAML 1.1のみで属性が送信されることを確認したい場合も、上記オプションをお使いください。

NameID設定

個別のページに移動

NameIDはconf/attribute-filter.xmlに記述しなくてもconf/saml-nameid.propertiesとconf/saml-nameid.xmlの設定により、SPメタデータの<NameID Format>に従って下記の通り送信します。

SPメタデータの<NameIDFormat>の値	送信する属性

urn:oasis:names:tc:SAML:2.0:nameid-format:transient	transient-id
urn:oasis:names:tc:SAML:2.0:nameid-format:persistent	persistent-id
<NameIDFormat>がない	saml-nameid.propertiesのidp.nameid.saml2.defaultに従う。 デフォルトはurn:oasis:names:tc:SAML:2.0:nameid-format:transient

SPメタデータに複数の<NameIDFormat>がある場合は、SPメタデータの並び順で送信可能な属性を送信します。persistent-idの設定を行っていないなど送信可能な属性がない場合は、//saml2:Subject/saml2:NameID自体が送信されません。

<NameIDFormat>がないSPの場合と<NameIDFormat>がurn:oasis:names:tc:SAML:2.0:nameid-format:persistentの場合の//saml2:Subject/saml2:NameIDの例を下記に示します。

- <NameIDFormat>がないSPの場合

```
<saml2:Subject>
    <saml2:NameID
        Format="urn:oasis:names:tc:SAML:2.0:nameid-format:transient"
        NameQualifier="https://idp.example.ac.jp/idp/shibboleth"
        SPNameQualifier="https://sp1.example.jp/shibboleth-sp">AAdzZWNyZXQxgUnobM3/AN3fn8DfZPDqBp
/GnKNxc5JR4nxXAxDAXZZSg0AZSrDh1Sip1fl9JGYrm2NWjl8zHKxHmbsgS/mFZ1ZlSYQ2U
/Kz7tCQ+SDswixwLRcGg3tDvVSAY8imKSrElGWSm5gMM45D4rkeQONJYr7gQZ13</saml2:NameID>
```

- <NameIDFormat>がurn:oasis:names:tc:SAML:2.0:nameid-format:persistentの場合

```
<saml2:Subject>
    <saml2:NameID
        Format="urn:oasis:names:tc:SAML:2.0:nameid-format:persistent"
        NameQualifier="https://idp.example.ac.jp/idp/shibboleth"
        SPNameQualifier="https://sp2.example.jp/shibboleth-sp">oiUiApwGnBP8pS3HZJ02ZW/aOTI=</saml2:NameID>
```

transient-idの設定

transient-idのデフォルトはCryptoTransientIdに変更になりました。CryptoTransientIdの使用例を下記に示します。

```
<saml2:Subject>
    <saml2:NameID
        Format="urn:oasis:names:tc:SAML:2.0:nameid-format:transient"
        NameQualifier="https://idp.example.ac.jp/idp/shibboleth"
        SPNameQualifier="https://sp1.example.jp/shibboleth-sp">AAdzZWNyZXQxgUnobM3/AN3fn8DfZPDqBp
/GnKNxc5JR4nxXAxDAXZZSg0AZSrDh1Sip1fl9JGYrm2NWjl8zHKxHmbsgS/mFZ1ZlSYQ2U
/Kz7tCQ+SDswixwLRcGg3tDvVSAY8imKSrElGWSm5gMM45D4rkeQONJYr7gQZ13</saml2:NameID>
```

IdP 2系と同じ短いtransient-idを使いたい場合は下記の変更を行います。

- conf/saml-nameid.properties
  idp.transientId.generatorをアンコメントして、値をshibboleth.StoredTransientIdGeneratorに変更します。

conf/saml-nameid.properties

```
# Set to shibboleth.StoredTransientIdGenerator for server-side transient ID storage
idp.transientId.generator = shibboleth.StoredTransientIdGenerator
```

差分

```
 # Set to shibboleth.StoredTransientIdGenerator for server-side transient ID storage
-#idp.transientId.generator = shibboleth.CryptoTransientIdGenerator
+idp.transientId.generator = shibboleth.StoredTransientIdGenerator
```

StoredTransientIdの使用例を下記に示します。

```
<saml2:Subject>
    <saml2:NameID
        Format="urn:oasis:names:tc:SAML:2.0:nameid-format:transient"
        NameQualifier="https://idp.example.ac.jp/idp/shibboleth"
        SPNameQualifier="https://sp1.example.jp/shibboleth-sp">_f358fb015b9b45c7d18a4a2647e79c33</saml2:NameID>
```

関連: [Shibboleth Wiki] Disable use of internal encryption key

persistent-idの設定

computedId

computedIdでの設定を下記に示します。

- conf/saml-nameid.xml
  <ref bean="shibboleth.SAML2PersistentGenerator" /> をアンコメントして有効にします。

conf/saml-nameid.xml

```
        <!-- Uncommenting this bean requires configuration in saml-nameid.properties. -->
        <!-- -->
        <ref bean="shibboleth.SAML2PersistentGenerator" />
        <!-- -->
```

差分

```
         <!-- Uncommenting this bean requires configuration in saml-nameid.properties. -->
-        <!--
+        <!-- -->
         <ref bean="shibboleth.SAML2PersistentGenerator" />
-        -->
+        <!-- -->
```

一部のSPにだけpersistent-idを送信したい場合、当該箇所をアンコメントせずに、以下を挿入すると対象SPを指定することができます。

```
        <bean parent="shibboleth.SAML2PersistentGenerator">
            <property name="activationCondition">
                <bean parent="shibboleth.Conditions.RelyingPartyId" c:candidates="#{{'https://test-sp1.gakunin.
nii.ac.jp/shibboleth-sp', 'https://test-sp2.gakunin.nii.ac.jp/shibboleth-sp'}}" />
            </property>
        </bean>
```

- conf/saml-nameid.properties
  idp.persistentId.generatorのデフォルトはComputedIdの設定のため、idp.persistentId.sourceAttributeとidp.persistentId.saltのみを設定します。idp.persistentId.saltには他人が推測できないランダムな値を指定してください。古いIdPから設定を引き継ぐ場合は同じ値を指定してください。

conf/saml-nameid.properties

```
# Persistent IDs can be computed on the fly with a hash, or managed in a database

# For computed IDs, set a source attribute and a secret salt:
idp.persistentId.sourceAttribute = uid
#idp.persistentId.useUnfilteredAttributes = true
# Do *NOT* share the salt with other people, it's like divulging your private key.
#idp.persistentId.algorithm = SHA
idp.persistentId.salt = XXXXXXXXXXXXXXXXXXXXXXXXXXX
```

差分

```
 # Persistent IDs can be computed on the fly with a hash, or managed in a database

 # For computed IDs, set a source attribute and a secret salt:
-#idp.persistentId.sourceAttribute = changethistosomethingreal
+idp.persistentId.sourceAttribute = uid
 #idp.persistentId.useUnfilteredAttributes = true
 # Do *NOT* share the salt with other people, it's like divulging your private key.
 #idp.persistentId.algorithm = SHA
-#idp.persistentId.salt = changethistosomethingrandom
+idp.persistentId.salt = XXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

- conf/attribute-resolver.xml
  idp.persistentId.sourceAttributeで指定した属性がLDAPで定義されているのみでconf/attribute-resolver.xmlの対応するresolver:AttributeDefinitionがコメントアウトされている場合は、当該resolver:AttributeDefinitionをアンコメントします。（以下はsourceAttributeとしてuidを指定した場合の例）

conf/attribute-resolver.xml

```
    <!-- Schema: Core schema attributes-->
    <!-- -->
    <resolver:AttributeDefinition xsi:type="ad:Simple" id="uid" sourceAttributeID="uid">
        <resolver:Dependency ref="myLDAP" />
        <resolver:AttributeEncoder xsi:type="enc:SAML1String" name="urn:mace:dir:attribute-def:uid" encodeType="
false" />
        <resolver:AttributeEncoder xsi:type="enc:SAML2String" name="urn:oid:0.9.2342.19200300.100.1.1" friendlyName="
uid" encodeType="false" />
    </resolver:AttributeDefinition>
    <!--
```

差分

```
     <!-- Schema: Core schema attributes-->
-    <!--
+    <!-- -->
     <resolver:AttributeDefinition xsi:type="ad:Simple" id="uid" sourceAttributeID="uid">
         <resolver:Dependency ref="myLDAP" />
         <resolver:AttributeEncoder xsi:type="enc:SAML1String" name="urn:mace:dir:attribute-def:uid" encodeType="
false" />
         <resolver:AttributeEncoder xsi:type="enc:SAML2String" name="urn:oid:0.9.2342.19200300.100.1.1" friendlyName="
uid" encodeType="false" />
     </resolver:AttributeDefinition>
+    <!--
```


他の用途に使用しない場合はresolver:AttributeEncoderの2行はコメントアウトしてかまいません。

Shibboleth IdP 3.1の情報

computedIdでの設定を下記に示します。

- conf/saml-nameid.xml
  <ref bean="shibboleth.SAML2PersistentGenerator" /> をアンコメントして有効にします。

conf/saml-nameid.xml

```
<!-- Uncommenting this bean requires configuration in saml-nameid.properties. -->
<!-- -->
<ref bean="shibboleth.SAML2PersistentGenerator" />
<!-- -->
```

- conf/saml-nameid.properties
  idp.persistentId.generatorのデフォルトはComputedIdの設定のため、idp.persistentId.sourceAttributeとidp.persistentId.saltのみを設定します。

conf/saml-nameid.properties

```
# Set to shibboleth.StoredPersistentIdGenerator for db-backed storage
# and uncomment/name the PersistentIdStore bean to use
#idp.persistentId.generator = shibboleth.ComputedPersistentIdGenerator

# Otherwise for computed PersistentIDs set the source attribute and salt.
idp.persistentId.sourceAttribute = uid4persistentId
idp.persistentId.salt = changethistosomethingrandom
```

- conf/attribute-resolver.xmlとconf/attribute-filter.xml
  idp.persistentId.sourceAttributeで指定した属性がLDAPで定義されているのみでconf/attribute-resolver.xmlのresolver:AttributeDefinitionで定義されていない場合は、PersistentIdGeneratorから参照できませんので以下のように定義し、conf/attribute-filter.xmlで送信設定を行います。他の用途に使用しない場合resolver:AttributeEncoderの2行は不要です。

conf/attribute-resolver.xml

```
<!-- ========================================== -->
<!--      PersistentId Definition               -->
<!-- ========================================== -->

<!-- Attribute Definition for %{idp.persistentId.sourceAttribute} -->
<resolver:AttributeDefinition id="%{idp.persistentId.sourceAttribute}" xsi:type="ad:Simple"
    sourceAttributeID="uid">
    <resolver:Dependency ref="myLDAP" />
</resolver:AttributeDefinition>
```

conf/attribute-filter.xml

```
<!--  Release to anyone -->
<afp:AttributeFilterPolicy id="PolicyforAnyone">
    <afp:PolicyRequirementRule xsi:type="basic:ANY" />

    <afp:AttributeRule attributeID="%{idp.persistentId.sourceAttribute}">
        <afp:PermitValueRule xsi:type="basic:ANY" />
    </afp:AttributeRule>

</afp:AttributeFilterPolicy>
```

- conf/intercept/consent-intercept-config.xml
  ユーザ同意画面にて%{idp.persistentId.sourceAttribute}を表示しないように、util:list[@id="shibboleth.consent.attribute-release.BlacklistedAttributeIDs"]に%{idp.persistentId.sourceAttribute}を追加します。

conf/intercept/consent-intercept-config.xml

```
<util:list id="shibboleth.consent.attribute-release.BlacklistedAttributeIDs">
    <value>transientId</value>
    <value>persistentId</value>
    <value>eduPersonTargetedID</value>
    <value>%{idp.persistentId.sourceAttribute}</value>
</util:list>
```

storedId

storedIdでの設定を下記に示します。

MySQL上にデータベース shibboleth が存在することを前提としております。また、MySQL Connector/J (mysql-connector-java-5.1.xx-bin.jar)をインストールしておいてください。

- conf/saml-nameid.xml
  <ref bean="shibboleth.SAML2PersistentGenerator" /> をアンコメントして有効にします。

conf/saml-nameid.xml

```
        <!-- Uncommenting this bean requires configuration in saml-nameid.properties. -->
        <!-- -->
        <ref bean="shibboleth.SAML2PersistentGenerator" />
        <!-- -->
```

差分

```
         <!-- Uncommenting this bean requires configuration in saml-nameid.properties. -->
-        <!--
+        <!-- -->
         <ref bean="shibboleth.SAML2PersistentGenerator" />
-        -->
+        <!-- -->
```

一部のSPにだけpersistent-idを送信したい場合、当該箇所をアンコメントせずに、以下を挿入すると対象SPを指定することができます。

```
        <bean parent="shibboleth.SAML2PersistentGenerator">
            <property name="activationCondition">
                <bean parent="shibboleth.Conditions.RelyingPartyId" c:candidates="#{{'https://test-sp1.gakunin.
nii.ac.jp/shibboleth-sp', 'https://test-sp2.gakunin.nii.ac.jp/shibboleth-sp'}}" />
            </property>
        </bean>
```

- conf/saml-nameid.properties
  idp.persistentId.sourceAttribute, idp.persistentId.salt, idp.persistentId.generatorとidp.persistentId.storeを設定します。idp.persistentId.saltには他人が推測できないランダムな値を指定してください。古いIdPから設定を引き継ぐ場合は同じ値を指定してください。

conf/saml-nameid.properties

```
# Persistent IDs can be computed on the fly with a hash, or managed in a database

# For computed IDs, set a source attribute and a secret salt:
idp.persistentId.sourceAttribute = uid
#idp.persistentId.useUnfilteredAttributes = true
# Do *NOT* share the salt with other people, it's like divulging your private key.
#idp.persistentId.algorithm = SHA
idp.persistentId.salt = XXXXXXXXXXXXXXXXXXXXXXXXXXX

# To use a database, use shibboleth.StoredPersistentIdGenerator
idp.persistentId.generator = shibboleth.StoredPersistentIdGenerator
# For basic use, set this to a JDBC DataSource bean name:
idp.persistentId.dataSource = MyDataSource
# For advanced use, set to a bean inherited from shibboleth.JDBCPersistentIdStore
#idp.persistentId.store = MyPersistentIdStore
# Set to an empty property to skip hash-based generation of first stored ID
#idp.persistentId.computed = shibboleth.ComputedPersistentIdGenerator
```

差分

```
 # Persistent IDs can be computed on the fly with a hash, or managed in a database

 # For computed IDs, set a source attribute and a secret salt:
-#idp.persistentId.sourceAttribute = changethistosomethingreal
+idp.persistentId.sourceAttribute = uid
 #idp.persistentId.useUnfilteredAttributes = true
 # Do *NOT* share the salt with other people, it's like divulging your private key.
 #idp.persistentId.algorithm = SHA
-#idp.persistentId.salt = changethistosomethingrandom
+idp.persistentId.salt = XXXXXXXXXXXXXXXXXXXXXXXXXXX

 # To use a database, use shibboleth.StoredPersistentIdGenerator
-#idp.persistentId.generator = shibboleth.ComputedPersistentIdGenerator
+idp.persistentId.generator = shibboleth.StoredPersistentIdGenerator
 # For basic use, set this to a JDBC DataSource bean name:
-#idp.persistentId.dataSource = PersistentIdDataSource
+idp.persistentId.dataSource = MyDataSource
 # For advanced use, set to a bean inherited from shibboleth.JDBCPersistentIdStore
 #idp.persistentId.store = MyPersistentIdStore
 # Set to an empty property to skip hash-based generation of first stored ID
 #idp.persistentId.computed = shibboleth.ComputedPersistentIdGenerator
```

- conf/attribute-resolver.xml
  idp.persistentId.sourceAttributeで指定した属性がLDAPで定義されているのみでconf/attribute-resolver.xmlの対応するresolver:AttributeDefinitionがコメントアウトされている場合、当該resolver:AttributeDefinitionをアンコメントします。（以下はsourceAttributeとしてuidを指定した場合の例）

conf/attribute-resolver.xml

```
    <!-- Schema: Core schema attributes-->
    <!-- -->
    <resolver:AttributeDefinition xsi:type="ad:Simple" id="uid" sourceAttributeID="uid">
        <resolver:Dependency ref="myLDAP" />
        <resolver:AttributeEncoder xsi:type="enc:SAML1String" name="urn:mace:dir:attribute-def:uid" encodeType="
false" />
        <resolver:AttributeEncoder xsi:type="enc:SAML2String" name="urn:oid:0.9.2342.19200300.100.1.1" friendlyName="
uid" encodeType="false" />
    </resolver:AttributeDefinition>
    <!--
```

差分

```
     <!-- Schema: Core schema attributes-->
-    <!--
+    <!-- -->
     <resolver:AttributeDefinition xsi:type="ad:Simple" id="uid" sourceAttributeID="uid">
         <resolver:Dependency ref="myLDAP" />
         <resolver:AttributeEncoder xsi:type="enc:SAML1String" name="urn:mace:dir:attribute-def:uid" encodeType="
false" />
         <resolver:AttributeEncoder xsi:type="enc:SAML2String" name="urn:oid:0.9.2342.19200300.100.1.1" friendlyName="
uid" encodeType="false" />
     </resolver:AttributeDefinition>
+    <!--
```


他の用途に使用しない場合はresolver:AttributeEncoderの2行はコメントアウトしてかまいません。

- shibpidテーブルの作成
  shibpidテーブルを作成します。

shibpid

```
CREATE TABLE shibpid (
    localEntity VARCHAR(255) NOT NULL,
    peerEntity VARCHAR(255) NOT NULL,
    persistentId VARCHAR(50) NOT NULL,
    principalName VARCHAR(50) NOT NULL,
    localId VARCHAR(50) NOT NULL,
    peerProvidedId VARCHAR(50) NULL,
    creationDate TIMESTAMP NOT NULL,
    deactivationDate TIMESTAMP NULL,
    PRIMARY KEY (localEntity, peerEntity, persistentId)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

- conf/global.xml
  conf/global.xmlでbean MyDataSourceを定義します。ユーザ同意の情報をMySQLに保存する設定もしくはuApproveJP等で設定済みの場合、重複となるためこの定義は不要です。

conf/global.xml

```
    <!-- Use this file to define any custom beans needed globally. -->

    <!-- A DataSource bean suitable for use in the idp.persistentId.dataSource property. -->
    <bean id="MyDataSource"
        class="org.apache.commons.dbcp2.BasicDataSource"
        p:driverClassName="com.mysql.jdbc.Driver"
        p:url="jdbc:mysql://localhost:3306/shibboleth"
        p:username="username"
        p:password="password"
        p:maxTotal="10"
        p:maxIdle="5"
        p:maxWaitMillis="15000"
        p:testOnBorrow="true"
        p:validationQuery="select 1"
        p:validationQueryTimeout="5" />
```

差分

```
     <!-- Use this file to define any custom beans needed globally. -->

+    <!-- A DataSource bean suitable for use in the idp.persistentId.dataSource property. -->
+    <bean id="MyDataSource"
+        class="org.apache.commons.dbcp2.BasicDataSource"
+        p:driverClassName="com.mysql.jdbc.Driver"
+        p:url="jdbc:mysql://localhost:3306/shibboleth"
+        p:username="username"
+        p:password="password"
+        p:maxTotal="10"
+        p:maxIdle="5"
+        p:maxWaitMillis="15000"
+        p:testOnBorrow="true"
+        p:validationQuery="select 1"
+        p:validationQueryTimeout="5" />
```

Shibboleth IdP 3.1の情報

- conf/saml-nameid.xml
  <ref bean="shibboleth.SAML2PersistentGenerator" />をアンコメントして有効にします。

conf/aml-nameid.xml

```
<!-- Uncommenting this bean requires configuration in saml-nameid.properties. -->
<!-- -->
<ref bean="shibboleth.SAML2PersistentGenerator" />
<!-- -->
```

- conf/saml-nameid.properties
  idp.persistentId.generator, idp.persistentId.store, idp.persistentId.sourceAttributeとidp.persistentId.saltを設定します。

conf/saml-nameid.properties

```
# Set to shibboleth.StoredPersistentIdGenerator for db-backed storage
# and uncomment/name the PersistentIdStore bean to use
idp.persistentId.generator = shibboleth.StoredPersistentIdGenerator
idp.persistentId.store = PersistentIdStore
# Set this to null to skip hash-based generation of first stored ID
#idp.persistentId.computed = shibboleth.ComputedPersistentIdGenerator

# Otherwise for computed PersistentIDs set the source attribute and salt.
idp.persistentId.sourceAttribute = uid4persistentId
idp.persistentId.salt = changethistosomethingrandom
```

- conf/global.xml
  idp.persistentId.storeの値をconf/global.xmlで定義します。

conf/global.xml (Tomcat 7の場合)

```
<!-- Use this file to define any custom beans needed globally. -->
<bean id="MyDataSource"
      class="org.apache.tomcat.dbcp.dbcp.BasicDataSource"
      p:driverClassName="com.mysql.jdbc.Driver"
      p:url="jdbc:mysql://localhost:3306/shibboleth"
      p:username="username"
      p:password="password"
      p:maxActive="10"
      p:maxIdle="5"
      p:maxWait="15000"
      p:testOnBorrow="true"
      p:validationQuery="select 1"
      p:validationQueryTimeout="5" />

<bean id="PersistentIdStore"
      class="net.shibboleth.idp.saml.nameid.impl.JDBCPersistentIdStore"
      p:dataSource-ref="MyDataSource" />
```

conf/global.xml (Tomcat 8の場合)

```
<!-- Use this file to define any custom beans needed globally. -->
<bean id="MyDataSource"
      class="org.apache.tomcat.dbcp.dbcp2.BasicDataSource"
      p:driverClassName="com.mysql.jdbc.Driver"
      p:url="jdbc:mysql://localhost:3306/shibboleth"
      p:username="username"
      p:password="password"
      p:maxIdle="5"
      p:maxTotal="10"
      p:maxWaitMillis="15000"
      p:testOnBorrow="true"
      p:validationQuery="select 1"
      p:validationQueryTimeout="5" />

<bean id="PersistentIdStore"
      class="net.shibboleth.idp.saml.nameid.impl.JDBCPersistentIdStore"
      p:dataSource-ref="MyDataSource" />
```

Tomcat 8付属のDBCP2から、p:maxActiveはp:maxTotalに、p:maxWaitはp:maxWaitMillisに変更になりました。

- conf/attribute-resolver.xmlとconf/attribute-filter.xml
  idp.persistentId.sourceAttributeで指定した属性がLDAPで定義されているのみでconf/attribute-resolver.xmlのresolver:AttributeDefinitionで定義されていない場合は、PersistentIdGeneratorから参照できませんので以下のように定義し、conf/attribute-filter.xmlで送信設定を行います。他の用途に使用しない場合resolver:AttributeEncoderの2行は不要です。

conf/attribute-resolver.xml

```
<!-- ========================================== -->
<!--      PersistentId Definition               -->
<!-- ========================================== -->

<!-- Attribute Definition for %{idp.persistentId.sourceAttribute} -->
<resolver:AttributeDefinition id="%{idp.persistentId.sourceAttribute}" xsi:type="ad:Simple"
    sourceAttributeID="uid">
    <resolver:Dependency ref="myLDAP" />
</resolver:AttributeDefinition>
```

conf/attribute-filter.xml

```
<!--  Release to anyone -->
<afp:AttributeFilterPolicy id="PolicyforAnyone">
    <afp:PolicyRequirementRule xsi:type="basic:ANY" />

    <afp:AttributeRule attributeID="%{idp.persistentId.sourceAttribute}">
        <afp:PermitValueRule xsi:type="basic:ANY" />
    </afp:AttributeRule>

</afp:AttributeFilterPolicy>
```

- conf/intercept/consent-intercept-config.xml
  ユーザ同意画面にて%{idp.persistentId.sourceAttribute}を表示しないように、util:list[@id="shibboleth.consent.attribute-release.BlacklistedAttributeIDs"]に%{idp.persistentId.sourceAttribute}を追加します。

conf/intercept/consent-intercept-config.xml

```
<util:list id="shibboleth.consent.attribute-release.BlacklistedAttributeIDs">
    <value>transientId</value>
    <value>persistentId</value>
    <value>eduPersonTargetedID</value>
    <value>%{idp.persistentId.sourceAttribute}</value>
</util:list>
```

eduPersonTargetedID属性の送信

NameIDとは別に//saml2:AttributeStatement/saml2:Attribute[@FriendlyName="eduPersonTargetedID"]としてeduPersonTargetedID属性を送信する設定は下記の通りです。

computedId

computedIdでの設定を下記に示します。persistent-idの設定をあらかじめ実行しておくことが前提で、定義されたconf/saml-nameid.propertiesのプロパティを参照しています。

- conf/attribute-resolver.xml

conf/attribute-resolver.xml

```
<!-- ========================================== -->
<!--      Attribute Definitions                 -->
<!-- ========================================== -->

<!-- Schema: eduPerson attributes -->

<!-- Attribute Definition for eduPersonTargetedID -->
<resolver:AttributeDefinition id="eduPersonTargetedID" xsi:type="SAML2NameID" xmlns="urn:mace:shibboleth:2.0:resolver:
ad"
    nameIdFormat="urn:oasis:names:tc:SAML:2.0:nameid-format:persistent"
    sourceAttributeID="computedID">
    <resolver:Dependency ref="computedID" />
    <resolver:AttributeEncoder xsi:type="SAML1XMLObject" xmlns="urn:mace:shibboleth:2.0:attribute:encoder"
            name="urn:oid:1.3.6.1.4.1.5923.1.1.1.10" />
    <resolver:AttributeEncoder xsi:type="SAML2XMLObject" xmlns="urn:mace:shibboleth:2.0:attribute:encoder"
            name="urn:oid:1.3.6.1.4.1.5923.1.1.1.10" friendlyName="eduPersonTargetedID" />
</resolver:AttributeDefinition>


<!-- ========================================== -->
<!--      Data Connectors                       -->
<!-- ========================================== -->

<!-- Computed targeted ID connector -->
<resolver:DataConnector xsi:type="ComputedId" xmlns="urn:mace:shibboleth:2.0:resolver:dc"
                        id="computedID"
                        generatedAttributeID="computedID"
                        sourceAttributeID="%{idp.persistentId.sourceAttribute}"
                        salt="%{idp.persistentId.salt}">
    <resolver:Dependency ref="%{idp.persistentId.sourceAttribute}" />
</resolver:DataConnector>
```

- conf/attribute-filter.xmlの例

conf/attribute-filter.xml

```
<!--  Release to sp.example.jp -->
<afp:AttributeFilterPolicy id="PolicyforSP1ExampleJP">
    <afp:PolicyRequirementRule xsi:type="basic:AttributeRequesterString" value="https://sp.example.jp/shibboleth-sp"
/>
    <afp:AttributeRule attributeID="eduPersonTargetedID">
        <afp:PermitValueRule xsi:type="basic:ANY" />
    </afp:AttributeRule>
</afp:AttributeFilterPolicy>
```

storedId

storedIdでの設定を下記に示します。persistent-idの設定をあらかじめ実行しておくことが前提で、定義されたconf/global.xmlのbean MyDataSourceとconf/saml-nameid.propertiesのプロパティを参照します。

- conf/attribute-resolver.xml

conf/attribute-resolver.xml

```
<!-- ========================================== -->
<!--      Attribute Definitions                 -->
<!-- ========================================== -->

<!-- Schema: eduPerson attributes -->

<!-- Attribute Definition for eduPersonTargetedID -->
<resolver:AttributeDefinition id="eduPersonTargetedID" xsi:type="SAML2NameID" xmlns="urn:mace:shibboleth:2.0:resolver:
ad"
    nameIdFormat="urn:oasis:names:tc:SAML:2.0:nameid-format:persistent"
    sourceAttributeID="storedID">
    <resolver:Dependency ref="storedID" />
    <resolver:AttributeEncoder xsi:type="SAML1XMLObject" xmlns="urn:mace:shibboleth:2.0:attribute:encoder"
            name="urn:oid:1.3.6.1.4.1.5923.1.1.1.10" />
    <resolver:AttributeEncoder xsi:type="SAML2XMLObject" xmlns="urn:mace:shibboleth:2.0:attribute:encoder"
            name="urn:oid:1.3.6.1.4.1.5923.1.1.1.10" friendlyName="eduPersonTargetedID" />
</resolver:AttributeDefinition>


<!-- ========================================== -->
<!--      Data Connectors                       -->
<!-- ========================================== -->

<!-- Stored targeted ID connector -->
<resolver:DataConnector xsi:type="StoredId" xmlns="urn:mace:shibboleth:2.0:resolver:dc"
                        id="storedID"
                        generatedAttributeID="storedID"
                        sourceAttributeID="%{idp.persistentId.sourceAttribute}"
                        salt="%{idp.persistentId.salt}">
    <resolver:Dependency ref="%{idp.persistentId.sourceAttribute}" />
    <BeanManagedConnection>MyDataSource</BeanManagedConnection>
</resolver:DataConnector>
```

- conf/attribute-filter.xmlの例

conf/attribute-filter.xml

```
<!--  Release to sp.example.jp -->
<afp:AttributeFilterPolicy id="PolicyforSP1ExampleJP">
    <afp:PolicyRequirementRule xsi:type="basic:AttributeRequesterString" value="https://sp.example.jp/shibboleth-sp"
/>
    <afp:AttributeRule attributeID="eduPersonTargetedID">
        <afp:PermitValueRule xsi:type="basic:ANY" />
    </afp:AttributeRule>
</afp:AttributeFilterPolicy>
```

同じ値が再割り当てされないeduPersonTargetedIDの生成方法

個別のページに移動

eduPersonTargetedID(ePTID)を生成するときにはLDAP上の属性としてuidを代表とした属性値が利用されますが、これらの属性値では再割り当ての問題があります。

例えばuidとして test001 を使っていた人が異動になり、さらに年月が経って同じuidを使いたいという人が現われた場合にはそのまま割り当ててしまうことはできません。これはSP 側で uid=test001 という属性値を基に生成されたePTIDで個人を識別していた場合に、再割り当て前の人物と、再割り当て後の人物を区別できず同一人物とみなして再割り当て前のアカウントの情報を利用してしまうこと（当人が意図しないなりすまし）が起こるためです。

ePTIDでStoredIDを利用している場合には失効処理を行うことで新しいePTIDを生成できることから、再割り当てされる属性値をソースとした上で再割り当て時に失効することでも対処可能です。今回は別の方法として、LDAP上のuidの付加情報としてLDAPエントリの作成時間(createTimestamp)を加えた値をソースとしてePTIDを生成する方法を紹介します（ComputedIDをベースに設定方法を紹介していますが、StoredIDの場合も同様です）。

例えば <uid>-<createTimestamp> のように2つの属性値をハイフンでつなげて test001-20130314110740Z といった値をソースとしてePTIDを生成すれば、uid再割り当てごとの失効処理が不要となります。ただし、再割り当ての際に必ず「作成時間」が変更されるようにアカウント作成処理をすることが前提となります。（LDAPエントリを再利用するような運用ではcreateTimestampが変更されない可能性があります）

- eduPersonTargetedIDのAttributeDefinitionはデフォルトのままで利用可能です。

/opt/shibboleth-idp/conf/attribute-resolver.xml の設定

```
    <!-- Attribute Definition for eduPersonTargetedID (computedID) -->
    <resolver:AttributeDefinition xsi:type="ad:SAML2NameID" id="eduPersonTargetedID"
                                  nameIdFormat="urn:oasis:names:tc:SAML:2.0:nameid-format:persistent" sourceAttributeID="
computedID">
        <resolver:Dependency ref="computedID" />
        <resolver:AttributeEncoder xsi:type="enc:SAML1XMLObject"
                                   name="urn:oid:1.3.6.1.4.1.5923.1.1.1.10" encodeType="false" />
        <resolver:AttributeEncoder xsi:type="enc:SAML2XMLObject"
                                   name="urn:oid:1.3.6.1.4.1.5923.1.1.1.10" friendlyName="eduPersonTargetedID"
encodeType="false" />
    </resolver:AttributeDefinition>
```

- Template Attribute Definitionで uid-createTimestamp の文字列を返すAttributeDefinitionを定義して、ComputedID用DataConnectorのsourceAttributeID, Dependencyで参照できるようにします。ここでは「templateePTID」という名前を用います。

/opt/shibboleth-idp/conf/attribute-resolver.xml の設定

```
    <!-- Computed targeted ID connector -->
    <resolver:DataConnector id="computedID" xsi:type="dc:ComputedId"
                            generatedAttributeID="computedID"
                            sourceAttributeID="%{idp.persistentId.sourceAttribute}"
                            salt="%{idp.persistentId.salt}">
        <resolver:Dependency ref="%{idp.persistentId.sourceAttribute}" />
    </resolver:DataConnector>
    ↓以下の行を追加
    <resolver:AttributeDefinition id="templateePTID" xsi:type="Template" xmlns="urn:mace:shibboleth:2.0:resolver:ad">
        <resolver:Dependency ref="myLDAP" />

        <Template>
            <![CDATA[
                ${uid}-${createTimestamp}
            ]]>
        </Template>

        <SourceAttribute>uid</SourceAttribute>
        <SourceAttribute>createTimestamp</SourceAttribute>
    </resolver:AttributeDefinition>
```

/opt/shibboleth-idp/conf/saml-nameid.properties の設定

```
# For computed IDs, set a source attribute and a secret salt:
idp.persistentId.sourceAttribute = templateePTID   ← 変更
```

- LDAPから追加でcreateTimestampを取得するためにLDAP DataConnectorにReturnAttributesを定義します。

/opt/shibboleth-idp/conf/attribute-resolver.xml の設定

```
    <resolver:DataConnector id="myLDAP" xsi:type="dc:LDAPDirectory"
        ldapURL="%{idp.attribute.resolver.LDAP.ldapURL}"
        baseDN="%{idp.attribute.resolver.LDAP.baseDN}"
        principal="%{idp.attribute.resolver.LDAP.bindDN}"
        principalCredential="%{idp.attribute.resolver.LDAP.bindDNCredential}"
        useStartTLS="%{idp.attribute.resolver.LDAP.useStartTLS:true}">
        <dc:FilterTemplate>
            <![CDATA[
                %{idp.attribute.resolver.LDAP.searchFilter}
            ]]>
        </dc:FilterTemplate>
        <dc:ReturnAttributes>* createTimestamp</dc:ReturnAttributes>   ← 追加(dc:FilterTemplateの直後である必要があります)
    </resolver:DataConnector>
```

SAML 2 persistent IDでのAttribute Queryの許可

⚠ この機能はcomputedIdを使っている場合は使用できません。まずstoredIdを使うように設定変更してください。

- conf/c14n/subject-c14n.xml
  conf/c14n/subject-c14n.xmlの<ref bean="c14n/SAML2Persistent" />をアンコメントします。

conf/c14n/subject-c14n.xml

```
<!-- Handle a SAML 2 persistent ID, provided a stored strategy is in use. -->
<ref bean="c14n/SAML2Persistent" />
```

差分

```
        <!-- Handle a SAML 2 persistent ID, provided a stored strategy is in use. -->
-       <!-- <ref bean="c14n/SAML2Persistent" /> -->
+       <ref bean="c14n/SAML2Persistent" />
```

SAML1でフロントチャネルにAttributeStatementを含める設定

以下は全てのSPに対して適用する方法です。特定のSPに対してのみ適用する場合は、そのSP用のbeanをRelyingPartyOverridesに作成し、プロファイルに以下の設定を行ってください。

- conf/relying-party.xml
  bean[@parent="Shibboleth.SSO"]にp:includeAttributeStatement="true"を追加します。

conf/relying-party.xml

```
<bean id="shibboleth.DefaultRelyingParty" parent="RelyingParty">
    <property name="profileConfigurations">
       <list>
            <bean parent="Shibboleth.SSO" p:postAuthenticationFlows="attribute-release" p:includeAttributeStatement="
true" />
            <ref bean="SAML1.AttributeQuery" />
            <ref bean="SAML1.ArtifactResolution" />
            <bean parent="SAML2.SSO" p:postAuthenticationFlows="attribute-release" />
            <ref bean="SAML2.ECP" />
            <ref bean="SAML2.Logout" />
            <ref bean="SAML2.AttributeQuery" />
            <ref bean="SAML2.ArtifactResolution" />
            <ref bean="Liberty.SSOS" />
        </list>
    </property>
</bean>
```

差分

```
     <bean id="shibboleth.DefaultRelyingParty" parent="RelyingParty">
        <property name="profileConfigurations">
            <list>
-               <bean parent="Shibboleth.SSO" p:postAuthenticationFlows="attribute-release" />
+               <bean parent="Shibboleth.SSO" p:postAuthenticationFlows="attribute-release" p:includeAttributeStatement="
true" />
                <ref bean="SAML1.AttributeQuery" />
                <ref bean="SAML1.ArtifactResolution" />
                <bean parent="SAML2.SSO" p:postAuthenticationFlows="attribute-release" />
                <ref bean="SAML2.ECP" />
                <ref bean="SAML2.Logout" />
                <ref bean="SAML2.AttributeQuery" />
                <ref bean="SAML2.ArtifactResolution" />
                <ref bean="Liberty.SSOS" />
            </list>
        </property>
     </bean>
```

特定のSPへのアサーションを暗号化しない設定

- conf/relying-party.xml
  以下のようにshibboleth.RelyingPartyOverridesの子要素として当該SP向けの設定を追加してください。
  ・SPのentityIDは適切なものに置き換えてください。
  ・変化を最小限にするため基本的には設定は同ファイルのshibboleth.DefaultRelyingPartyの設定と同じくし、SAML2.SSOにp:encryptAssertions="false"を追加してください。DefaultRelyingPartyにある他のbeanも必要ならコピー&ペーストしてください。
  ・他にもRelyingPartyOverridesの子要素があり当該SPが他のoverrideにすでに記述されている場合、マージは行われませんので同等の設定になるように1つにまとめるようにしてください。

conf/relying-party.xml

```
    <!-- Container for any overrides you want to add. -->

    <util:list id="shibboleth.RelyingPartyOverrides">

        ...

        <bean p:id="example.NoEncryptAssertions" parent="RelyingPartyByName">
            <constructor-arg name="relyingPartyIds">
                <list>
                    <value>https://sp.example.ac.jp/shibboleth-sp</value>
                </list>
            </constructor-arg>
            <property name="profileConfigurations">
                <list>
                    <bean parent="SAML2.SSO" p:postAuthenticationFlows="attribute-release" p:encryptAssertions="false" />
                    <ref bean="SAML2.Logout" />
                </list>
            </property>
        </bean>

    </util:list>
```

差分

```
     <!-- Container for any overrides you want to add. -->

     <util:list id="shibboleth.RelyingPartyOverrides">

         ...

+        <bean p:id="example.NoEncryptAssertions" parent="RelyingPartyByName">
+            <constructor-arg name="relyingPartyIds">
+                <list>
+                    <value>https://sp.example.ac.jp/shibboleth-sp</value>
+                </list>
+            </constructor-arg>
+            <property name="profileConfigurations">
+                <list>
+                    <bean parent="SAML2.SSO" p:postAuthenticationFlows="attribute-release" p:encryptAssertions="false" />
+                    <ref bean="SAML2.Logout" />
+                </list>
+            </property>
+        </bean>
+
     </util:list>
```

LDAPにエントリがない場合にIdP上でエラーにする方法

個別のページに移動

概要

Shibboleth IdPにおいて、属性の生成手段としてLDAPサーバを参照する方法がありますが、LDAPサーバにエントリが存在しない場合の挙動について注意が必要です。Shibboleth IdPのデフォルト設定における挙動は以下の通りです。

- LDAPにエントリーが存在しない場合も処理は中断しない

- SPに向けて空の属性が送信される

SP側に属性を送信する前にIdP上でエラーにするには、ここで紹介する設定が必要になります。同時に、Template AttributeDefinitionなど他の要因でエラーが発生した場合もIdP上でエラーとなります。現在の運用でエラーが発生していないことをログにより確認しておいてください。

前提条件

- Shibboleth IdP 3.3.x向けテンプレート(3.3.0以降)のattribute-resolver.xmlを使用していること
  LDAP DataConnectorに以下のような設定が含まれます。

```
        noResultIsError="%{idp.attribute.resolver.LDAP.noResultIsError:true}"
```

設定方法

- services.properties

  idp.service.attribute.resolver.maskFailuresの値をfalseに変更します。

conf/services.properties

```
idp.service.attribute.resolver.maskFailures = false
```

差分

```
 #idp.service.attribute.resolver.resources = shibboleth.AttributeResolverResources
 #idp.service.attribute.resolver.failFast = false
 idp.service.attribute.resolver.checkInterval = PT15M
-#idp.service.attribute.resolver.maskFailures = true
+idp.service.attribute.resolver.maskFailures = false

 #idp.service.attribute.filter.resources = shibboleth.AttributeFilterResources
 # NOTE: Failing the filter fast leaves no filters enabled.
```

- errors.xml

  IdP上でエラーとするため、<util:map id="shibboleth.LocalEventMap">の子要素として以下の要素を追加します。

conf/services.properties

```
<entry key="UnableToResolveAttributes" value="true"/>
```

差分

```
    <util:map id="shibboleth.LocalEventMap">
        <entry key="ContextCheckDenied" value="true" />
        <entry key="AttributeReleaseRejected" value="true" />
        <entry key="TermsRejected" value="true" />
        <entry key="RuntimeException" value="false" />
+       <entry key="UnableToResolveAttributes" value="true"/>
        <!--
        <entry key="IdentitySwitch" value="false" />
        <entry key="NoPotentialFlow" value="false" />
        -->
    </util:map>
```

確認方法

aacli.shで設定が正しいか確認することができます。

- LDAPにエントリ(ユーザ)が存在しない場合

```
$ /opt/shibboleth-idp/bin/aacli.sh -n user1 -r https://sp.example.ac.jp/shibboleth-sp
    { "error": "UnableToResolveAttributes" }
```

- LDAPにエントリ(ユーザ)が存在する場合

```
$ /opt/shibboleth-idp/bin/aacli.sh -n user1 -r https://sp.example.ac.jp/shibboleth-sp
{
"requester": "https://sp.example.ac.jp/shibboleth-sp",
"principal": "user1",
"attributes": [


  {
    "name": "eduPersonPrincipalName",
    "values": [
              "ScopedStringAttributeValue{value=user1, scope=example.ac.jp}"          ]
  }

]
}
```