# IDP2 USI Wien eduPersonEntitlement

**Legacy documentation!**

This is old documentation for the legacy Shibboleth IDP v2! Use the current documentation for IDP v3 instead!

Voll funktionales *Beispiel* einer Konfiguration für den Shibboleth 2.4.x IDP (und Java <= 7), um das gewünschte USI Wien-eduPersonEntitlement zu berechnen, mittels dessen Studierende unter 25 Jahre ohne zusätzliche Umstände vergünstigte Tarife bei Kursen des USI Wien in Anspruch nehmen können. Dieses etwas komplexere Vorgehen verhindert, daß das Alter oder Geburtsdatum von Personen an den Service Provider geschickt werden muss. Gleichzeitig dient es als Beispiel für die Erzeugung bzw. komplexe Manipulation von Attributen im Shibboleth IDP.

Seit der Übertragung aus dem alten AAI-Wiki (und dadurch bedingter Überarbeitung) wurde dieses Beispiel nicht getestet. Bitte ggfs. um Rückmeldung.

## LDAP oder RDBMS

Zur Berechnung des jeweils aktuellen Alters wird klarweise das Geburtsdatum einer Person benötigt. Hat man dafür noch kein eigenes Attribut im LDAP-Verzeichnisdienst vorgesehen, bietet sich schacDateOfBirth aus dem SCHAC-Schema an. Dazu muß das Schema dem LDAP DSA bekannt gemacht, sowie allen Objekte, denen ein schacDateOfBirth zugewiesen werden soll, die "auxiliary objectClass" `schacPersonalCharacteristics` zugewiesen werden. Wie im SCHAC-Schema angegeben, wird das Datum im Format `JJJJMMTT` gespeichert, der 1. April 1970 würde also etwa zu 19700401.

## IDP-Konfiguration

### attribute-resolver.xml

Ergänzung eines `DataConnectors`, um das Geburtsdatum einer Person mit abzufragen. Wird dazu ein LDAP-Verzeichnisdienst genutzt **und** werden die abgefragten LDAP-Attribute eigens in einem <ReturnAttributes>-Element aufgelistet, (nur dann) muß diese Liste um `schacDateOfBirth` erweitert werden, z.B.:

**Gegebenfalls dc:ReturnAttributes ergänzen**

```
<resolver:DataConnector id="myLDAP" xsi:type="dc:LDAPDirectory"
    ldapURL="ldap://ldap.example.edu" baseDN="dc=example,dc=edu" principal="cn=idp,ou=accounts,dc=example,
dc=edu"
    principalCredential="somePassword" useStartTLS="true">
    <dc:FilterTemplate>
        <![CDATA[
            (uid=$requestContext.principalName)
        ]]>
    </dc:FilterTemplate>
    <dc:ReturnAttributes>uid mail sn givenname eduPersonAffiliation schacDateOfBirth</dc:ReturnAttributes>
</resolver:DataConnector>
```

Damit sollte das Geburtsdatum im IDP prinzipiell verfügbar sein. Um daraus das Alter zu errechnen, braucht es etwas eigenen Code, über eine Script Attribute Definition des Shiboleth IDP. Damit wird die Integration von dynamischen Programmiersprachen wie ECMAScript/JavaScript mit den Java-Objekten des IDP möglich, ohne komplexe Erweitungen für den IDP in Java schreiben zu müssen.

Für eine Script-AttributeDefinition *muß* die interne id der `AttributeDefinition` (in der IDP-Konfiguration) gleich dem Attributnamen lauten, der im enthaltenden Script selbst definiert wird, via `new BasicAttribute()` o.ä.!

Weiters muß (wie üblich beim Shibboleth IDP) jedes Element innerhalb der Konfiguration eine eindeutige `id` haben. D.h. man kann nicht einfach der "Simple" AttributeDefinition für `eduPersonEntitlement` aus der IDP-Standardkonfiguration eine Abhängigkeit auf diese neue Script-AttributeDefinition eintragen. Die "Simple" müsste eine andere `id` bekommen.

Im weiter unten angegebenen Beispiel erzeugen wir `eduPersonEntitlement`-Werte nur aus den Berechnungen von `schacDateOfBirth`-Werten und nur für den Zweck des USI Wien-Services. Sollte hingegen der IDP *auch* andere Entitlements erzeugen können (etwa auch Werte aus dem LDAP nachschlagen, o.ä.), ist es am einfachsten, eine "Simple" AttributeDefinition mit *anderer* `id` (hier etwa `id="entitlements"`) zu erstellen, und in dieser Abhängigkeiten auf "eduPersonEntitlement" (aus LDAP und der Script-AttibuteDefinition) einzutragen. In der Script-`AttributeDefinition` könnten dann die AttributeEncoder-Elemente entfallen sowie ein `dependencyOnly="true"` Attribut ergänzt werden. Nur das Attribut mit der `id="entitlements"` würde dann ja in der Datei `attribute-filter.xml` freigegeben werden, und es enthält dann die Vereinigungsmenge aller Werte aus den Abhängigkeiten:

**eduPersonEntitlements aus untenstehendem Script und LDAP erzeugen**

```
<resolver:AttributeDefinition xsi:type="ad:Simple" id="entitlements" sourceAttributeID="eduPersonEntitlement">
    <resolver:Dependency ref="eduPersonEntitlement" />
    <resolver:Dependency ref="myLDAP" />
    <resolver:AttributeEncoder xsi:type="enc:SAML1String" name="urn:mace:dir:attribute-def:
eduPersonEntitlement" />
    <resolver:AttributeEncoder xsi:type="enc:SAML2String" name="urn:oid:1.3.6.1.4.1.5923.1.1.1.7" friendlyName="
eduPersonEntitlement" />
</resolver:AttributeDefinition>
```

Hier nun das Beispiel für den einfacheren Fall: Werden eduPersonEntitlements nur für USI Wien gebraucht, *reicht nur diese eine AttributeDefinition alleine aus*!

**eduPersonEntitlements nur aus diesem Script erzeugen**

```
<resolver:AttributeDefinition xsi:type="ad:Script" id="eduPersonEntitlement">
    <resolver:Dependency ref="myLDAP" />
    <resolver:AttributeEncoder xsi:type="enc:SAML1String" name="urn:mace:dir:attribute-def:
eduPersonEntitlement" />
    <resolver:AttributeEncoder xsi:type="enc:SAML2String" name="urn:oid:1.3.6.1.4.1.5923.1.1.1.7" friendlyName="
eduPersonEntitlement" />
    <ad:Script><![CDATA[
importPackage(Packages.edu.internet2.middleware.shibboleth.common.attribute.provider);
importPackage(Packages.org.slf4j);
importPackage(Packages.org.joda.time);
var logger = LoggerFactory.getLogger("edu.internet2.middleware.shibboleth.resolver.Script.usidiscount");
var tz = DateTimeZone.UTC;

if (eduPersonEntitlement == null) {
    var eduPersonEntitlement = new BasicAttribute("eduPersonEntitlement");
}

if (typeof schacDateOfBirth != "undefined" && schacDateOfBirth != null) {
    var dob = schacDateOfBirth.getValues().get(0);
    logger.trace("schacDateOfBirth is " + dob);
    var birth_date = new DateMidnight(dob.slice(0,4), dob.slice(4,6), dob.slice(6,8), tz);
    logger.trace("birth_date is " + birth_date.toYearMonthDay());
    var age = new Years.yearsBetween(birth_date, DateTime(tz)).getYears();
    logger.trace("age is " + age);

    if (age < 25) {
        logger.debug("age within USI limits, adding usi-entitlement");
        logger.trace("eduPersonEntitlement has values: " + eduPersonEntitlement.getValues());
        eduPersonEntitlement.getValues().add("http://usi.at/student-discount");
        logger.trace("eduPersonEntitlement has values: " + eduPersonEntitlement.getValues());
    } else {
        logger.debug("age not within USI limits, doing nothing");
    }
} else {
    logger.debug("no schacDateOfBirth, ignoring");
}
]]></ad:Script>
</resolver:AttributeDefinition>
```

Hier wird mit JavaScript (der Standard-Scripting Engine) ein Attribut namens `eduPersonEntitlement` definiert. Ausgangsbasis für dessen Werte ist hier das Attribut `schacDateOfBirth`, das aus einem LDAP-Verzeichnisdienst gelesen wird (Dependency "`myLDAP`"). Ist `schacDateOfBirth` definiert (weil die LDAP-Abfrage einen Wert zurückgeliefert hat), wird der Wert zerlegt und in ein Datumsobjekt `birth_date` verwandelt. Die Differenz in Jahren zwischen diesem Objekt und dem heutigen Tag `DateTime()` ergibt das aktuelle Alter der zugreifenden Person.

Ist nun das Alter kleiner 25, wird ein Entitlement spezifisch für USI Wien erzeugt. Ist das Alter größer oder gleich 25 oder gab es kein `schacDateOfBirth` in der LDAP-Abfrage, wird kein Wert erzeugt.

## logging.xml

Um bei Bedarf im laufenden IDP zu sehen, was im Detail bei der obigen "Altersberechnung" passiert, kann dies gezielt mit einem Eintrag in der Datei logging.xml aktiviert (und wieder deaktiviert) werden:

**Debug-Logging für Script-AttributeDefinition**

```
<logger name="edu.internet2.middleware.shibboleth.resolver.Script.usidiscount" level="DEBUG" />
```

Stellt man das Loglevel für dieses spezifische Modul auf DEBUG (und wartet maximal 10 Minuten, kein IDP/Tomcat neustart nötig!), schreibt der IDP ins idp-prozess.log, ob die Werte vorhanden sind oder nicht, sowie ob ein Entitlement erzeugt wurde. Nur, wenn man das Loglevel hier auf TRACE stellt, sind auch das tatsächlich errechnete Alter und die eduPersonEntitlement-Werte zu sehen. Ist man mit der IDP-Konfiguration zufrieden, kann die ganze Zeile entfernt oder das Loglevel auf INFO gestellt werden, womit weitere solche Zeilen nach wenigen Minuten ohne Neustart des IDP wieder aus dem Log verschwinden.

## attribute-filter.xml

Für die gezielte Weitergabe der/des erzeugten eduPersonEntitlements kann nun eine neue `AttributeFilterPolicy` eingefügt werden:

**Freigabe in der Attribute Filter-Konfiguration**

```
<!-- USI-Wien Student Discount -->
<afp:AttributeFilterPolicy id="USIWien">
    <afp:PolicyRequirementRule xsi:type="basic:AND">
        <basic:Rule xsi:type="basic:AttributeRequesterString" value="https://www.usi-wien.at/shibboleth" />
        <basic:Rule xsi:type="basic:AttributeValueString" attributeID="eduPersonAffiliation" value="student" />
    </afp:PolicyRequirementRule>
    <afp:AttributeRule attributeID="eduPersonEntitlement">
        <afp:PermitValueRule xsi:type="basic:AttributeValueString" value="http://usi.at/student-discount" />
    </afp:AttributeRule>
</afp:AttributeFilterPolicy>
```

Hierdurch wird für das Attribut mit id="eduPersonEntitlements" nur der Wert `http://usi.at/student-discount` weitergegeben, und zwar nur, wenn das Attribut nach der Berechnung oben einen Wert hat, wenn der Service Provider jener des USI Wien ist *und* wenn die zugreifende Person eine eduPersonAffiliation mit dem Wert `student` hat. Andere Service Provider bekommen das Attribut nicht, das USI Wien bekommt andere Entitlements nicht, und für Personen, die nicht auch Studierende sind, wird ebenfalls nichts weitergegeben.