Shibboleth Auto Login Modul für die Portalsoftware Liferay

Projekt „Nutzung von kurzlebigen Zertifikaten in Portalbasierten Grids“

– Förderkennzeichen 01IG09003 –

„Service Grids für Forschung und Entwicklung“
des Bundesministeriums für Bildung und Forschung (BMBF)

Autoren:	Stefan Pinkernell
Version:	1.0
Publikationsdatum:	28.02.2011
Kontakt:	Stefan Pinkernell
Email:	Stefan.Pinkernell@awi.de

GEFÖRDERT VOM

Bundesministerium
für Bildung
und Forschung

D-GRID

1 Einführung

Die Portalsoftware Liferay bietet schon von Haus aus eine sehr mächtige Benutzerverwaltung, bestehend aus umfangreichen Funktionen zur Verwaltung von Nutzer-Accounts, sowie einem sehr mächtigen Rollensystem zur feingranularen Konfiguration von Berechtigungen. Darüber hinaus unterstützt Liferay eine Reihe weiterer Authentifizierungsverfahren, wie z.B. LDAP und OpenId (siehe [1]). Im Rahmen des Projektes GapSLC wurde eine Evaluierung der Shibboleth-Integration in die Portalsoftware Liferay durchgeführt. Diese Evaluierung kommt zu dem Ergebnis, dass bisher noch kein geeignetes Plugin für einen Shibboleth Login für Liferay existiert [2].

Dieses Dokument beschreibt ein neu entwickeltes Plugin, das einen Shibboleth Account für die Anmeldung an einem Liferay Portal nutzt. Die als Auto Login Hook implementierte Software greift, wenn das Portal über eine definierte Adresse aufgerufen wird. Das Konzept ist dabei wie folgt: Wird das Portal über die per Shibboleth geschützte Seite aufgerufen, erkennbar an der an die Portaladresse angehängte Pfadangabe */c/portal/login*, so wird der Nutzer zunächst per Shibboleth authentifiziert. Nach erfolgreicher Authentifizierung steht in der Session dann eine für jeden Nutzer eindeutige, vom Identity Provider generierte Id zur Verfügung, die sog. Shibboleth t*argeted-Id.* Diese wird genutzt, um den Nutzer einem bereits bestehenden Liferay Account, mit dem diese Id verknüpft ist, zuzuordnen. Ist diese Zuordnung erfolgreich, so wird der Nutzer ohne weitere Interaktion mit dem zugeordneten Account am Liferay Portal angemeldet. Existiert noch kein Liferay-Account mit dieser Id, so wird automatisch ein neuer Account mit Attributen aus der Shibboleth Umgebung (*Vorname*, *Nachname* und *E-Mail Adresse*) erstellt und der Nutzer mit diesem neu erstellten Account angemeldet. Wird die Portal-Adresse ohne den Zusatz */c/portal/login* aufgerufen, ist ein Login mit den bereits bestehenden Nutzer-Accounts weiterhin möglich, was für den Zugriff mit dem Admin Account notwendig ist. Dies funktioniert natürlich nur, wenn es im Browser keine aktive Shibboleth-Session gibt.

Dieses Login Verfahren vermeidet eine doppelte Authentifizierung des Nutzers, wenn vom Portal aus per Shibboleth geschützte Services genutzt werden sollen, da ein auf diese Weise am Liferay Portal angemeldeter Nutzer bereits die dazu erforderliche Shibboleth Session besitzt. Verwendung findet dies z.B. in einem Szenario, bei dem kurzlebige Zertifikate per Portal Delegation vom DFN-SLCS bezogen werden. Da das Portal Delegation Verfahren eine per Shibboleth geschützte Umgebung erfordert und vom Portal aus gestartet wird, musste sich ein Nutzer bisher zunächst per Shibboleth authentifizieren und sich nach erfolgreicher Authentifizierung zusätzlich am Portal anmelden (siehe auch [3]).

2 Installation der Software

Diese Kapitel beschreibt, wie der entwickelte Shibboleth Auto Login Hook in ein bestehendes Liferay Portal integriert werden kann. Zudem werden Hinweise zur Konfiguration des Shibboleth Service und Identity Providers gegeben.

Die Software wurde in folgender Umgebung entwickelt und evaluiert:

- Portalrechner mit:
  - Ubuntu Linux Version 10.04 LTS
  - Shibboleth Service Provider Version 2.2.1
  - Apache2 Webserver Version 2.2.8
  - Liferay Portal Version 6.0.5 (im Bundle mit Apache Tomcat Version 6.0.26)
- Shibboleth Identity Provider Version 2.1.5

2.1 Konfiguration von Shibboleth

In der Shibboleth Umgebung müssen für die Nutzung des Auto Login Hooks einige Attribute konfiguriert werden. Folgende Shibboleth-Attribute werden genutzt:

- Shib-uid
- targeted-id
- Ship-eppn
- Shib-givenName
- Shib-surname
- Shib-mail

Konfiguration der Attribute am Identity- und Service Provider

Vom Shibboleth Identity Provider müssen diese Attribute an den das Portal schützenden Service Provider gereicht werden. Die Konfiguration dazu findet sich am IdP in den beiden Dateien *attribute-filter.xml* und *attribute-resolver.xml*, die beide im Unterverzeichnis *conf* der IdP-Installation liegen. Am Service Provider wird die Entgegennahme dieser Attribute und die Bereitstellung für die Anwendung in der Datei *attribute-map.xml* konfiguriert. Ein Auszug aus der Konfiguration dieser drei Dateien findet sich im Anhang.

Konfiguration des Shibboleth Schutzes am Service Provider

Der Login über diesen Hook geschieht über eine per Shibboleth geschützte Seite. Im Browser wird dazu die Adresse des Portals angegeben, gefolgt vom Pfad */c/portal/login*.

Die Adresse des Portals wird so konfiguriert, dass eine Shibboleth-Authentifizierung nur erfolgt, wenn der Pfad */c/portal/login* mit angegeben wurde. Ist der Nutzer authentifiziert besteht auch für alle anderen Seiten der Shibboleth Schutz und auch die Shibboleth Attribute sind auf diesen anderen Seiten zugänglich.

Ausschnitt aus der Konfiguration des Apache* Virtual *Host am Shibboleth SP

```
<Location / >
   AuthType shibboleth
   require shibboleth
   ShibUseHeaders On
   ShibRedirectToSSL 443
   ShibExportAssertion On
</Location>

<Location /c/portal/login>
   AuthType shibboleth
   require valid-user
   ShibUseHeaders On
   ShibRequireSession On
   ShibRedirectToSSL 443
   ShibRequireAll On
   ShibExportAssertion On
</Location>
```

2.2 Deployment der Software

Die Software wird, fertig kompiliert und als war-Datei gepackt, über die Projekt-Website[1] bereitgestellt und kann über die Web-Oberfläche des Liferay Portals deployed werden. Dazu wird ein Nutzer-Account mit Administrator-Rechten benötigt.

Für das Deployment wird über das *Control Panel* auf der Seite *Plugins Installation* die Schaltfläche *Install More Portlets* gewählt und auf der folgenden Seite auf *Upload File* geklickt. Auf dieser Seite kann über den File-Dialog das war-Archive selektiert werden. Durch einen Klick auf *Install* wird die Software deployed.

Zusätzlich muss ein Feld für die *targeted-Id* des Nutzer angelegt werden. Dazu wird, ebenfalls im *Control Panel*, die Seite *Custom Fields* aufgerufen und in der Kategorie *User* ein neues Feld vom Typ *Text* erstellt. Als Schlüssel muss der Wert *shibTargetedId* eingetragen werden. Zudem **muss das Feld *Searchable* auf *False* gesetzt werden**. Auf der Seite *Permissions* müssen die Berechtigungen für dieses Feld für die unterschiedlichen Nutzergruppen gesetzt werden. Dazu müssen die Berechtigungen *Update* und *View* für die Rolle *Owner* gesetzt werden.

Grundsätzlich sind die Plugin-Erweiterungen Hook und Portlet in Liferay „hotdeploy"-fähig. Das bedeutet, dass die neu hinzugefügten Module direkt nach dem Deployment ohne Neustart der gesamten Portalsoftware zur Verfügung stehen. Kommt es beim Deployment oder dem Redeployment von Plugins zu Fehlern, hilft oft ein Neustart des Portals bzw. des umgebenden Tomcat-Containers. Ein Neustart wird daher grundsätzlich nach der Installation empfohlen.

3 Technische Details

3.1 Erweiterungsmöglichkeiten von Liferay

Hooks

Um die Portalsoftware Liferay zu erweitern, stehen verschiedene Typen von Plugins zur Auswahl: Portlets, Hooks, Themes und Extensions. Die in diesem Dokument vorgestellte Software wurde als Hook implementiert. Ein Hook ist ein relativ neuer Plugin-Typ , der sich vor allem für kleinere Anpassungen an der Portalsoftware eignet. Im Gegensatz zu der in der Evaluierung [2] beschriebenen Software bietet die Entwicklung als Plugin den Vorteil, dass nicht der gesamte Quellcode des Portals mit übersetzt werden muss. Zudem ist das Deployment sehr viel einfacher.

Custom Fields

Liferay bietet mit den Custom Fields eine sehr einfache und elegante Möglichkeit, nutzerspezifische Attribute über das bestehende Nutzermanagement zu verwalten. Die Attribute werden in der Liferay-Datenbank abgelegt und sind über den sog. Expando-Service auch programmatisch nutzbar. Jedes Feld erhält einen eindeutiger Schlüssel, über den es angesprochen werden kann. Für Textfelder besteht zudem die Möglichkeit, einen Lucene-Index anzulegen, um eine performante Volltextsuche über den gespeicherten Text zu ermöglichen. Allerdings werden dabei durch interne Optimierungen Modifikationen an den Feldern vorgenommen, um die Performanz zu steigern. Beispiele dafür sind die Entfernung von Sonderzeichen (eine Id enthält u.a. Gleichzeichen) sowie eine durchgängige Umwandlung in

[1] http://gap-slc.awi.de

Kleinbuchstaben. Für die hier benötigte exakte Suche nach der richtigen Id dürfen solche Änderungen aber nicht vorgenommen werden, so dass die Indizierung nicht verwendet werden kann.

Das Liferay Plugins SDK

Die Software wurde mit dem Liferay-Plugins-SDK-6.0.5 erstellt. Dieses SDK bietet neben den erforderlichen Bibliotheken für die Entwicklung von Liferay Plugins auch vorkonfigurierte Ant-Scripte, um ein neues Plugin-Projekt anzulegen, zu übersetzen und zu packen. In der aktuellen Version wird außer den hier verwendeten Hooks auch die Entwicklung von Portlets, Liferay-Themes und Liferay-Extensions unterstützt.

Um die Ant-Scripts nutzen zu können, muss Apache Ant (mindestens Version 1.7.0) installiert sein. Zudem muss die Systemvariable *ANT_HOME* auf den Ordner zeigen, in dem Apache Ant installiert ist.

Im Verzeichnis des SDK existiert ein eigener Unterordner für jeden Plugin-Typ, in dem die Scripts zur Projekterstellung abgelegt sind. Um beispielsweise einen neuen Liferay Hook anzulegen, wird per Kommandozeile in das Unterverzeichnis *hooks* navigiert. Dort kann mit dem Create-Script (unter MS Windows create.bat bzw. unter Unix-System create.sh) das neue Projekt erstellt werden. Dazu sind zwei Parameter anzugeben: zuerst ein kurzer Projektname, der keine Leerzeichen enthalten darf und anschließend ein mit doppelten Anführungszeichen maskierter String, der den Anzeige-Namen des Hooks im Liferay Portal bildet. Zum Beispiel: *create.bat autoLogin „Shibboleth Auto Login Hook"*

Daraufhin wird ein Unterverzeichnis erstellt, dessen Name sich aus dem Projektnamen gefolgt vom Typ des Projekts zusammensetzt, in diesem Beispiel wäre dies *autoLogin-hook*. Dieses Verzeichnis enthält schon eine einfache Projektstruktur sowie ein Build-Script. Mit folgendem Befehl wird das Projekt übersetzt und in einer war-Datei gepackt: *ant build*

Das fertige war-Archiv findet sich dann im Unterverzeichnis *dist*, direkt im Hauptverzeichnis des SDK.

3.2 Details zur Implementierung

Die Software wurde als Auto Login Hook implementiert. Einzelne Codefragmente und die Konfiguration für den AutoLogin konnten aus der unter [2] evaluierten Software für diese Implementierung übernommen werden. In der Datei *portal.properties* wird die Einbindung des Hooks in die Authentifizierungspipeline wie im folgenden dargestellt konfiguriert.

Datei portal.properties

```
auth.pipeline=pre
auth.pipeline=post
auth.pipeline.enable.liferay.check=false
auto.login.hooks=com.liferay.portal.security.auth.ShibAutoLogin
```

Der AutoLoginHook besteht aus nur einer Klasse, die das Auto Login Interface von Liferay (*com.liferay.portal.security.auth.AutoLogin.java*) implementiert. Wichtig in diesem Zusammenhang ist die überschriebene Methode *login*, die ein String-Array mit den Login Daten des identifizierten Nutzers zurückgibt. Sobald das Array zurückgegeben wird, wird der Nutzer mit dem zugehörigen Account automatisch am Liferay-Portal angemeldet. Das Objekt vom Typ *com.liferay.portal.model.User.java* repräsentiert dabei einen User-Account in Liferay.

Zunächst werden über die Methode *fetchLoginAttributes* nur die Shibboleth Attribute, die für den Login mit einem bestehenden Account notwendig sind, geladen. Dies sind die Shibboleth Attribute *Shib-uid* und *targetedId* sowie eine *companyId*.

Anhand dieser Attribute wird versucht, über die Methode *findUser* den passenden Liferay-Account zu identifizieren. Gelingt dies, wird das String-Array erzeugt und zurückgegeben.

AutoLogin Credentials

```
User user = findUser();
String[] credentials = new String[3];
credentials[0] = String.valueOf(user.getUserId());
credentials[1] = user.getPassword();
credentials[2] = Boolean.TRUE.toString();
return credentials;
```

Die Methode *findUser* enthält in der aktuellen Version der Software derzeit noch einen Workaround zur Identifikation des Nutzers. Es existiert eine Funktion, um User-Accounts anhand eines Eintrages in einem Custom Field finden zu können. Allerdings funktionierte diese bei den Tests nicht zuverlässig, wofür vermutlich die bereits erwähnte Optimierungsfunktion von Lucene verantwortlich ist. In dieser Version wird daher zunächst eine Liste mit allen Accounts geladen. Dann wird in einer Iteration über alle Accounts geprüft, ob die jeweilige Id eingetragen ist und bei Erfolg das User-Objekt für den AutoLogin genutzt.

Schlägt der automatische Login fehl, werden zunächst über die Methode *fetchRemainingAttributes* die Attribute zum Anlegen eines neuen Liferay Accounts nachgeladen. Dabei handelt es sich um die Attribute *Shib-eppn*, *Shib-givenName*, *Shib-surname* und *Shib-mail*. Zudem werden mittels Passwortgenerator ein Zufallspasswort erzeugt und einige Initialisierungen vorgenommen.

Anschließend wird versucht, einen neuen Nutzer-Account mit Hilfe der Methode *addUser* aus der Klasse *com.liferay.portal.model.User.java* anzulegen. Ist dies erfolgreich, gibt diese Methode ein Objekt vom Typ *User* zurück, das im zuvor erwähnten Code zum automatischen Login genutzt wird.

Über die Methoden *setCustomAttr* und *getCustomAttr* können die Attribute in den Custom Fields geschrieben und gelesen werden. Intern wird dazu die Klasse *ExpandoValueLocalServiceUtil* aus der Liferay-API verwendet.

3.3 Alternative Implementierung

Zusätzlich zu der im vorigen Kapitel erläuterten Implementierung existiert noch eine ältere Version dieser Software. Sie verwendet das gleiche Feld, das eigentlich zur Speicherung der OpenId vorgesehen ist. Daher können beide Techniken (OpenId und Shibboleth-Login) gleichzeitig vom Portal genutzt werden können. Da die Authentifizierung mittels OpenID bereits fest im Funktionsumfang von Liferay enthalten ist, existiert für das OpenID-Feld bereits eine sehr performante Suche, die man auch nutzen kann, wenn das Feld für die TargetedID von Shibboleth verwendet wird. Daher ist der oben beschriebene Workaround dann notwendig, um einen Nutzer-Account anhand einer Targeted-Id zu finden. Der Quellcode dieser alternativen Version liegt den anderen Quellen bei und kann wie beschrieben übersetzt und deployed werden.

3.4 Shibboleth Single Logout

Über die hier beschriebene Software wird Single Sign On mit Shibboleth auch an einem Liferay Portal möglich. Einmal angemeldet hat ein Nutzer dann auch ohne erneute Anmeldung auf alle anderen, per Shibboleth geschützten Services Zugriff.

Problematischer ist allerdings das (Single-) Log Out: Leider wird diese Funktionalität von Shibboleth noch nicht komplett unterstützt. Bei dem hier vorgestellten Auto Login Hook wird der Nutzer z.B. nach einem Klick auf *Abmelden* zunächst wirklich vom Portal abgemeldet. Da die Shibboleth Session aber noch gültig ist, greift der Auto Login Hook wieder und der Nutzer wird automatisch sofort wieder angemeldet. Die derzeit einzige Methode, sich vom Portal abzumelden und die Shibboleth Session zu beenden, ist das Schließen ALLER(!) Instanzen des genutzten Browsers.

4 Anhang

Auszug aus Datei attribute-filter.xml (am IdP)

```
<AttributeFilterPolicy id="AWI Test Service Provider 2">
  <PolicyRequirementRule xsi:type="basic:AttributeRequesterString"
    value="https://shib2-sp.awi.de/shibboleth" />
  <AttributeRule attributeID="eduPersonPrincipalName">
      <PermitValueRule xsi:type="basic:ANY" />
  </AttributeRule>
  <AttributeRule attributeID="email">
      <PermitValueRule xsi:type="basic:ANY" />
  </AttributeRule>
  <AttributeRule attributeID="surname">
      <PermitValueRule xsi:type="basic:ANY" />
  </AttributeRule>
  <AttributeRule attributeID="givenName">
      <PermitValueRule xsi:type="basic:ANY" />
  </AttributeRule>
  <AttributeRule attributeID="uid">
      <PermitValueRule xsi:type="basic:ANY" />
  </AttributeRule>
  <AttributeRule attributeID="eduPersonTargetedID">
    <PermitValueRule xsi:type="basic:ANY" />
  </AttributeRule>
  <AttributeRule attributeID="transientId">
      <PermitValueRule xsi:type="basic:ANY" />
  </AttributeRule>
</AttributeFilterPolicy>
```

Auszug aus Datei attribute-resolver.xml (IdP)

```
<?xml version="1.0" encoding="UTF-8"?>

<AttributeResolver xmlns="urn:mace:shibboleth:2.0:resolver" xmlns:resolver="urn:mace:shibboleth:2.0:resolver"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:pc="urn:mace:shibboleth:2.0:resolver:pc"
xmlns:ad="urn:mace:shibboleth:2.0:resolver:ad" xmlns:dc="urn:mace:shibboleth:2.0:resolver:dc"
xmlns:enc="urn:mace:shibboleth:2.0:attribute:encoder" xmlns:sec="urn:mace:shibboleth:2.0:security"
xsi:schemaLocation="urn:mace:shibboleth:2.0:resolver classpath:/schema/shibboleth-2.0-attribute-resolver.xsd
urn:mace:shibboleth:2.0:resolver:pc classpath:/schema/shibboleth-2.0-attribute-resolver-pc.xsd
urn:mace:shibboleth:2.0:resolver:ad classpath:/schema/shibboleth-2.0-attribute-resolver-ad.xsd
urn:mace:shibboleth:2.0:resolver:dc classpath:/schema/shibboleth-2.0-attribute-resolver-dc.xsd
urn:mace:shibboleth:2.0:attribute:encoder classpath:/schema/shibboleth-2.0-attribute-encoder.xsd
urn:mace:shibboleth:2.0:security classpath:/schema/shibboleth-2.0-security.xsd">
```

```
   <resolver:AttributeDefinition id="eduPersonPrincipalName" xsi:type="Simple"
xmlns="urn:mace:shibboleth:2.0:resolver:ad"
      sourceAttributeID="eduPersonPrincipalName">
      <resolver:Dependency ref="myLDAP" />
      <resolver:AttributeEncoder xsi:type="SAML1String" xmlns="urn:mace:shibboleth:2.0:attribute:encoder"
         name="urn:mace:dir:attribute-def:eduPersonPrincipalName" />
      <resolver:AttributeEncoder xsi:type="SAML2String" xmlns="urn:mace:shibboleth:2.0:attribute:encoder"
         name="urn:oid:1.3.6.1.4.1.5923.1.1.1.6" friendlyName="eduPersonPrincipalName" />
   </resolver:AttributeDefinition>

   <resolver:AttributeDefinition id="email" xsi:type="Simple" xmlns="urn:mace:shibboleth:2.0:resolver:ad"
      sourceAttributeID="mail">
      <resolver:Dependency ref="myLDAP" />
      <resolver:AttributeEncoder xsi:type="SAML1String" xmlns="urn:mace:shibboleth:2.0:attribute:encoder"
         name="urn:mace:dir:attribute-def:mail" />
      <resolver:AttributeEncoder xsi:type="SAML2String" xmlns="urn:mace:shibboleth:2.0:attribute:encoder"
         name="urn:oid:0.9.2342.19200300.100.1.3" friendlyName="mail" />
   </resolver:AttributeDefinition>

   <resolver:AttributeDefinition id="surname" xsi:type="Simple" xmlns="urn:mace:shibboleth:2.0:resolver:ad"
      sourceAttributeID="sn">
      <resolver:Dependency ref="myLDAP" />
      <resolver:AttributeEncoder xsi:type="SAML1String" xmlns="urn:mace:shibboleth:2.0:attribute:encoder"
         name="urn:mace:dir:attribute-def:sn" />
      <resolver:AttributeEncoder xsi:type="SAML2String" xmlns="urn:mace:shibboleth:2.0:attribute:encoder"
         name="urn:oid:2.5.4.4" friendlyName="sn" />
   </resolver:AttributeDefinition>

   <resolver:AttributeDefinition id="commonName" xsi:type="Simple"
xmlns="urn:mace:shibboleth:2.0:resolver:ad"
      sourceAttributeID="cn">
      <resolver:Dependency ref="myLDAP" />
      <resolver:AttributeEncoder xsi:type="SAML1String" xmlns="urn:mace:shibboleth:2.0:attribute:encoder"
         name="urn:mace:dir:attribute-def:cn" />
      <resolver:AttributeEncoder xsi:type="SAML2String" xmlns="urn:mace:shibboleth:2.0:attribute:encoder"
         name="urn:oid:2.5.4.3" friendlyName="cn" />
   </resolver:AttributeDefinition>

   <resolver:AttributeDefinition id="givenName" xsi:type="Simple" xmlns="urn:mace:shibboleth:2.0:resolver:ad"
      sourceAttributeID="givenName">
      <resolver:Dependency ref="myLDAP" />
      <resolver:AttributeEncoder xsi:type="SAML1String" xmlns="urn:mace:shibboleth:2.0:attribute:encoder"
         name="urn:mace:dir:attribute-def:givenName" />
      <resolver:AttributeEncoder xsi:type="SAML2String" xmlns="urn:mace:shibboleth:2.0:attribute:encoder"
         name="urn:oid:2.5.4.42" friendlyName="givenName" />
   </resolver:AttributeDefinition>

   <resolver:AttributeDefinition id="uid" xsi:type="Simple" xmlns="urn:mace:shibboleth:2.0:resolver:ad"
      sourceAttributeID="uid">
      <resolver:Dependency ref="myLDAP" />
      <resolver:AttributeEncoder xsi:type="SAML1String" xmlns="urn:mace:shibboleth:2.0:attribute:encoder"
         name="urn:mace:dir:attribute-def:uid" />
      <resolver:AttributeEncoder xsi:type="SAML2String" xmlns="urn:mace:shibboleth:2.0:attribute:encoder"
         name="urn:oid:0.9.2342.19200300.100.1.1" friendlyName="uid" />
   </resolver:AttributeDefinition>

   <resolver:AttributeDefinition id="transientId" xsi:type="TransientId"
xmlns="urn:mace:shibboleth:2.0:resolver:ad">
      <resolver:AttributeEncoder xsi:type="SAML1StringNameIdentifier"
xmlns="urn:mace:shibboleth:2.0:attribute:encoder"
```

```
        nameFormat="urn:mace:shibboleth:1.0:nameIdentifier" />
      <resolver:AttributeEncoder xsi:type="SAML2StringNameID"
xmlns="urn:mace:shibboleth:2.0:attribute:encoder"
        nameFormat="urn:oasis:names:tc:SAML:2.0:nameid-format:transient" />
  </resolver:AttributeDefinition>

  <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>

  <resolver:DataConnector id="myLDAP"
    xsi:type="LDAPDirectory"
    xmlns="urn:mace:shibboleth:2.0:resolver:dc"
    ldapURL="ldap://xxx.xxx.de"
    baseDN="ou=People,dc=xxxxx,dc=de">
    <FilterTemplate>
      <![CDATA[
        (uid=$requestContext.principalName)
      ]]>
    </FilterTemplate>
  </resolver:DataConnector>


  <resolver:DataConnector xsi:type="ComputedId" xmlns="urn:mace:shibboleth:2.0:resolver:dc"
      id="computedID"
      generatedAttributeID="computedID"
      sourceAttributeID="uid"
      salt="your random string here">
    <resolver:Dependency ref="myLDAP" />
  </resolver:DataConnector>

  <resolver:PrincipalConnector xsi:type="Transient" xmlns="urn:mace:shibboleth:2.0:resolver:pc"
id="shibTransient"
    nameIDFormat="urn:mace:shibboleth:1.0:nameIdentifier" />

  <resolver:PrincipalConnector xsi:type="Transient" xmlns="urn:mace:shibboleth:2.0:resolver:pc"
id="saml1Unspec"
    nameIDFormat="urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified" />

  <resolver:PrincipalConnector xsi:type="Transient" xmlns="urn:mace:shibboleth:2.0:resolver:pc"
id="saml2Transient"
    nameIDFormat="urn:oasis:names:tc:SAML:2.0:nameid-format:transient" />

</AttributeResolver>
```

Auszug aus der Datei attribute-map.xml (SP)

```
<Attributes xmlns="urn:mace:shibboleth:2.0:attribute-map"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

      <Attribute name="urn:mace:dir:attribute-def:eduPersonPrincipalName" id="eppn">
        <AttributeDecoder xsi:type="ScopedAttributeDecoder"/>
      </Attribute>

      <Attribute name="urn:oid:1.3.6.1.4.1.5923.1.1.1.6" id="eppn">
        <AttributeDecoder xsi:type="ScopedAttributeDecoder"/>
```

```
        </Attribute>

        <Attribute name="urn:mace:dir:attribute-def:eduPersonTargetedID" id="targeted-id">
                <AttributeDecoder xsi:type="ScopedAttributeDecoder"/>
        </Attribute>

        <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>

        <Attribute name="urn:oasis:names:tc:SAML:2.0:nameid-format:persistent" id="persistent-id">
                <AttributeDecoder xsi:type="NameIDAttributeDecoder"
                formatter="$NameQualifier!$SPNameQualifier!$Name" defaultQualifiers="true"/>
        </Attribute>

</Attributes>
```

5 Literatur

[1] Liferay Portal; www.liferay.com

[2] M. Haase, P. Gietz; „Evaluierung der Shibboleth-Integration der Portalsoftware Liferay“; DAASI; 21.10.2010; http://gap-slc.awi.de/documents/Evaluierung_Liferay_v0.05.pdf

[3] S. Pinkernell, B.Fritzsch; „Einsatz von Portal Delegation und SAML Assertions bei der Authentifizierung“; 24.02.2011; http://gap-slc.awi.de/documents/portalDelegation-1.0.pdf