TIES V8 統一認証基盤構築マニュアル

平成 26 年 9 月 30 日

発行：NPO 法人 CCC-TIES

目次

はじめに

本マニュアルでは、TIES V8 を学術認証フェデレーション (以下状況に応じて「学認」) に対応させるための予備知識と設定方法を解説します。

本マニュアルは以降 3 つの章から成り立っています。

- 「知識編」では、学認自体の説明と、TIES V8 を学認対応とする上での技術の概略を説明します。
- 「設定編」では、TIES V8 を学認対応とするためのサーバ設定方法について述べます。
- 「補足編」では、「設定編」において起こり得る課題と対処方法の例を示します。具体的には、学認に依らず SAML 機構のみを用いて TIES V8 の SP・IdP を構築する方法と、ファイアウォールと AD への接続に関係する特殊な問題について解説します。

「設定編」「補足編」におけるサーバ設定の項目では、一般的な Linux サーバ管理についての知識を前提としています。

知識編

本マニュアルで TIES V8 と連携させる学術認証フェデレーション (学認) は、Single Sign On (SSO) 技術の一つである SAML を用いた大学機関等向けのフェデレーションです。本章ではそれらに関連する技術について概略を説明します。

Single Sign On (SSO) について

Web 上のサービスにおいて、一箇所でログインすれば他の複数箇所のサービスで、最初ログインした状態を再利用できる仕組みを Single Sign On (SSO) と呼びます。

例えば、インターネットの大手サービス事業者である Google では、メールサービスである GMail のためにログインした結果として、同社が提供するカレンダーサービスである Google Calendar 等でも、同じ認証情報を使うことができます。同社のサービス毎に異なるパスワードを準備し、異なるログイン情報を管理する必要はありません。これは SSO の一つです。

上記事例は 1 つの事業者内の複数のサービスにおけるログイン情報の共有事例ですが、技術的には、複数の異なる事業者間でも同様のサービスを行うことができます。

例えば、A 大学の学生が B 大学のオンライン講義サービスを受けられるよう、両大学が協定を結んだとします。

このとき、A 大学生が B 大学にアカウントがなくてもそのサービスを受けられるようにするには、B 大学で A 大学でのログイン情報を再利用するよう、システムを再設定すると良いでしょう。それにより、A 大学生は A 大学への 1 度のログインで、A 大学のサービスだけでなく B 大学のオンライン講義サービスも受けられるようになります。これも

また、典型的な SSO の一つです。

上記の例では A 大学が A 大学生 (ユーザ) を「**認証**」し、B 大学のオンライン講義サービスがその認証情報を利用します。ユーザに関するその他の情報 (ユーザ名、氏名、メールアドレス等) も、必要に応じて B 大学側に送られる可能性があります。これらの情報を本マニュアルでは以降「**属性情報**」と呼びます。

SSO を実現する技術にはいくつかあり、例えば OpenID や SAML があります。学術的分野に置いては SAML が選択されることが多く、学認においても SAML が利用されています。

Security Assertion Markup Language (SAML) について

SAML は SSO を実現する技術仕様の一つです。認証および属性情報を XML (eXtensible Markup Language) を用いてやりとりすることから、このような名前が付けられています。

SAML プロトコルにおいては、以下の 2 つの主体があります。

Identity Provider (IdP)	SP の求めに応じて認証を行い、認証情報と属性情報を SP に与える
Service Provider (SP)	IdP へ認証要求を送り、返ってきた認証・属性情報を元にサービスを行う

IdP と SP は通常、前もって相互に信頼した相手とだけ上記のやりとりを行います。相互に信頼し合う IdP・SP のグループを「**フェデレーション**」と呼びます。学認 (学術認証フェデレーション) はその一つです。

SAML では、IdP・SP は自ホストに関する情報が記載された「メタデータ」と呼ばれる一式の情報を公開し、信頼する相手のメタデータを自ホスト側に保存しておきます。メタデータには以下のような情報が含まれます。

- 設定ファイル等で自身を参照する際に用いる一意の識別子 entityID
- その IdP・SP を提供する組織名、管理者、連絡先
- その IdP・SP を保証し、署名等の検証でも用いられるサーバ証明書

メタデータも XML 形式で表現されます。以下は 2014 年 3 月時点での学認の運用フェデレーション (後述) が提供するメタデータの一部で、国立情報学研究所 (National Institute of Informatics) の IdP のメタデータ部分を抜粋したものです。

```
<EntitiesDescriptor xmlns="urn:oasis:names:tc:SAML:2.0:metadata" Name="GakuNin" validUntil="2013-10-08T14:00:00Z">
  …
  <EntityDescriptor ID="PI0001JP" entityID="https://idp.nii.ac.jp/idp/shibboleth">
  <IDPSSODescriptor protocolSupportEnumeration="urn:oasis:names:tc:SAML:1.1:protocol urn:mace:shibboleth:1.0
urn:oasis:names:tc:SAML:2.0:protocol">
    <Extensions>
      <shibmd:Scope xmlns:shibmd="urn:mace:shibboleth:metadata:1.0" regexp="false">nii.ac.jp</shibmd:Scope>
      <mdui:UIInfo xmlns:mdui="urn:oasis:names:tc:SAML:metadata:ui">
        <mdui:DisplayNamexml:lang="ja">国立情報学研究所</mdui:DisplayName>
        <mdui:DisplayName xml:lang="en">National Institute of Informatics</mdui:DisplayName>
        <mdui:Keywords xml:lang="en">category:location:kanto category:organizationType:researchInstitution</mdui:Keywords>
```

```
        </mdui:UIInfo>
      </Extensions>
      <KeyDescriptor>
        <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
          <ds:X509Data>
          <ds:X509Data>
            <ds:X509Certificate>MIIErTCCA5WgAwIBAgIIbJHSo5G+FP8wDQYJKoZIhvcNAQEFBQAwfTELMAkGA1UE
BhMCSlAxETAPBgNVBAcTCEFjYWRlbWUyMSowKAYDVQQKEyFOYXRpb25hbCBJbnN0
..
13+SYpAcExnLx95S50t1CxCU+QrSFcdBmgJR7e47qYGx39o3rRb/zIuKlGZ6oisC
pA==
</ds:X509Certificate>
          </ds:X509Data>
        </ds:KeyInfo>
      </KeyDescriptor>
      <NameIDFormat>urn:mace:shibboleth:1.0:nameIdentifier</NameIDFormat>
      <NameIDFormat>urn:oasis:names:tc:SAML:2.0:nameid-format:transient</NameIDFormat>
      <SingleSignOnService Binding="urn:mace:shibboleth:1.0:profiles:AuthnRequest"
Location="https://idp.nii.ac.jp/idp/profile/Shibboleth/SSO"/>
      <SingleSignOnService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
Location="https://idp.nii.ac.jp/idp/profile/SAML2/POST/SSO"/>
      <SingleSignOnService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"
Location="https://idp.nii.ac.jp/idp/profile/SAML2/Redirect/SSO"/>
    </IDPSSODescriptor>
    <AttributeAuthorityDescriptor protocolSupportEnumeration="urn:oasis:names:tc:SAML:1.1:protocol urn:oasis:names:tc:SAML:2.0:protocol">
        <Extensions>
          <shibmd:Scope xmlns:shibmd="urn:mace:shibboleth:metadata:1.0" regexp="false">nii.ac.jp</shibmd:Scope>
        </Extensions>
        <KeyDescriptor>
          <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
            <ds:X509Data>
              <ds:X509Certificate>MIIErTCCA5WgAwIBAgIIbJHSo5G+FP8wDQYJKoZIhvcNAQEFBQAwfTELMAkGA1UE
BhMCSlAxETAPBgNVBAcTCEFjYWRlbWUyMSowKAYDVQQKEyFOYXRpb25hbCBJbnN0
..
13+SYpAcExnLx95S50t1CxCU+QrSFcdBmgJR7e47qYGx39o3rRb/zIuKlGZ6oisC
pA==
</ds:X509Certificate>
            </ds:X509Data>
          </ds:KeyInfo>
      </KeyDescriptor>
      <AttributeService Binding="urn:oasis:names:tc:SAML:1.0:bindings:SOAP-binding"
Location="https://idp.nii.ac.jp:8443/idp/profile/SAML1/SOAP/AttributeQuery"/>
      <AttributeService Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
Location="https://idp.nii.ac.jp:8443/idp/profile/SAML2/SOAP/AttributeQuery"/>
      <NameIDFormat>urn:mace:shibboleth:1.0:nameIdentifier</NameIDFormat>
      <NameIDFormat>urn:oasis:names:tc:SAML:2.0:nameid-format:transient</NameIDFormat>
    </AttributeAuthorityDescriptor>
    <Organization>
      <OrganizationName xml:lang="en">National Institute of Informatics</OrganizationName>
      <OrganizationName xml:lang="ja">国立情報学研究所</OrganizationName>
      <OrganizationDisplayName xml:lang="en">National Institute of Informatics</OrganizationDisplayName>
      <OrganizationDisplayName xml:lang="ja">国立情報学研究所</OrganizationDisplayName>
      <OrganizationURL xml:lang="en">http://www.nii.ac.jp/</OrganizationURL>
    </Organization>
    <ContactPerson contactType="technical">
      <GivenName>Takumi</GivenName>
      <SurName>ISHII</SurName>
      <EmailAddress>mailto:lan-admin@nii.ac.jp</EmailAddress>
    </ContactPerson>
</EntityDescriptor>
```


上記から、例えば以下のような情報を読み取ることができます。

- NII の IdP の一意の識別子 (entityID) は https://idp.nii.ac.jp/idp/shibboleth
  - IDPSSODescriptor タグを用いていることから、このサーバが IdP だと分かる
- 管理する組織は英語では"National Institute of Informatics" 、日本語では「国立情報学研究所」
- NII の利用するサーバ証明書

- メッセージのやりとりを行う際に署名や暗号化に用いる

それぞれの IdP・SP は、自身のメタデータを公開し、信頼すべき相手のメタデータをローカルに保存しておきます。ユーザが SP のサービスを利用するために SP へアクセスすると、SP は自分が信頼している IdP の一つへ認証要求を送信します。IdP はその SP が自分の信頼する SP であることを確認した上でそのユーザを認証し、SP へ「このユーザは適切に認証された」と通知します。SP はその結果として、そのユーザに実際のサービスを行うことになります。

SAML の Identity Provider (IdP)とは

Identity Provider (IdP)は、ユーザの認証を行い、属性情報を Service Provider (SP)へ送るサーバです。

学認で IdP を運営する例を挙げて説明します。その IdP は、所属する大学に登録している学生や教員であれば、学生証番号・職員番号とそれに対応するパスワードの組を覚えているはずです。認証では、それらを要求することで、学内の正しいユーザかどうかを確認します。

正しく IdP にログイン出来たなら、その人はその大学に所属している適切な利用者です。そして、学認に大学が所属していれば、そのユーザは学認というフェデレーション全体で「適切なユーザ」として、そのログイン情報を再利用することができます。

参考: IdP の概要 https://meatwiki.nii.ac.jp/confluence/pages/viewpage.action?pageId=12158179

SAML の Service Provider　(SP) とは

IdP が認証したユーザに対して、実際の Web サービスを提供するサーバです。

例えば、TIES V8 の講義システムを学認対応にした場合には、SP に当たります。講義システムをゲスト以外のアカウントで利用する際、学認の IdP において認証を行います。ただし、IdP は他大学のものでも良いのです。そのユーザが正しい利用者かを特定するのは IdP の役割になります。

参考: SP の概要 https://meatwiki.nii.ac.jp/confluence/pages/viewpage.action?pageId=12158183

学術認証フェデレーション (学認) について

SAML 技術を用いて、日本において学術組織間でのフェデレーション構築に成功している事例の一つが「学術認証フェデレーション (学認)」です。

TIES V8 を新規に設定し、このフェデレーションに登録することで、学認に参加している他の大学からもそのサービスを利用できるようになります。

学認について、公式 Web ページの 2014 年 3 月時点での記述を抜粋します。

学術認証フェデレーションとは、学術 e-リソースを利用する大学，学術 e-リソースを提供する機関・出版社等から構成された連合体のことです。各機関はフェデレーションが定めた規程（ポリシー）を信頼しあうことで，相互に認証連携を実現することが可能となります。

認証連携を実現することができれば，学内でのシングルサインオン（一つの ID・パスワードであらゆるシステムが利用可能であること）を実現することが可能になるとともに，他大学や商用のサービスにおいても，1 つのパスワードを利用し，かつ ID・パスワードの再入力を行わずに利用できる環境を実現することができます。例えば，他大学の無線 LAN をいつも大学で使用している ID とパスワードで利用することができ，かつ自大学が契約している電子ジャーナルへシームレスにアクセスすることも可能となります。

(中略)

学認は、技術的には米国 Internet2 が開発した Shibboleth（注 1）を使用して実現しています。

Shibboleth は海外の多くの国で既に運用されていますが、日本語対応、運営組織の構築、規程の作成等、技術面・運用面で決めていくべきことが多数あります。特に、個人情報保護に関する法律は日本の法律に適合させる必要がある等、日本独自で開発すべきものが多数あります。

この実現のために学認では平成 20 年度に実証実験を行い、平成 21 年度から試行運用を実施しました。
そしてその成果をもとにして、平成 22 年度より学認の本格運用が開始されました。

（注 1）Shibboleth とは .. 米国 EDUCAUSE/Internet2 にて 2000 年に発足したプロジェクトで、SAML、eduPerson 等の標準仕様を利用した認証・認可のための標準仕様策定とオープンソース提供を行っています。
(https://www.gakunin.jp/fed)

TIES V8 を導入し、新規に学認へ参加される大学においては、技術的設定と並行して、学認の運用規定に従って申請作業を行うことになります。

学認に参加することによって、参加する大学等は以下のようなメリットを得ることができます。

- IdP として参加する大学は、他大学の SP 経由で在学の利用者に多様なサービスを提供できます。
- SP として参加する大学・企業は、単一の組織ににとどまらないユーザに対してサービスを展開することが出来るようになります。

学認においては、信頼するべき相手のグループとして 2 つ定義されており、そのため上記のメタデータも二種類あります。本当のユーザデータを扱うことなく SAML 技術の検証等を行える「テストフェデレーション」と、本番環境を提供する「運用フェデレーション」です。

学認への申請の段階に応じて 2 つの異なるフェデレーションを経由するため、登録されている IdP と SP はそれらに対して異なるメタデータを用います。それぞれのメタデータは 2014 年現在以下に掲載されています。

テストフェデレーション	https://metadata.gakunin.nii.ac.jp/gakunin-test-metadata.xml
運用フェデレーション	https://metadata.gakunin.nii.ac.jp/gakunin-metadata.xml

設定編

本章では TIES V8 を学認によった SAML 環境に対応させる際の設定方法について解説します。なお、設定を行うにあたって、以下の環境を想定しています。

- CentOS release 6.5 (Final)
- JDK 7u67
- Java Servlet Container として Tomcat 6.0.41
- SAML の IdP として Shibboleth IdP 2.4.2
- SAML の SP として、 yum パッケージとして得られる Shibboleth SP 2.5.3-1.1

IdP がバックエンドとなるデータベースから属性情報を取得する際に用いるプロトコルとして LDAP を用いることを仮定し、具体的なバックエンドとして、OpenLDAP と、Windows Server (2003, 2008 等) 等で動作する Active Directory の一般的な設定を元に解説します。

本マニュアルでは 2014 年 3 月時点の学認技術ガイドを元にし、その時点で判明している不足部分について、上記の配布元 Wiki ページやメーリングリストの情報等を追加して記述しています。
技術動向に応じて IdP と SP の設定方法が大きく変更されることがあるため、本マニュアルを参照いただくとともに、下記の技術ガイド等も適宜参照することをお勧めします。

- 学認の技術ガイド
  https://meatwiki.nii.ac.jp/confluence/pages/viewpage.action?pageId=12158127
- Shibboleth IdP/SP 配布元による IdP インストールガイド (Wiki ページ)
  https://wiki.shibboleth.net/confluence/display/SHIB2/IdPInstall
- Shibboleth IdP/SP 配布元による SP インストールガイド (Wiki ページ)
  https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPLinuxInstall

サーバ設定の前に準備しておきたいこと

あらかじめ以下の点について運営者側で確認しておくと、TIES V8 のサービス公開に至るまでの作業がスムーズになります。

学認への登録の流れに沿った構築スケジュールを組む

学認には「テストフェデレーション」「運用フェデレーション」の 2 種類のフェデレーションがあります。

TIES V8 を導入し、新規に学認へ登録する際には、まず「テストフェデレーション」への参加を申請して、その環境下で IdP と SP の動作検証を行うことになります。そしてその後、正式運用の申請を行い、申請が承認された段階で、最終的に「運用フェデレーション」において実際の運用を行うことになります。テストフェデレーションと運用フェデレーションでは、後述するように若干異なる設定を行う必要があります。

TIES V8 の運用スケジュールを組む際には、学認への申請者と実際のサーバ設定を行う作業者との間で、いつテストフェデレーションで稼動させ、いつ運用フェデレーションへ稼働させるかについての合意がある方が、作業がスムーズになります。

下記 Web ページ等から学認における登録の流れを確認することができますので、再度ご確認ください 。

学認 参加情報 https://www.gakunin.jp/join/

信頼できる CA から発行されたサーバ証明書を準備しておく

SAML では、安全な通信を行うため信頼できる証明書発行機関 (Certificate Authority, 以降必要に応じて CA) から発行されたサーバ証明書を用います。

サーバ証明書の取得には時間がかかることがあるため、学認対応を行うことが決まった段階であらかじめ取得準備をしておくことが望ましいでしょう。本マニュアルでは、適切なサーバ証明書を初めから取得していることを前提として、設定例を記述しています。

参考: 大学の場合、 UPKI イニシアチブ (https://upki-portal.nii.ac.jp/docs/odcert) からサーバ証明書を取得できる可能性があります。

IdP のバックエンドの設定を確認する

IdP を運営する際には、認証するユーザ (例えば学生や教員) の属性情報 (例えば学生証番号や氏名) を提供するバックエンドデータベース等から、IdP がそれらの情報を適切に引き出せるよう設定する必要があります。

一方、そういった情報は、セキュリティへの配慮からファイアウォールやアクセス制約で守られていることがあります。バックエンド自体の設定の違いも去ることながら、セキュリティ設定も組織毎に多様です。

特に IdP を設置する際には、前もって上記のようなデータを管理する担当者に相談し、前もって IdP で接続できるよう調整をおこなっておいてください。

本章では、比較的緩やかな制約のもとで準備された LDAP に対応したサーバ (OpenLDAP や Active Directory 等)を用いて認証し、TIES V8 で必要な属性を得る方法について解説します。

参考: 「補足編」ではより込み入った例の一つとして、NAT 経由でファイアウォール内の Active Directory に SSL 接続を用いてデータを取得する方法の一例を解説しています。

Shibboleth IdP の構築

本節では、SAML の SP である TIES V8 のオンライン講義サービスと連携する IdP を学認上に設置する例として、Shibboleth IdP をインストール・設定する方法について解説し

ます。

OS は CentOS 6.5 を仮定し、原則として同一サーバ上で Shibboleth IdP 以外の Web サービスが実行されていないものとします。SELinux が無効になっていることも確認してください。

IdP のホスト名は idp.example.com とし、entityID は便宜上 https://idp.example.com/idp/shibboleth として以下では説明しています。

JDK のインストール

インストールする先の CentOS が 32bits 環境であるか 64bits 環境であるかを確認し、適切な JDK をインストールします。

"uname -a"コマンドによって、現在の環境においてどちらをインストールするべきかを確認できます。

```
# uname -a
Linux idp.example.com 2.6.32-220.el6.x86_64 #1 SMP Tue Dec 6 19:48:22 GMT 2011 x86_64
x86_64 x86_64 GNU/Linux
```

64bits 環境である場合には、ここで x86_64 と記載されます。

Oracle の Web サイト(http://www.oracle.com/technetwork/java/javase/downloads/index.html)より、JDK7 の中で最新のバージョンを選んでインストールします。

2014 年 9 月 30 日時点では jdk-7u67-linux-x64.rpm が最新の JDK7 です。7u??の以下 2 ケタはセキュリティフィックス等のアップデート版です。

下記のようなコマンドラインを入力することで、JDK7 をインストールします。

```
# yum install jdk-7u67-linux-x64.rpm
```

参考: https://meatwiki.nii.ac.jp/confluence/pages/viewpage.action?pageId=12158257

注意: Shibboleth IdP は CentOS に標準でインストールされている GNU 版の Java Compiler・Virtual Machine では動作しません。仮に java 等のコマンドが既にインストールされているように見えても、JDK のインストールについて検討をおこなってください。

> Red Hat and CentOS ship with the GNU Java compiler and VM by default. These are not usable with Shibboleth so you must install another JVM. The Sun HotSpot VM is the most commonly used. On recent versions of these distros, you can also install a compatible version of OpenJDK 1.6 using yum. To find the appropriate package name, run yum makecache && yum search openjdk. You will also need to ensure that Tomcat is using OpenJDK rather than the GNU tools.
> (https://wiki.shibboleth.net/confluence/display/SHIB2/IdPInstall)

Tomcat 6 のインストール

Shibboleth IdP は Tomcat や Jetty といった Java Servlet Container の上で動く Java Servlet のソフトウェアです。本マニュアル、及び学認技術ガイドでは Tomcat 6 を選択することを仮定しています。

参考: Shibboleth IdP の配布元 Wiki ページでは、Tomcat 6 以外の Java Servlet Container ソフトウェア実装を利用して Shibboleth IdP をインストールする方法も解説されています (Jetty 7, JBoss Tomcat)。他の Container を利用する場合、以降の設定についても一部異なる設定を要求する箇所がありますので、ご注意ください。

注意: Servlet Container として Tomcat 7 を選ばないようにしてください。現時点では下記の通りの制限があります。

> Tomcat 7 is not yet supported unless you don't need support for SOAP traffic on the additional port, or are prepared to deploy an Apache web server as the "front-end" to the Tomcat service. This is a consequence of there not being a specialized certificate evaluation override for Tomcat that delegates evaluation of SP certificates to the IdP application. It is impractical in most deployments to rely on Tomcat to perform its standard methods of client certificate validation.
> (https://wiki.shibboleth.net/confluence/display/SHIB2/IdPApacheTomcatPrepare)

Tomcat のダウンロードサイト (http://tomcat.apache.org/download-60.cgi) の 「Binary Distributions」欄から apache-tomcat-6.?.??.tar.gz を選びます。2014 年 9 月 30 日現在での最新のバージョンは 6.0.41 です。

ダウンロード後、ダウンロード先において下記のようなコマンドを入力し、Tomcat 6 のインストールを行います。

```
# tar zxv -C /usr/java -f apache-tomcat-6.?.??.tar.gz
# ln -s /usr/java/apache-tomcat-6.?.?? /usr/java/tomcat
```

また、学認技術ガイド (https://meatwiki.nii.ac.jp/confluence/pages/viewpage.action?pageId=12158257) に記載されている「自動起動スクリプト」、及び、ホスト起動時の自動起動の設定も、必要に応じておこなってください。

Shibboleth IdP が依存する各種の環境変数を設定するため、下記を「# /etc/profile」の下（2 行目）に挿入します。

```
JAVA_HOME=/usr/java/default
MANPATH=$MANPATH:$JAVA_HOME/man
CATALINA_HOME=/usr/java/tomcat
TOMCAT_HOME=$CATALINA_HOME
PATH=$JAVA_HOME/bin:$CATALINA_HOME/bin:$PATH
export PATH JAVA_HOME CATALINA_HOME
```

追加した環境設定を、現在のログインターミナルで読み込みます。

```
# source /etc/profile
```

参考: chkconfig 等を用いて、システムの起動時に自動的に tomcat 6 を起動させることができます。

```
# chkconfig --add tomcat6
# chkconfig --level 345 tomcat6 on
```

httpd の設定

ここでは Web サーバである httpd と Tomcat を連携させる設定を行います。

注意: httpd の設定は、今回設定する Shibboleth IdP のインストール先ホストをどのように利用するかでも異なってきます。例えば、他の用途に一切転用するつもりがない場合には、80 番ポートへの http による接続を制限するといった運用も可能です。IdP のサーバ運用計画を元に変更をおこなってください。

学認技術ガイド (https://meatwiki.nii.ac.jp/confluence/pages/viewpage.action?pageId=12158257) では、ServerName の設定変更と、IdP のための ProxyPath の設定を行なっています。

```
ServerName idp.example.com:80
```

```
ServerName idp.example.com:443
ProxyPass /idp/ ajp://localhost:8009/idp/
```

参考: なお、学認 技術ガイドでは、ここで Tomcat 6 の設定ファイルの一つである /usr/java/tomcat/conf/server.xml の修正を行なっていますが、本稿では後述します。これは、IdP インストール後に、IdP への接続試験を行うためです
(https://wiki.shibboleth.net/confluence/display/SHIB2/IdPInstall の "A Quick Test" の項目参照)

参考: システム起動後に自動的に Shibboleth IdP を動作させるには、chkconfig 等で httpd もシステム起動時に立ち上がるようにします。

Shibboleth IdP のインストール

http://shibboleth.net/downloads/identity-provider/latest/ から最新の Shibboleth IdP (2014 年 9 月 30 日時点では 2.4.2) をダウンロードし、インストールします。

```
# unzip shibboleth-identityprovider-2.?.?-bin.zip
```

```
# cd shibboleth-identityprovider-2.?.?
# chmod a+x install.sh
# ./install.sh
```

Shibboleth IdP が正しくインストールされ、Tomcat6 上で起動している場合には、以下の Web ページへアクセスした際、Shibboleth IdP から"ok"という応答が得られます:

http://(IdP のホスト名):8080/idp/profile/Status

参考: 状況によっては、正しくインストールされている場合でも、Tomcat 自体の再起動に時間がかかる結果として、上記の正常な応答がなされるまで数分を要する場合があります。

この確認は、後述する httpd と tomcat の設定にて 8080 番ポートを閉じるまで、IdP の動作確認に用いることができます
(https://wiki.shibboleth.net/confluence/display/SHIB2/IdPInstall の "A Quick Test" の項目参照)

参考: https://meatwiki.nii.ac.jp/confluence/pages/viewpage.action?pageId=12158257

shibboleth-jce-1.1.0.jar (もしくは tomcat6-dta-ssl-1.0.0.jar) のインストール

shibboleth-jce-1.1.0.jar もしくは tomcat6-dta-ssl-1.0.0.jar を用いて JDK の java.security の設定を行います。本項目では 2014 年 3 月時点の学認技術ガイドに従って、前者の方法を記載します。これらの設定の違いについては本項の最後に説明します。

shibboleth-jce-1.1.0.jar を 下記 URL へアクセスすることでダウンロードします。

https://build.shibboleth.net/nexus/content/repositories/releases/edu/internet2/middleware/shibboleth-jce/1.1.0/shibboleth-jce-1.1.0.jar

参考: もし上記ページに存在しない場合、Shibboleth IdP 2.3.3 以前のソフトウェアから取得します。例えば、 http://shibboleth.net/downloads/identity-provider/2.3.3/shibboleth-identityprovider-2.3.3-bin.zip より Shibboleth IdP 2.3.3 を得、unzip コマンドで解凍の上、解凍されたファイルの中から lib/shibboleth-jce-1.1.0.jar を得ます。

取得した shibboleth-jce-1.1.0.jar を /usr/java/jdk1.7.0_??/jre/lib/ext/ にコピーし、/usr/java/jdk1.7.0_??/jre/lib/security/java.security (??は JDK6 のマイナーバージョンでインストールした JDK ごとに変化します) に下記のように "security.provider.10=edu.internet2.middleware.shibboleth.DelegateToApplicationProvider" の行を追加します。

```
#
# List of providers and their preference orders (see above):
#
security.provider.1=sun.security.provider.Sun
...
```

```
security.provider.8=org.jcp.xml.dsig.internal.dom.XMLDSigRI
security.provider.9=sun.security.smartcardio.SunPCSC
security.provider.10=edu.internet2.middleware.shibboleth.DelegateToApplicationProvider
```

なお、shibboleth-jce-1.1.0.jar の代わりに tomcat6-dta-ssl-1.0.0.jar を用いた設定を行う場合には、下記ページを参照してください。

- https://www.gakunin.jp/ml-archives/upki-fed/msg00501.html
- https://wiki.shibboleth.net/confluence/display/SHIB2/IdPApacheTomcatPrepare

2014 年 3 月現在、tomcat6-dta-ssl-1.0.0.jar を用いる方法が、Shibboleth IdP 配布元 Wiki における推奨設定になっております。本記事では学認の現時点での技術ガイドを元に設定する方法を優先している点、ご了承ください。

Tomcat 6 の設定

/usr/java/tomcat/conf/server.xml の修正

8009 番ポートに対する設定を下記の通りに変更します。

```
<Connector port="8009"
    protocol="AJP/1.3"    redirectPort="8443"
    enableLookups="false"
    tomcatAuthentication="false" address="127.0.0.1" />
```

8080 番ポートを以降利用しない場合には、該当する設定をコメントアウトします

```
<!--
    <Connector port="8080" protocol="HTTP/1.1"
                connectionTimeout="20000"
                redirectPort="8443" />
-->
```

後者を行い tomcat を再起動した時点で、前述した http://(IdP のホスト名):8080/idp/profile/Status へは接続できなくなります。

endorsed/ ディレクトリ下の jar ファイルのコピー

shibboleth-identityprovider-2.?.?-bin.zip を解凍した shibboleth-identityprovider-2.?.?/endorsed にある 5 つの jar ファイルを$CATALINA_HOME/endorsed ディレクトリ (通常 /usr/java/tomcat/endorsed) を作成して、そこへコピーします。

Shibboleth IdP 2.4.2 においては以下の 5 つのファイルです

- serializer-2.10.0.jar
- xalan-2.7.1.jar
- xercesImpl-2.10.0.jar
- xml-apis-2.10.0.jar

- xml-resolver-1.2.jar

これによって、Tomcat 6 の起動時にこれらの jar ファイルの内容も読み込まれ、Shibboleth IdP が適切に起動するようになります。

idp.war の設置

Shibboleth IdP のソフトウェア本体である idp.war を Tomcat の認識できる位置にコピーします。

/opt/shibboleth-idp/war/idp.war ファイルを、${CATALINA_HOME}/webapps (/usr/java/tomcat/endorsed) ディレクトリにコピーします。

```
# cp /opt/shibboleth-idp/war/idp.war ${CATALINA_HOME}/webapps
```

Shibboleth IdP の動作確認

httpd と Tomcat を再起動します。

学認技術ガイドの提供する tomcat 用起動・停止スクリプトをインストールしている場合には、例えば下記の通りに入力します。

```
# service tomcat6 stop
# service httpd restart
# service tomcat6 start
```

そうでない場合には、下記の通りに入力します。

```
# sh /usr/java/tomcat/bin/shutdown.sh
# service httpd restart
# sh /usr/java/tomcat/bin/startup.sh
```

上記の再起動後、以下の点を確認してください:

- Tomcat のログ /usr/java/tomcat/logs/catalina.out で Tomcat がエラーを出力していない
- Shibboleth IdP のログ /opt/shibboleth-idp/logs/idp-process.log で Shibboleth IdP がエラーを出力していない
- httpd のエラーログ /var/log/httpd/error_log で httpd がエラーを出力していない
- (8080 番ポートを開放している場合は) http://(IdP のホスト名):8080/idp/profile/Status にアクセスした際に ok と返答が来る (数分程度かかることがあります)

CCC-TIES 用 Shibboleth IdP の設定ファイルの変更

これまでの設定で Shibboleth IdP 自身は正常にインストールされました。ここからは、TIES V8 と学認に対応させるための設定を行います。

具体的には以下を行います。

- バックエンドデータベースへの接続設定。
- バックエンドデータベース上のデータを SAML プロトコル上の属性にマッピングする設定
- 上記マッピングを SP へ送るための設定
- 信頼する SP やフェデレーションに関する設定

バックエンドデータベースの準備

組織ごとにバックエンドデータベースの設定は異なるため、あらかじめ管理者に問い合わせてください。

本マニュアルでは、バックエンドとして LDAP プロトコルを用いて通信するサーバを想定します。学認技術ガイドで紹介されている OpenLDAP、または、一般的に用いられている Windows Server の Active Directory (以下必要に応じて AD) がこの分類に属します。

relying-party.xml の設定

このファイルには、現在設定している IdP が信頼するホストをどう決定し、それぞれに対してどのように振る舞うかを記述します。

信頼するホスト群を決定するために、ここでは次の 3 点を設定します。

- IdP に登録するフェデレーションのメタデータの設定
- IdP が SP に暗号化されたデータや署名を行うために用いる公開鍵・秘密鍵の設定
- 認証後の IdP の挙動を決定するための AnonymousRelyingParty, DefaultRelyingParty の設定

まず、利用するフェデレーションのメタデータを登録します。学認においては、学認の公式サーバから提供されるメタデータを利用します。このメタデータにはそのフェデレーションに参加する全てのホスト (IdP, SP) が記載されており、新規参加大学等が追加された場合にも適宜アップデートされた情報が IdP で使われます。

Shibboleth IdP 上で、提供されるメタデータを最新の状態で保ちつつ利用するために、relying-party.xml 上で FileBackedHTTPMetadataProvider を用いた設定を行います。これは、Web 上の URL を指定してメタデータを取得してローカルのファイルシステムに保存し、定期的に Web 上のメタデータを取得することでアップデートを行う MetadataProvider です。この際、オンライン上のメタデータの提供元を信用するためのサーバ証明書を取得するため、併せて TrustEngine タグの設定も行います。

テストフェデレーションと運用フェデレーションで必要なメタデータが異なることに気をつけてください。

- テストフェデレーション
  - メタデータの場所: https://metadata.gakunin.nii.ac.jp/gakunin-test-metadata.xml
  - 用いる証明書: gakunin-test-signer-2011.cer

(https://metadata.gakunin.nii.ac.jp/gakunin-test-signer-2011.cer から取得)

- 運用フェデレーション
  - メタデータの場所: https://metadata.gakunin.nii.ac.jp/gakunin-metadata.xml
  - 用いる証明書: gakunin-signer-2010.cer
    (https://metadata.gakunin.nii.ac.jp/gakunin-signer-2010.cer から取得)

併せて、この IdP が情報を送信する際に暗号化・署名につかう証明書・秘密鍵についても設定します。/path/to/idp.key には秘密鍵を、/path/to/idp.crt には証明書へのパスを設定します。

テストフェデレーションに登録する際の設定は、例えば以下のようになります
(/opt/shibboleth-idp/credentials/gakunin-test-signer-2011.cer に証明書を保存したものとします)

```
<metadata:MetadataProvider id="ShibbolethMetadata"
                                    xsi:type="metadata:ChainingMetadataProvider">
    <metadata:MetadataProvider id="URLMD"
          xsi:type="metadata:FileBackedHTTPMetadataProvider"
          metadataURL="https://metadata.gakunin.nii.ac.jp/gakunin-test-metadata.xml"
          backingFile="/opt/shibboleth-idp/metadata/gakunin-test-metadata.xml">
        <metadata:MetadataFilter xsi:type="metadata:ChainingFilter">
            <metadata:MetadataFilter xsi:type="metadata:RequiredValidUntil"
                                              maxValidityInterval="1296000" />
            <metadata:MetadataFilter xsi:type="metadata:SignatureValidation"
                                        trustEngineRef="shibboleth.MetadataTrustEngine"
                                        requireSignedMetadata="true" />
                <metadata:MetadataFilter xsi:type="metadata:EntityRoleWhiteList">
                <metadata:RetainedRole>samlmd:SPSSODescriptor</metadata:RetainedRole>
            </metadata:MetadataFilter>
        </metadata:MetadataFilter>
    </metadata:MetadataProvider>
</metadata:MetadataProvider>
<security:Credential id="IdPCredential" xsi:type="security:X509Filesystem">
    <security:PrivateKey>/path/to/idp.key</security:PrivateKey>
    <security:Certificate>/path/to/idp.crt</security:Certificate>
</security:Credential>
<security:TrustEngine id="shibboleth.MetadataTrustEngine"
                               xsi:type="security:StaticExplicitKeySignature">
    <security:Credential id="MyFederation1Credentials"
                                  xsi:type="security:X509Filesystem">
    <security:Certificate>/opt/shibboleth-idp/credentials/gakunin-test-signer-2011.cer</security:Certificate>
    </security:Credential>
</security:TrustEngine>
```

運用フェデレーションの場合、 backingFile に関する設定と Certificate の項目が変わり、以下のようになります。

```
<metadata:MetadataProvider id="ShibbolethMetadata"
                                    xsi:type="metadata:ChainingMetadataProvider">
    <metadata:MetadataProvider id="URLMD"
                                       xsi:type="metadata:FileBackedHTTPMetadataProvider"
```

```
            metadataURL="https://metadata.gakunin.nii.ac.jp/gakunin- metadata.xml"
            backingFile="/opt/shibboleth-idp/metadata/gakunin-metadata.xml">
          <metadata:MetadataFilter xsi:type="metadata:ChainingFilter">
              <metadata:MetadataFilter xsi:type="metadata:RequiredValidUntil"
                                       maxValidityInterval="1296000" />
              <metadata:MetadataFilter xsi:type="metadata:SignatureValidation"
                                       trustEngineRef="shibboleth.MetadataTrustEngine"
                                       requireSignedMetadata="true" />
                  <metadata:MetadataFilter xsi:type="metadata:EntityRoleWhiteList">
                  <metadata:RetainedRole>samlmd:SPSSODescriptor</metadata:RetainedRole>
              </metadata:MetadataFilter>
          </metadata:MetadataFilter>
      </metadata:MetadataProvider>
</metadata:MetadataProvider>
<security:Credential id="IdPCredential" xsi:type="security:X509Filesystem">
      <security:PrivateKey>/path/to/idp.key</security:PrivateKey>
      <security:Certificate>/path/to/idp.crt</security:Certificate>
</security:Credential>
<security:TrustEngine id="shibboleth.MetadataTrustEngine"
                              xsi:type="security:StaticExplicitKeySignature">
      <security:Credential id="MyFederation1Credentials"
                                 xsi:type="security:X509Filesystem">
          <security:Certificate>/opt/shibboleth-idp/credentials/gakunin-signer-
2010.cer</security:Certificate>
      </security:Credential>
</security:TrustEngine>
```

参考: もし秘密鍵にパスワードが設定されている場合にはそのパスワードを以下のように指定する必要があります。

```
<security:PrivateKey password="(password)"> ...    </security:PrivateKey>
```

参考: https://wiki.shibboleth.net/confluence/display/SHIB2/IdPCredentials

最後に、AnonymousRelyingParty と DefaultRelyingParty に対して自身の entityID を provider 部分に登録します。これは認証後、信頼する相手に対してどのように振る舞うかを決定する設定です。

Shibboleth IdP 2.4.2 に標準で用意されている relying-party.xml 内の設定を provider 部分だけ変えて設定します。

```
<rp:AnonymousRelyingParty provider="https://idp.example.com/idp/shibboleth"
defaultSigningCredentialRef="IdPCredential"/>
<rp:DefaultRelyingParty provider="https://idp.example.com/idp/shibboleth"
                                defaultSigningCredentialRef="IdPCredential">
    …
</rp:DefaultRelyingParty>
```

AnonymousRelyingParty と DefaultRelyingParty の詳細についてはこちらを参照してください: https://wiki.shibboleth.net/confluence/display/SHIB2/IdPUnderstandingRP

参考: https://meatwiki.nii.ac.jp/confluence/pages/viewpage.action?pageId=12158410

handler.xml と login.config の設定

LDAP をバックエンドとしている場合、まずログイン方法として「ユーザにユーザ名とパスワードの入力を促す」ように IdP を設定し、そのユーザ名とパスワードを LDAP データベース上で検索するよう設定します。

参考: ユーザ名とパスワードの入力を促す画面のカスタマイズも行えます。この設定の詳細については以下を参照してください:
https://wiki.shibboleth.net/confluence/display/SHIB2/IdPAuthUserPass

Shibboleth IdP 2.4.2 に標準で付属している handler.xml 内で

- RemoteUser を用いた LoginHandler をコメントアウト (無効化)
- UsernamePassword を用いた LoginHandler をコメントイン (有効化)

```
<!--
<ph:LoginHandler xsi:type="ph:RemoteUser">
<ph:AuthenticationMethod>urn:oasis:names:tc:SAML:2.0:ac:classes:unspecified</ph:AuthenticationMethod>
</ph:LoginHandler>
-->
..
<ph:LoginHandler xsi:type="UsernamePassword"
                        jaasConfigurationLocation="file:///opt/shibboleth-idp/conf/login.config">
<ph:AuthenticationMethod>urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport</ph:Authenti
cationMethod>
</ph:LoginHandler>
```

参考: https://meatwiki.nii.ac.jp/confluence/pages/viewpage.action?pageId=12158412

login.config 内で LDAP の検索に関する設定をします。**この設定はバックエンドデータベースの状況によって大きく異なります**。ここでは 3 つの例を挙げますので、参考にしてください。

例 1: ローカルに LDAP サーバを設置してあるケース

- 同じホストに OpenLDAP のような LDAP サーバが設置してある
- 検索時に LDAP サーバ自体へのログインは必要ない
- SSL は用いない
- uid をログインする際のユーザ名とする
- LDAP 内での検索の支点になる base dn が o=test_o,dc=ac,c=JP である
- LDAP 木構造上で、 base dn の子供の木も再帰的に検索する

この場合、login.config の設定は以下のようになります。

```
   edu.vt.middleware.ldap.jaas.LdapLoginModule required
       ldapUrl="ldap://localhost"
       baseDn="o=test_o,dc=ac,c=JP"
       ssl="false"
       userFilter="uid={0}"
       subtreeSearch="true";
```

例 2: 外部から LDAP サーバが提供されているケース

- ldap.example.org に OpenLDAP のような LDAP サーバが設置してある
- 検索時にログインが必要
  - ユーザ名: uid=user,ou=People,dc=example,dc=org
  - パスワード: password
- baseDn が ou=People,dc=example,dc=org

この場合、login.config の設定は以下のようになります。

```
edu.vt.middleware.ldap.jaas.LdapLoginModule required
    ldapUrl="ldap://ldap.example.org"
    bindDn="uid=user,ou=People,dc=example,dc=org"
    bindCredential="password"
    baseDn="ou=People,dc=example,dc=org"
    ssl="false"
    userFilter="uid={0}"
    subtreeSearch="true";
```

例 3: Active Directory へ SSL アクセスするケース

- ldap.example.com に ldaps (SSL 接続された ldap) で接続する
- 特権ユーザとして admin@tune.local, パスワード admin でログインして検索を行う
- sAMAccountName をログインする際のユーザ名とする
- base dn は dc=tune,dc=local

といった場合には、例えば下記のようになります。

```
edu.vt.middleware.ldap.jaas.LdapLoginModule required
    ldapUrl="ldaps://ldap.example.com"
    bindDn="admin@tune.local"
    bindCredential="admin"
    ssl="true"
    baseDn="dc=tune,dc=local"
    userFilter="sAMAccountName={0}"
    subtreeSearch="true";
```

参考: 前もって、Shibboleth IdP に依存しない LDAP 接続ツールを用いて、サーバが 期待通りに動作していることを確認しておくことをお勧めします。例えば、Linux 上では ldapsearch を用いることが考えられます。CentOS 6.5 では下記のような形でインストールできます:

```
# yum install openldap-clients
```

利用例の一つを示します:

```
# ldapsearch -x -H ldaps://ldap.example.com -D 'admin@tune.local' -b dc=tune,dc=local
sAMAccountName=test -w admin
```

attribute-resolver.xml の設定

attribute-resolver.xml を編集することで、IdP 内での属性値の取得方法を指定します。主に、データの取得元 (例: LDAP や AD)と取得データの加工方法 (例: 受け取ったデータそのまま、末尾に文字列を追加して送信、等) を指定します。

attribute-resolver.xml で利用できる機能は多彩であるため、本項では CCC-TIES の設定に利用する項目についてのみ紹介します。

参考: 学認で利用できる属性値とそれに対する attribute-resolver.xml の設定方法については、以下の Web サイトを参照してください:
https://meatwiki.nii.ac.jp/confluence/pages/viewpage.action?pageId=12158166

CCC-TIES (www.tieskun.net) と相互接続を行う際には、以下の属性を SP へ公開することが必須です:

- eduPersonPrincipalName (一意の ID として用います)

以下は、IdP が公開した場合には、 CCC-TIES のサービス上で適切に用いられます。

- mail (各利用者の連絡先メールアドレスとして用います)
- unscoped-affliation
  - faculty, staff の場合には educator となり、コース作成権限がつきます
  - student, member, 空の場合には learner となり、履修のみ行えます
- sn (姓)
- givenName (名)

例えば、それぞれの属性について、LDAP から以下のように情報が得られるとします

- LDAP 上の mail, sn, givenName, eduPersonAffiliation を IdP 上の同一の属性名にマッピングする
- LDAP 上で uid として得られる値に、この IdP (idp.example.com) の scope である idp.example.com を追加したものを、IdP 上の属性名 eduPersonPrincipalName にマッピングする
  - つまり、uid として "example" が LDAP から得られた場合に、eduPersonPrincipalName は "example@idp.example.com" としたい

Data Connector

まず LDAP から情報を取得するための Data Connector を attribute-resolver.xml 内に定義します。下記の例では ldap.example.com と言う LDAP サーバから得られる情報を myLDAP という id を持つ Data Connector に割り当てます。

```
    <resolver:DataConnector id="myLDAP" xsi:type="dc:LDAPDirectory"
        ldapURL="ldap://ldap.example.com"
        baseDN="ou=TiesTest,o=test_o,dc=ac,c=JP"
```

```
        principal="uid=test001,o=test_o,dc=ac,c=JP"
        principalCredential="test001">
        <dc:FilterTemplate>
            <![CDATA[
                (uid=$requestContext.principalName)
            ]]>
        </dc:FilterTemplate>
    </resolver:DataConnector>
```

Data Connector 設定後、attribute-resolver.xml でその情報を Attribute Definition を設定することで IdP の属性値に変換します。本マニュアルでは 3 種類の Attribute Definition を利用する設定を紹介します。

参考: Shibboleth IdP の提供する他の種類の Attribute Definition についてはこちらを参照してください: https://wiki.shibboleth.net/confluence/display/SHIB2/IdPAddAttribute

Simple Attribute Definition

LDAP 上の値をそのまま IdP 上の属性値に割り当てる場合には、Simple Attribute Definition と呼ばれる属性定義を用います。Simple Attribute Definition は取得した値を加工せずにそのまま使用する、Attribute Definition の一種です

例えば givenName という属性値を定義するために、LDAP の同名の属性を利用する場合には以下のように設定します。

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

この例の場合、myLDAP という前述した Data Connector から得られる givenName (sourceAttributeID の側) を Shibboleth IdP の givenName (id の側)と結びつけ、AttributeEncoder タグを用いて SAML の仕様に沿った属性名へマッピングしています。ここでは AttributeEncoder の詳細は省略します。attribute-resolver.xml 内には学認において利用できる全属性値についての AttributeEncoder 設定例があるため、各属性については適宜コメントアウト・コピーするといった形で設定すれば十分です。

注意: 以降の例を含め、Shibboleth の設定を理解するためには、XML の仕様、特に XML における「名前空間」の概念についても事前に理解しておく必要があります。上記の例の場合、resolver:AttributeDefinition という文字列は「"resolver"という事前に指定された名前空間内の AttributeDefinition タグ」を示しており、"resolver"という名前空間につい

ての定義が、この設定の外部で事前に終了していることを暗黙に仮定しています。後述する attribute-resolver.xml では名前空間に関する定義も含めて完全な XML を掲載していますので、そちらも併せて確認してください。

Simple Attribute Definition のより詳しい説明は以下を参照してください。
https://wiki.shibboleth.net/confluence/display/SHIB2/ResolverSimpleAttributeDefinition

Scoped Attribute Definition

LDAP 上の文字列の後に、メタデータで指定した scope を@付きで結合したいケースがあります。例えば、scope が"idp.example.com"であったとき、LDAP から得られる uid に "@idp.example.com" を結合した "(uid)@idp.example.com" を eduPersonPrincipalName としたい場合があります。その場合は、 Simple Attribute Definition ではなく、Scoped Attribute Definition を用います。Scoped Attribute Definition は、受け取った情報に scope を結合するための Attribute Definition です。

以下に eduPersonPrincipalName における例を示します。

```
<resolver:AttributeDefinition xsi:type="ad:Scoped"
                              id="eduPersonPrincipalName"
                              scope="idp.example.com"
                              sourceAttributeID="uid">
    <resolver:Dependency ref="myLDAP" />
    <resolver:AttributeEncoder xsi:type="enc:SAML1ScopedString"
                               name="urn:mace:dir:attribute-def:eduPersonPrincipalName" />
    <resolver:AttributeEncoder xsi:type="enc:SAML2ScopedString"
                               name="urn:oid:1.3.6.1.4.1.5923.1.1.1.6"
                               friendlyName="eduPersonPrincipalName" />
</resolver:AttributeDefinition>
```

この設定を行った場合、myLDAP から得られた uid を scope で指定された "idp.example.com" と @ を介して結合しています。

注意: Scoped Attribute Definition を用いて設定する際、IdP 自身のメタデータに記載されている scope と、attribute-resolver.xml に記載される scope が一致するようにしてください。一致していない場合、意図せず属性値が SP 側に届かない可能性があります。参考まで、「知識編」で掲載した NII のメタデータの一部に記載されている scope の例を再掲載します。自信の IdP を設定するにあたっても、同様に IdP のメタデータにおいて scope を設定しているはずです。

```
<EntityDescriptor ID="PI0001JP" entityID="https://idp.nii.ac.jp/idp/shibboleth">
  <IDPSSODescriptor protocolSupportEnumeration="urn:oasis:names:tc:SAML:1.1:protocol
urn:mace:shibboleth:1.0
urn:oasis:names:tc:SAML:2.0:protocol">
    <Extensions>
      <shibmd:Scope xmlns:shibmd="urn:mace:shibboleth:metadata:1.0"
regexp="false">nii.ac.jp</shibmd:Scope>
      <mdui:UIInfo xmlns:mdui="urn:oasis:names:tc:SAML:metadata:ui">
        <mdui:DisplayNamexml:lang="ja">国立情報学研究所</mdui:DisplayName>
```

```
            <mdui:DisplayName xml:lang="en">National Institute of Informatics</mdui:DisplayName>
            <mdui:Keywords xml:lang="en">category:location:kanto</mdui:Keywords>
          </mdui:UIInfo>
        </Extensions>
..
```

Scoped Attribute Definition のより詳しい説明は以下を参照してください。
https://wiki.shibboleth.net/confluence/display/SHIB2/ResolverScopedAttributeDefinition

Script Attribute Definition

「Data Connector から受け取った値の末尾に@と特定の文字列を付加する」という設定例において、@以降に結合する文字列が scope でないケースでは Scoped Attribute Definition は使えません。その場合には、より柔軟性のある Script Attribute Definition を用いることができます。

例えば mail を設定する際、@ 以降が(scope と異なり)"example.org" であった場合の例として、Script Attribute Definition を用いた例を下記に示します。

```
    <resolver:AttributeDefinition xsi:type="ad:Script" id="mail" sourceAttributeID="uid">
        <resolver:Dependency ref="myLDAP" />
        <resolver:AttributeEncoder xsi:type="enc:SAML1String"
                                   name="urn:mace:dir:attribute-def:mail" />
        <resolver:AttributeEncoder xsi:type="enc:SAML2String"
                                   name="urn:oid:0.9.2342.19200300.100.1.3"
                                   friendlyName="mail" />
        <ad:Script><![CDATA[
importPackage(Packages.edu.internet2.middleware.shibboleth.common.attribute.provider);
          mail = new BasicAttribute("mail");
          if (typeof(uid) !== 'undefined') {
            mail.getValues().add(uid.getValues().iterator().next() + '@example.org');
          }
        ]]></ad:Script>
    </resolver:AttributeDefinition>
```

Script Attribute Definition は JavaScript ベースの設定を行えるため、強力である一方やや複雑です。Shibboleth IdP 配布元のマニュアル等を参照しながら設定することをお勧めします。https://wiki.shibboleth.net/confluence/display/SHIB2/ResolverScriptAttributeDefinition

下記の attribute-resolver.xml 全文の例においては Script Attribute Definition は利用していません。

attribute-resolver.xml 設定例

attribute-resolver.xml 全体としては、例えば以下のようになります。

```
<?xml version="1.0" encoding="UTF-8"?>
<resolver:AttributeResolver 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 xsi:type="ad:Simple"
                                  id="mail"
                                  sourceAttributeID="mail">
        <resolver:Dependency ref="myLDAP" />
        <resolver:AttributeEncoder xsi:type="enc:SAML1String"
                                   name="urn:mace:dir:attribute-def:mail" />
        <resolver:AttributeEncoder xsi:type="enc:SAML2String"
                                   name="urn:oid:0.9.2342.19200300.100.1.3"
                                   friendlyName="mail" />
    </resolver:AttributeDefinition>
    <resolver:AttributeDefinition xsi:type="ad:Simple"
                                  id="givenName"
                                  sourceAttributeID="givenName">
        <resolver:Dependency ref="myLDAP" />
        <resolver:AttributeEncoder xsi:type="enc:SAML1String"
                                   name="urn:mace:dir:attribute-def:givenName" />
        <resolver:AttributeEncoder xsi:type="enc:SAML2String"
                                   name="urn:oid:2.5.4.42"
                                   friendlyName="givenName" />
    </resolver:AttributeDefinition>
    <resolver:AttributeDefinition xsi:type="ad:Simple"
                                  id="sn"
                                  sourceAttributeID="sn">
        <resolver:Dependency ref="myLDAP" />
        <resolver:AttributeEncoder xsi:type="enc:SAML1String"
                                   name="urn:mace:dir:attribute-def:sn" />
        <resolver:AttributeEncoder xsi:type="enc:SAML2String"
                                   name="urn:oid:2.5.4.4" friendlyName="sn" />
    </resolver:AttributeDefinition>

    <resolver:AttributeDefinition xsi:type="ad:Simple"
                                  id="eduPersonAffiliation"
                                  sourceAttributeID="eduPersonAffiliation">
        <resolver:Dependency ref="myLDAP" />
        <resolver:AttributeEncoder xsi:type="enc:SAML1String"
                                  name="urn:mace:dir:attribute-def:eduPersonAffiliation" />
        <resolver:AttributeEncoder xsi:type="enc:SAML2String"
                                   name="urn:oid:1.3.6.1.4.1.5923.1.1.1.1"
                                   friendlyName="eduPersonAffiliation" />
    </resolver:AttributeDefinition>
```

```
    <resolver:AttributeDefinition xsi:type="ad:Scoped"
                                          id="eduPersonPrincipalName"
                                          scope="idp.example.com"
                                          sourceAttributeID="uid">
        <resolver:Dependency ref="myLDAP" />
        <resolver:AttributeEncoder xsi:type="enc:SAML1ScopedString"
                                       name="urn:mace:dir:attribute-def:eduPersonPrincipalName" />
        <resolver:AttributeEncoder xsi:type="enc:SAML2ScopedString"
                                          name="urn:oid:1.3.6.1.4.1.5923.1.1.1.6"
                                          friendlyName="eduPersonPrincipalName" />
    </resolver:AttributeDefinition>

    <resolver:DataConnector id="myLDAP" xsi:type="dc:LDAPDirectory"
        ldapURL="ldap://ldap.example.com"
        baseDN="ou=TiesTest,o=test_o,dc=ac,c=JP"
        principal="uid=test001,o=test_o,dc=ac,c=JP"
        principalCredential="test001">
        <dc:FilterTemplate>
            <![CDATA[
                (uid=$requestContext.principalName)
            ]]>
        </dc:FilterTemplate>
    </resolver:DataConnector>
</resolver:AttributeResolver>
```

- LDAP バックエンドから属性情報を得るため、"myLDAP" という DataConnector を定義して、ldap.examplec.com 向けの設定をしています。attribute-resolver.xml の標準設定に即して、ファイル末尾に記載してあります。
- AttributeDefinition によってこの IdP で用いる属性を一つずつ設定します。IdP 内の本ファイルや attribute-filter.xml で用いる一意の識別子を"id"で設定します。上記の例では mail, sn, givenName, eduPersonAffiliation, eduPersonPrincipalName を指定してあります。
  - 各 AttributeDefinition の属性値を取るデータ元を指定するため、Dependency として myLDAP を指定しています。
  - sourceAttributeID を指定することで、myLDAP という DataConnector から得られた LDAP 内のデータのうち sourceAttributeID で指定した名前を持つ情報をこの属性にマッピングすることを要求します。
  - mail, sn, givenName, eduPersonAffiliation では LDAP から直接値が取れると仮定し、Simple Attribute Definition を利用しています。
  - eduPersonPrincipalName では "@idp.example.com" という接尾辞を myLDAP から取得した uid に追加するため、SimpleAttributeDefintion ではなく ScopedAttributeDefinition を用いています。
- AttributeEncoder によって、SAML の古いバージョンである Shib 1.3 と、現在のバージョンである SAML 2.0 において、IdP と SP が通信する際の属性名を指定します。
  - 学認では利用できる属性とその OID 等は定まっており、自由に変更することは出来ません

参考

- 属性リスト:
  https://meatwiki.nii.ac.jp/confluence/pages/viewpage.action?pageId=12158166
- AttributeNaming: https://wiki.shibboleth.net/confluence/display/SHIB2/AttributeNaming

attribute-filter.xml の設定

Shibboleth IdP では対向する SP に応じて公開する属性値を変更することが出来ます。Shibboleth IdP 2.4.2 の標準設定では、属性値はほぼ公開しない設定になっているため、attribute-resolver.xml に追加してこちらの設定も行う必要があります。この設定変更のため、attribute-filter.xml を編集します。

例えば、www.tieskun.net (entityID: https://www.tieskun.net/shibboleth-sp) に対して属性値を公開する場合には、以下のようにします。

```
<?xml version="1.0" encoding="UTF-8"?>
<afp:AttributeFilterPolicyGroup id="ShibbolethFilterPolicy"
        xmlns:afp="urn:mace:shibboleth:2.0:afp"
        xmlns:basic="urn:mace:shibboleth:2.0:afp:mf:basic"
        xmlns:saml="urn:mace:shibboleth:2.0:afp:mf:saml"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="urn:mace:shibboleth:2.0:afp classpath:/schema/shibboleth-2.0-afp.xsd
                            urn:mace:shibboleth:2.0:afp:mf:basic classpath:/schema/shibboleth-2.0-afp-mf-
basic.xsd
                            urn:mace:shibboleth:2.0:afp:mf:saml classpath:/schema/shibboleth-2.0-afp-mf-
saml.xsd">
    <afp:AttributeFilterPolicy id="AttributeForTIES">
        <afp:PolicyRequirementRule xsi:type="basic:AttributeRequesterString"
                                   value="https://www.tieskun.net/shibboleth-sp" />

        <afp:AttributeRule attributeID="mail">
            <afp:PermitValueRule xsi:type="basic:ANY" />
        </afp:AttributeRule>
        <afp:AttributeRule attributeID="sn">
            <afp:PermitValueRule xsi:type="basic:ANY" />
        </afp:AttributeRule>
        <afp:AttributeRule attributeID="givenName">
            <afp:PermitValueRule xsi:type="basic:ANY" />
        </afp:AttributeRule>
        <afp:AttributeRule attributeID="eduPersonAffiliation">
            <afp:PermitValueRule xsi:type="basic:ANY" />
        </afp:AttributeRule>
        <afp:AttributeRule attributeID="eduPersonPrincipalName">
            <afp:PermitValueRule xsi:type="basic:ANY" />
        </afp:AttributeRule>
    </afp:AttributeFilterPolicy>
</afp:AttributeFilterPolicyGroup>
```

参考: https://meatwiki.nii.ac.jp/confluence/pages/viewpage.action?pageId=12158418

Shibboleth IdP の動作確認

この IdP と対話する SP が既に準備できている場合には、以上の設定で属性値を SP から参照することが出来ます。その際には、その SP に関するメタデータや entityID を適切に IdP に指定してあることを再度確認してください。

学認テストフェデレーションにおいては、IdP の動作検証のために test-sp1, test-sp2, test-sp3 と言う 3 種類の検証用 SP が存在します。テストフェデレーションに登録された段階でこれらを用いて IdP の属性検証を行うことが可能です。

注意: **これらの SP と IdP との間でやりとりされる属性情報（氏名，メールアドレス等個人情報を含む）は，アクセスログに記録され，デバッグ用として Web 上で公開されます。このため，必ずダミーのデータを用いてテストしてください。**

参考: https://www.gakunin.jp/join/test/

Shibboleth SP の構築

TIES V8 においては、 SAML の SP として Shibboleth SP を用います。ここでは、学認を経由するなどして IdP への認証要求を行い、認証の後に IdP から送られてくる属性情報を、Moodle 上の Shibboleth-TIES へ渡すための設定を行います。

本章は、Shibboleth IdP と同様新規にインストールされた CentOS 6.5 を元に設定を行い、同一サーバ上で TIES V8 以外の Web サービスを行なっていないものとして書かれています。SELinux が無効になっていることも確認してください。

Moodle は /var/www/html/ 配下に直接インストールされており、結果としてホスト名が sp.example.com であったとき、TIES V8 のトップページへは https://sp.example.com/ によって アクセスできるものとします。entityID は https://sp.example.com/shibboleth-sp とします。

Shibboleth SP のインストール

Shibboleth の配布元が提供している yum レポジトリから Shibboleth SP のパッケージをダウンロード・インストールします。この際、下記の操作を行います。

- 新たに登録されるレポジトリで使われる PGP 鍵をシステムにインポートする
- レポジトリを登録する
- Shibboleth SP パッケージをレポジトリからインストールする

まず PGP 鍵のインポートを行います。 https://www.shibboleth.net/downloads/PGP_KEYS において、

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
..
-----END PGP PUBLIC KEY BLOCK-----
```

となっている部分 (PGP 鍵) のうち、"security:shibboleth OBS Project <security:shibboleth@build.opensuse.org>" の下にあるものをコピーし、ローカルのファイルに保存します (これを KEY) とします。

参考: 2014 年 9 月時点での該当部分は下記のようになっています:

```
pub     1024D/7D0A1B3D 2008-06-30 [expires: 2014-01-23]
uid                    security:shibboleth OBS Project <security:shibboleth@build.opensuse.org>

-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1.4.5 (GNU/Linux)

mQGiBEholYgRBACW/kTYZi5mAEMP8j1qui2bRnWmYblFbiZvb5JMJYUWL/jyMjHj
LLJvVrnOre/AxexH9KmaJwBuNoa/X9l/tQGIb49QRhR346QQUbQcwlYpckg5ccqN
qlAURHEdjCxRxMhzPs+C/F6Nqa/fHpAectW0JNRqAVVd9CWjCG3l6I2CywCgk9UI
SUAvaB5bVMxEVrFAKrVh4MUEAISeUwOaTIZftIamjo0VrnYemHS4SmGqMALEtHeG
..
cQCbBXZqIBQB4VexmP1wO8cZy5RoYJ+IRgQTEQIABgUCSGiViAAKCRA7MBG3a51l
Iy3vAKCUPurlZup+vzQtpij3FMo0JAVW9ACgjdkt/hWt0WsjfHb+/cPwXtgx9X8=
```

```
=++HN
-----END PGP PUBLIC KEY BLOCK-----
```

次に、新しく準備された KEY ファイルに以下のコマンドを入力し、

```
# gpg --quiet --import KEYS ; gpg --fingerprint 0x7D0A1B3D
```

コマンドライン入力の結果が以下のフィンガープリント(指紋)と一致することを確認してください。

```
pub     1024D/7D0A1B3D 2008-06-30 [満了: 2014-01-23]
                      指紋  = 6519 B5DB 7C1C 8340 A954    ED00 73C9 3745 7D0A 1B3D
uid                           security:shibboleth OBS Project
<security:shibboleth@build.opensuse.org>
```

参考: システムロケールが en_US 等である場合は以下のように英語で表記されることがあります。

```
pub     1024D/7D0A1B3D 2008-06-30 [expires: 2014-01-23]
        Key fingerprint = 6519 B5DB 7C1C 8340 A954    ED00 73C9 3745 7D0A 1B3D
uid                           security:shibboleth OBS Project
<security:shibboleth@build.opensuse.org>
```

フィンガープリントが正しいことを確認の上、最後に下記のコマンドを入力することで PGP 鍵をインポートします

```
# rpm --import KEYS
```

注意: 上記フィンガープリントが示す日付の通り、本稿執筆時点(2014 年 9 月)で使われている PGP 鍵は 2014 年 4 月に既に期限切れとなっています。今後、上記と異なる証明書とフィンガープリントに更新される可能性があります。学認の最新の技術ガイド等を適宜参照し、適切な PGP 鍵をインポートしてください。上記の証明書の処理は、Shibboleth SP インストール時にパッケージの正当性を確認するものですので、証明書の期限切れによって、Shibboleth SP 自体の動作に直ちに深刻な影響が発生するものではありません。

参考: https://meatwiki.nii.ac.jp/confluence/pages/viewpage.action?pageId=12158264

次に Shibboleth SP を CentOS の yum パッケージ管理システムでインストールできるよう、repository を追加します。

```
# wget http://download.opensuse.org/repositories/security://shibboleth/CentOS_CentOS-
6/security:shibboleth.repo
```

```
# cp security¥:shibboleth.repo /etc/yum.repos.d/shibboleth.repo
```

最後に Shibboleth SP をインストールします。

```
# yum install shibboleth
```

このとき、システムが x86_64 環境である場合には代わりに以下のように入力します。

```
# yum install shibboleth.x86_64
```

x86_64 環境であるかそうでないかは、"uname -a"コマンドによって確認できます。

```
# uname -a
Linux sp.example.com 2.6.32-220.el6.x86_64 #1 SMP Tue Dec 6 19:48:22 GMT 2011 x86_64
x86_64 x86_64 GNU/Linux
```

64bits 環境である場合には、ここで x86_64 と記載されます。

httpd の設定

前もって httpd と mod_ssl をインストールしておきます。

```
# yum install httpd mod_ssl
```

SSL 接続を行うための証明書、秘密鍵等を /etc/httpd/conf.d/ssl.conf にて設定します。証明書を発行した機関によっては中間 CA 証明書と呼ばれる証明書を含めて設定する必要があります。中間 CA 証明書は、証明書を発行した CA によって Web 上で公開されているのが一般的です。

- サーバ証明書 /etc/pki/tls/certs/server.crt
- サーバ秘密鍵 /etc/pki/tls/private/server.key
- 中間 CA 証明書 /etc/pki/tls/certs/server-chain.crt

上記のように証明書が保存されていた場合、ssl.conf では以下のように設定します (ここでは同時にホスト名を指定する ServerName 設定も掲載しています)

```
ServerName sp.example.com:443

SSLCertificateFile /etc/pki/tls/certs/server.crt
SSLCertificateKeyFile /etc/pki/tls/private/server.key
SSLCertificateChainFile /etc/pki/tls/certs/server-chain.crt
```

/etc/httpd/conf.d/shib.conf にて、Moodle の Shibboleth-TIES モジュールにアクセス制限を行い、モジュールへアクセスする際に IdP へ認証要求を送るように httpd の設定を変更します。例えば以下のような設定項目を追加します。

```
<Directory /var/www/html/auth/shibboleth_ties/index.php>
```

```
    AuthType shibboleth
    ShibRequestSetting requireSession 1
    require valid-user
</Directory>
```

本ホストで実行されるサービスが TIES V8 のみの場合、http 接続に対して https 接続を行うようリダイレクトを行うことも可能です。/etc/httpd/conf/httpd.conf 末尾に下記のように記述すると、全ての接続要求が https 接続にリダイレクトされます。

```
<VirtualHost *:80>
        Redirect / https://sp.example.com/
</VirtualHost>
```

ここまで設定した上で、仮に httpd をまだ起動していない場合には以下を入力します

```
# service httpd start
```

起動済の場合には以下を入力します。

```
# service httpd restart
```

参考: https://meatwiki.nii.ac.jp/confluence/pages/viewpage.action?pageId=12158268

shibboleth2.xml の設定

ここでは以下を設定します。

- SP 自身の entityID を指定します
- SP が IdP を探す際に学認の Discovery Service を用いて学認の IdP を選択する画面を有効にします
- SP が信頼するメタデータの設定を行います
- SP の用いる証明書と秘密鍵を指定します
- IdP から送られてきた属性情報を SP 内部の名前へマッピングします

属性値に関する設定以外は全て、/etc/shibboleth/shibboleth2.xml を変更することで Shibboleth SP に反映します。

まず、ApplicationDefaults の entityID を設定します。

```
<ApplicationDefaults entityID="https://sp.example.com/shibboleth-sp"
                     REMOTE_USER="eppn persistent-id targeted-id">
```

次に、学認の提供する Discovery Service (DS) の設定を行うため、SessionInitiator の設定を追加します。テストフェデレーションと本番フェデレーションで用いる DS は異なります。

- テストフェデレーション: https://test-ds.gakunin.nii.ac.jp/WAYF

- 運用フェデレーション: https://ds.gakunin.nii.ac.jp/WAYF

指定する SAMLDS 行も異なります。以下はテストフェデレーションにおける設定例です。

```
<Sessions lifetime="28800" timeout="3600" relayState="ss:mem"
            checkAddress="false" handlerSSL="false" cookieProps="http">
    <SSO discoveryProtocol="SAMLDS"
          discoveryURL="https://test-ds.gakunin.nii.ac.jp/WAYF">
        SAML2 SAML1
    </SSO>

    <Logout>SAML2 Local</Logout>

    <Handler type="MetadataGenerator" Location="/Metadata" signing="false"/>
    <Handler type="Status" Location="/Status" acl="127.0.0.1 ::1"/>
    <Handler type="Session" Location="/Session" showAttributeValues="false"/>
    <Handler type="DiscoveryFeed" Location="/DiscoFeed"/>

    <SessionInitiator type="Chaining" Location="/DS" isDefault="true" id="DS">
        <SessionInitiator type="SAML2" template="bindingTemplate.html"/>
        <SessionInitiator type="Shib1"/>
        <SessionInitiator type="SAMLDS"
                          URL="https://test-ds.gakunin.nii.ac.jp/WAYF"/>
    </SessionInitiator>
</Sessions>
```

注意: SessionInitiator の設定に関係なく SSO タグの設定は残しておくようにしてください。両方が必要です。

運用フェデレーションである場合、SessionInitiator の項目は以下のようになります。

```
    <SessionInitiator type="Chaining" Location="/DS" isDefault="true" id="DS">
        <SessionInitiator type="SAML2" template="bindingTemplate.html"/>
        <SessionInitiator type="Shib1"/>
        <SessionInitiator type="SAMLDS" URL="https://ds.gakunin.nii.ac.jp/WAYF"/>
    </SessionInitiator>
```

次に、Shibboleth SP で用いるメタデータの設定を行います。Shibboleth IdP と同様、テストフェデレーションと運用フェデレーションでは異なる URL と証明書を用います。

- テストフェデレーション
  - https://metadata.gakunin.nii.ac.jp/gakunin-test-metadata.xml
  - https://metadata.gakunin.nii.ac.jp/gakunin-test-signer-2011.cer
- 運用フェデレーション
  - https://metadata.gakunin.nii.ac.jp/gakunin-metadata.xml
  - https://metadata.gakunin.nii.ac.jp/gakunin-signer-2010.cer

テストフェデレーションのメタデータを利用する場合、shibboleth2.xml に以下のような設定を行います (gakunin-signer-2010.cer は所定の位置にダウンロードしておきます)

```
<MetadataProvider type="XML"
                  uri="https://metadata.gakunin.nii.ac.jp/gakunin-test-metadata.xml"
                  backingFilePath="federation-metadata.xml"
                  reloadInterval="7200">
    <MetadataFilter type="RequireValidUntil" maxValidityInterval="1296000"/>
    <MetadataFilter type="Signature"
                    certificate="/etc/shibboleth/cert/gakunin-test-signer-2011.cer"/>
</MetadataProvider>
```

運用フェデレーションのメタデータを利用する場合には、以下のような設定を行います。

```
<MetadataProvider type="XML"
                  uri="https://metadata.gakunin.nii.ac.jp/gakunin-metadata.xml"
                  backingFilePath="federation-metadata.xml"
                  reloadInterval="7200">
    <MetadataFilter type="RequireValidUntil" maxValidityInterval="1296000"/>
    <MetadataFilter type="Signature"
                    certificate="/etc/shibboleth/cert/gakunin-signer-2010.cer"/>
</MetadataProvider>
```

Shibboleth SP が暗号化、署名に用いる証明書、秘密鍵を以下のように設定します。

```
<CredentialResolver type="File"
                    key="/etc/pki/tls/private/server.key"
                    certificate="/etc/pki/tls/certs/server.crt"/>
```

注意: 鍵にパスワードが設定されている場合、password 属性の設定も必要です。

参考:

- https://meatwiki.nii.ac.jp/confluence/pages/viewpage.action?pageId=12158266
- https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPCredentialResolver

attribute-map.xml の設定

Shibboleth SP は、IdP から送られる属性値をローカルの環境変数に変換することで Moodle へ受け渡します。IdP からの属性値を環境変数へ適切にマッピングするため、attribute-map.xml に以下のような設定を行います。

```
<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:mail" id="mail"/>
    <Attribute name="urn:mace:dir:attribute-def:sn" id="sn"/>
    <Attribute name="urn:mace:dir:attribute-def:givenName" id="givenName"/>
    <Attribute name="urn:oid:2.5.4.42" id="givenName"/>
    <Attribute name="urn:oid:2.5.4.4" id="sn"/>
    <Attribute name="urn:oid:0.9.2342.19200300.100.1.3" id="mail"/>
    <Attribute name="urn:mace:dir:attribute-def:eduPersonAffiliation"
               id="unscoped-affiliation">
        <AttributeDecoder xsi:type="StringAttributeDecoder" caseSensitive="false"/>
    </Attribute>
    <Attribute name="urn:oid:1.3.6.1.4.1.5923.1.1.1.1" id="unscoped-affiliation">
        <AttributeDecoder xsi:type="StringAttributeDecoder" caseSensitive="false"/>
    </Attribute>
```

```
    <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>
</Attributes>
```

ここで、各 Attribute における name の値は SAML および学認で用いられる正式な属性値としてください。

Moodle に依存せずに shibd, httpd の動作確認を行う方法

Moodle を用いずに shibd と httpd が正しく動作しているかを確認するには、以下のようにします。この方法を利用するには、前もって SP となるサーバに php 等のパッケージがインストールされている必要があります。

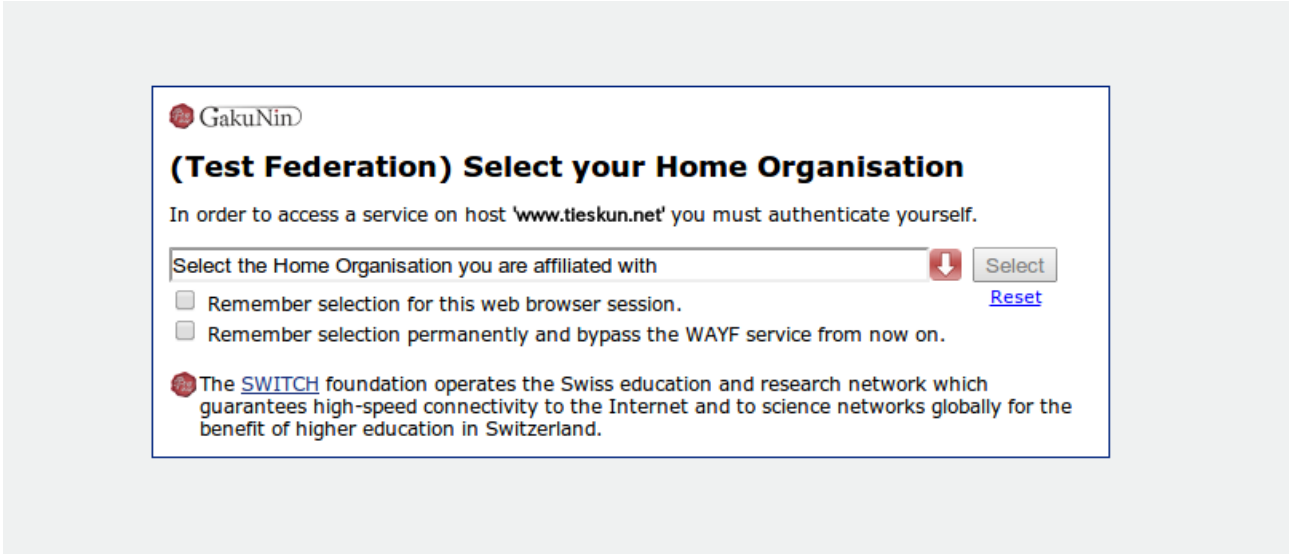
/etc/httpd/conf.d/shib.conf へ以下のような設定を追加します。

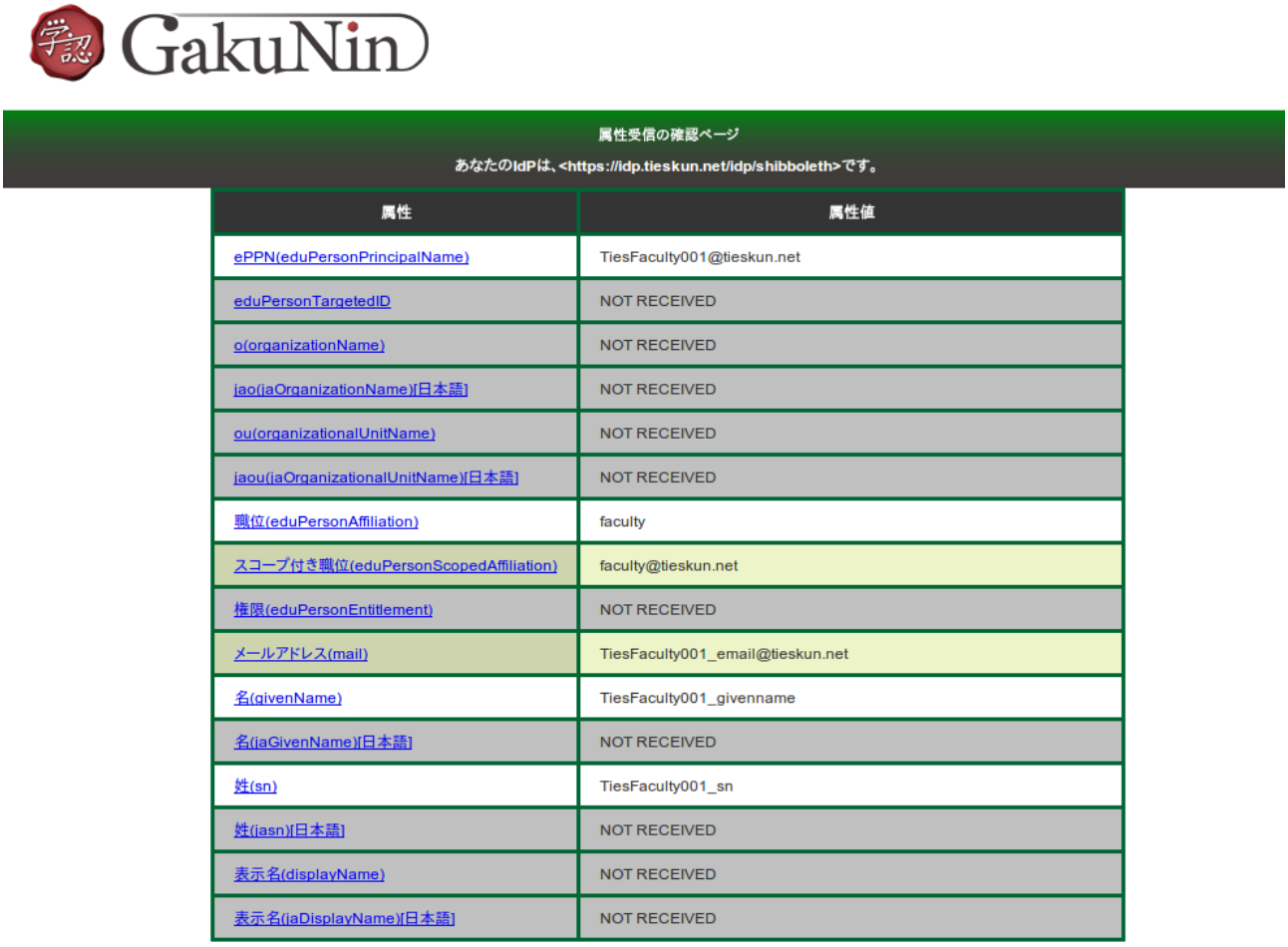
```
<Location /secure>
    AuthType shibboleth
    ShibRequestSetting requireSession 1
    require valid-user
</Location>
```

次に /var/www/html/secure ディレクトリを作成し、学認技術ガイドで提供されている属性情報確認用スクリプト (https://meatwiki.nii.ac.jp/confluence/pages/viewpage.action?pageId=13502421 からダウンロード出来ます) をそこに展開します。最終的に /var/www/html/secure/index.php が存在する状態にします。

その状態で https://sp.example.com/secure/phpinfo.php へアクセスします。

もし httpd が正しく動作していれば、ここで学認の Discovery Service を経由して IdP にログインして、その後に属性情報を表示する画面が表示されるはずです。





学認テストフェデレーションの Discovery Service (DS)
認証後の index.php 画面。IdP から提供された情報が表示される

Moodle 上の設定 (Shibboleth-TIES)

管理者アカウントにて、「サイト管理 >> プラグイン >> 認証 >> 認証管理」にて「Shibboleth-TIES」 モジュールを有効の上、モジュールの設定をおこなってください。

ユーザ名	eppn
名	givenName
姓	sn
メールアドレス	mail

注意: Shibboleth-TIES モジュールは、暗黙のうちに IdP から渡された unscoped-affiliation を参照し、そのアカウントが educator すなわちコース作成権限を持つものであるか、learner すなわちそうでないかを判定しています。Shibboleth-TIES の認証管理 UI 上で明示的に指定する必要はありません。

注意: ここで指定するフィールドは、Shibboleth SP の attribute-map.xml で書かれた SP 内での ID を指定します。例えば

```
<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>
```

という設定が Shibboelth SP の attribute-map.xml に書かれていた場合、eppn という id を用いることで、 eduPersonPrincipalName 属性値を Moodle から取得できます。

httpd と shibd を再起動するため、設定後に下記の通りのコマンドを実行します。

```
# service shibd restart
# service httpd restart
```

テストフェデレーションから運用フェデレーションへの移行設定

Shibboleth IdP におけるフェデレーション移行

relying-party.xml の設定におけるメタデータに関係する設定が異なります。

テストフェデレーションで用いるメタデータの URL と、証明書は以下から得られます。

- https://metadata.gakunin.nii.ac.jp/gakunin-test-metadata.xml
- https://metadata.gakunin.nii.ac.jp/gakunin-test-signer-2011.cer

一方運用フェデレーションではこれらが以下のように変わります。

- https://metadata.gakunin.nii.ac.jp/gakunin-metadata.xml
- https://metadata.gakunin.nii.ac.jp/gakunin-signer-2010.cer

フェデレーション切替時には上記の 2 つを変更する必要があります。

参考: テストフェデレーション申請時と運用フェデレーション申請時で、IdP 自身のサーバ証明書として異なるものを学認に提出した場合、httpd へ登録した証明書、Shibboleth IdP に指定した証明書をこの段階で切り替える必要も発生します。

Shibboleth SP におけるフェデレーション移行

IdP と同様、メタデータの切り替えが必要です。

また、Discovery Service をテストフェデレーションのものから運用フェデレーションのものに切り替える必要があります。

補足編

学認のフェデレーションに頼らない TIES V8 構築

「設定編」では学認を経由して TIES V8 をセットアップする方法について解説しました。

本節では、Shibboleth IdP、Shibboleth SP を用いつつも学認を経由せずに TIES V8 の設定を行う方法について説明します。

IdP・SP のメタデータの準備

まず、Shibboleth IdP と Shibboleth SP のメタデータをそれぞれ準備します。学認技術ガイドが提供するテンプレートを元に設定するのが便利です。

- IdP
  - IdP メタデータテンプレート:
    https://meatwiki.nii.ac.jp/confluence/pages/viewpage.action?pageId=12158233
  - Shibboleth 配布元 Wiki における解説:
    https://wiki.shibboleth.net/confluence/display/SHIB2/MetadataForIdP
- SP
  - SP メタデータテンプレート:
    https://meatwiki.nii.ac.jp/confluence/pages/viewpage.action?pageId=12158235
  - Shibboleth 配布元 Wiki における解説:
    https://wiki.shibboleth.net/confluence/display/SHIB2/MetadataForSP

以降では IdP のメタデータを収めたファイルと idp-metadata.xml とし、SP のメタデータを収めたファイルを sp-metadata.xml として説明します。

IdP・SP へのメタデータの登録

学認のメタデータを用いる代わりに、IdP では SP の、SP では IdP のメタデータを直接登録します。

Shibboleth IdP では relying-party.xml の設定を変更します。学認対応の設定では FileBackedHTTPMetadataProvider および TrustEngine を設定しましたが、今回はこれらの設定を行わず、ローカルのファイルをただ参照する FilesystemMetadataProvider を用います。sp-metadata.xml を /opt/shibboleth-idp/metadata/sp-metadata.xml へ配置したと仮定すると、以下のような設定になります。

```
<metadata:MetadataProvider id="ShibbolethMetadata"
                                    xsi:type="metadata:ChainingMetadataProvider">
    <metadata:MetadataProvider id="IdPMD"
                                        xsi:type="metadata:FilesystemMetadataProvider"
```

```
                                    metadataFile="/opt/shibboleth-idp/metadata/sp-metadata.xml"
                                    maxRefreshDelay="P1D" />
</metadata:MetadataProvider>
```

Shibboleth SP でも同様に idp-metadata.xml を設定します。

```
<MetadataProvider type="XML" file="/etc/shibboleth/idp-metadata.xml"/>
```

複数 SP・IdP に対応する場合

SP・IdP が複数ある場合には、以下の作業を行います

- 各 SP・IdP が信頼するべきホストを列挙したメタデータを作成・登録する
  - (学認のように全ての SP・IdP のメタデータを列挙した単一のメタデータを作成して良い)
- Discovery Service を設定する

個別の IdP・SP のメタデータは EntityDescriptor を使用していましたが、複数の IdP・SP のメタデータを統合する場合には、それらのメタデータを EntitiesDescriptor (複数形に注意) で囲います。例えば以下のような構造になるでしょう

```
<?xml version="1.0"?>
<md:EntitiesDescriptor xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
                       xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
                       xmlns:shibmd="urn:mace:shibboleth:metadata:1.0"
                       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                       xmlns:mdui="urn:oasis:names:tc:SAML:metadata:ui"
                       Name="SP1">
  <md:EntityDescriptor entityID="https://sp1.example.com/..">
  ..
  </md:EntityDescriptor>

  <md:EntityDescriptor entityID="https://sp2.example.com/..">
  ..
  </md:EntityDescriptor>

  <md:EntityDescriptor entityID="https://idp1.example.com/..">
  ..
  </md:EntityDescriptor>

  .. (その他の IdP・SP の設定)

</md:EntitiesDescriptor>
```

注意: XML の xmlns の扱いに注意してください。EntityDescriptor 等は urn:oasis:names:tc:SAML:2.0:metadata 名前空間にあるタグです。EntitiesDescriptor の attribute である xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata" が抜けていると、md:EntityDescriptor の "md" 名前空間を発見できないため、メタデータ全体が不正なもの

として Shibboleth がエラーを送出します。

SP から認証要求を行う際、複数の IdP からユーザに 1 つの IdP を選ばせる場合、Discovery Service を準備する必要があります。Shibboleth 配布元の下記ページを参考にして設定をおこなってください。

https://wiki.shibboleth.net/confluence/display/EDS10/Embedded+Discovery+Service

TIES V8 における属性値マッピングの一覧

TIES V8 で用いられる属性値とその IdP・SP・Moodle での扱いは以下の通りとなります。

IdP 内で使われる id	IdP・SP 間の通信での属性値 (SAML 2.0 での場合)	SP 内の変数	Moodle の設定値
eduPersonPrincipalName	urn:oid:1.3.6.1.4.1.5923.1.1.1.6	eppn	ユーザ名
mail	urn:oid:0.9.2342.19200300.100.1.3	mail	メールアドレス
sn	urn:oid:2.5.4.4	sn	姓
givenName	urn:oid:2.5.4.42	givenName	名
eduPersonAffiliation	urn:oid:1.3.6.1.4.1.5923.1.1.1.1	unscoped-affiliation	(educator/learner の切り分け)

- IdP においては、**eduPersonPrincipalName と mail を SP へ送ることが必須**です。
- sn、givenName が指定されていない場合には Moodle 上で空の姓名が設定されます。
- eduPersonAffiliation が指定された場合、値に応じた educator/learner の振り分けがされます。
  - faculty, staff → educator (コース作成権限あり)
  - 他 (student, member, 空) → learner
- eduPersonAffiliation の指定がない場合、learner としてユーザが登録されます。

自己署名されたサーバ証明書を持つ Active Directory 向け設定方法

本節では、Active Directory (AD)に関する設定の内、表題の状況に対処する方法の一例を示します。**ここで示す方法はあくまで AD への修正を行えない場合の迂回策であることをご理解ください。**

参考: 本節の説明は、本来信頼されていない CA により署名された証明書を用いている LDAP サーバに共通して適用できます。例えば自己署名サーバ証明書を用いる OpenLDAP サーバ、というケースでも症状と対処方法はほぼ同じです。

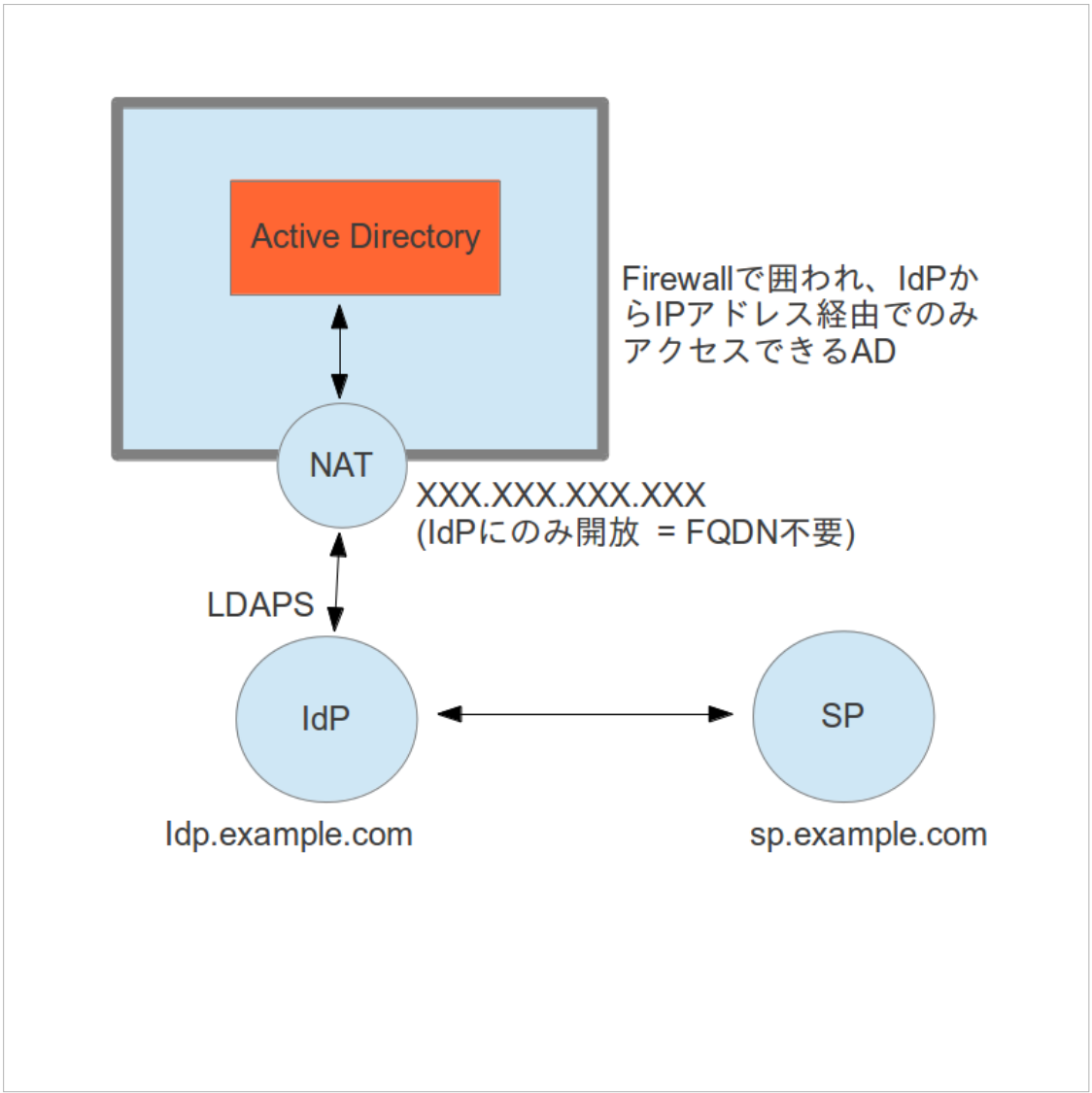
Shibboleth IdP と AD を連携させる際の情報については、例えば以下を参照してください

- 学認技術ガイド Active Directory における eduPerson スキーマの利用（成城大学提供）
  https://meatwiki.nii.ac.jp/confluence/pages/viewpage.action?pageId=12158440

- Microsoft Active Directory Configuration Issues
  https://wiki.shibboleth.net/confluence/pages/viewpage.action?pageId=4360293

本節で仮定する状況

下図に、本節で対処するサーバ環境を示します。



本節で想定する IdP、SP、AD の構成

- LDAPS 接続を許すポート (636 もしくは 3269) のみ NAT を介して開放されている
- 公開された DNS 上に AD へ接続するためのホスト名が登録されていない
- AD は Windows Server が標準で発行する自己署名のサーバ証明書を利用している
- IdP のみ AD へ接続出来るように、IP アドレスによる制限等がかかっている

上記の場合、SSL 接続において本来期待される「信頼できる CA によるサーバ証明書」を使うことができないため、Windows Server の出す自己署名サーバ証明書を、IdP が信頼する証明書として利用するよう、IdP 側で特殊な設定を行う必要があります。

この際、問題点となるのは以下の 2 点です:

- IdP 側のサーバが、標準では CA を信頼していません
- サーバ証明書のホスト名が、アクセスする実際のホスト名と一致しません

上図において、AD は固定的なホスト名 (例: ldap.example.com)を持ちません。そのため、AD 運用者は、IP アドレスでこのサーバにアクセスするよう、IdP 設定者に依頼します。しかし、IdP が LDAPS 接続を行おうとすると、そもそもサーバ証明書が信頼される発行機関のものではない上、しかも実際の接続先が、証明書に記載されているホスト名と異なるため、両方の理由で「この接続先は正しくない」と認識し、クライアント側が接続を断念します。

この状況に対処するため、２つの設定を IdP ホスト側に追加で行います。

1. 上記の Windows Server の CA 証明書を、新たに信頼されたものとして IdP に登録します。
2. #1 でなお接続が失敗する場合、AD が利用する自己署名されたサーバ証明書に記載されている仮のホスト名を /etc/hosts に記述し、その名前を使って AD へアクセスします

CA 証明書の取得と確認

まず、CA 証明書を準備します。これは Windows Server の管理者から取得する必要があります。

Windows Server が発行する自身の CA 証明書は、複数の異なる形式でエクスポートすることが可能です。一方、今回説明する方法では X.509 の証明書形式である必要があります。Windows Server の管理者「PEM 形式でエンコードされた X.509 の CA 証明書」を発行するよう依頼してください。

PEM 形式でエンコードされた X.509 とは、例えば以下のような形になります。

```
-----BEGIN CERTIFICATE-----
MIIDeDCCAmCgAwIBAgIQeJf9Y5wkzqVIOdjpf3CHzzANBgkqhkiG9w0BAQsFADBE
MRUwEwYKCZImiZPyLGQBGRYFbG9jYWwxFDASBgoJkiaJk/IsZAEZFgR0dW5lMRUw
EwYDVQQDEwx0dW5lLUFEMDEtQ0EwHhcNMTMwMzAxMDEyMDI0WhcNMTgwMzAxMDEz
MDIzWjBEMRUwEwYKCZImiZPyLGQBGRYFbG9jYWwxFDASBgoJkiaJk/IsZAEZFgR0
dW5lMRUwEwYDVQQDEwx0dW5lLUFEMDEtQ0EwggEiMA0GCSqGSIb3DQEBAQUAA4IB
DwAwggEKAoIBAQCcBG6eLaYYJSPgJBCPw6VM4WJeQXlr7sgb59rgJKtKvZAaPgVY
OOBcY63UNE51ZCm74wnyCcYzqNOR93z4qGgmHVlJsDX0Jnzl6lDCW9byfynEiWLb
7AqkRRzPw8FmO/nVRuuRtC4JRgqhMO0iQHOsuqvuAPPkef5FcUMoxBSO82FBjKE+
CFUg05g2FIK0a+ak70TU61tvjsPCcaKzQSk5nxjdYcKK+TKzYOUAXZNB5F4VBhaD
CmPR1IWcDZtE3UfnQI5XbZ6vXEp0Yk2bN2cLa7LfVzMvIlzNRu6PYVUo+NLzpPsW
iiuj6Xdkeqb0sK6R4CrABPLdd+uNhXH2LIP7AgMBAAGjZjBkMBMGCSsGAQQBgjcU
AgQGHgQAQwBBMAsGA1UdDwQEAwIBhjAPBgNVHRMBAf8EBTADAQH/MB0GA1UdDgQW
BBQx2tlzfZyuW/tZF9hNRrvglZA7eTAQBgkrBgEEAYI3FQEEAwIBADANBgkqhkiG
9w0BAQsFAAOCAQEAIlSTbJ2P6NfXxvQbErY85IdFwUAbXCl9xnHIkW6f8R3QGhcQ
KWpSa3yQZlVc5uhmlrAciRAKJhg5kSU2CTDx27REEnrfgB/4xChN7pM54uPQ//j/
```

```
FVAFUA5fI0eSrljuY9CCKtR2fxCOnUa8g9j/ikHn99dME5H5VQZxGDFRPoPfTflp
KBBRkmk05vKEE8oCKJ/AswGNGzc66JfVS+yJPfW6mflSHDeKCHZqpZuuKqahs2p0
DeYkzN54xxAXQ28vS3gLLJOiJgRz0S1A93xTMKireRr0/MJR+h/kfdH/yZzFRJvi
C7MrXSz/9J1S8x+eFwhLRaF4Ez70mZSoUdLHYg==
-----END CERTIFICATE-----
```

参考: 受け取った証明書が人が読めないフォーマットで記述されていた場合、PEM 形式ではなく DER 形式である可能性があります。

上記の X509 の証明書の中身を確認するには、例えば以下のように入力します。

```
# openssl x509 -in /etc/openldap/certs/tune-ca01-ca.crt -text -noout
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            78:97:fd:63:9c:24:ce:a5:48:39:d8:e9:7f:70:87:cf
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: DC=local, DC=tune, CN=tune-AD01-CA
        Validity
            Not Before: Mar  1 01:20:24 2013 GMT
            Not After : Mar  1 01:30:23 2018 GMT
        Subject: DC=local, DC=tune, CN=tune-AD01-CA
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:9c:04:6e:9e:2d:a6:18:25:23:e0:24:10:8f:c3:
                    a5:4c:e1:62:5e:41:79:6b:ee:c8:1b:e7:da:e0:24:
                    ab:4a:bd:90:1a:3e:05:58:38:e0:5c:63:ad:d4:34:
                    4e:75:64:29:bb:e3:09:f2:09:c6:33:a8:d3:91:f7:
                    7c:f8:a8:68:26:1d:59:49:b0:35:f4:26:7c:e5:ea:
                    50:c2:5b:d6:f2:7f:29:c4:89:62:db:ec:0a:a4:45:
                    1c:cf:c3:c1:66:3b:f9:d5:46:eb:91:b4:2e:09:46:
                    0a:a1:30:ed:22:40:73:ac:ba:ab:ee:00:f3:e4:79:
                    fe:45:71:43:28:c4:14:8e:f3:61:41:8c:a1:3e:08:
                    55:20:d3:98:36:14:82:b4:6b:e6:a4:ef:44:d4:eb:
                    5b:6f:8e:c3:c2:71:a2:b3:41:29:39:9f:18:dd:61:
                    c2:8a:f9:32:b3:60:e5:00:5d:93:41:e4:5e:15:06:
                    16:83:0a:63:d1:d4:85:9c:0d:9b:44:dd:47:e7:40:
                    8e:57:6d:9e:af:5c:4a:74:62:4d:9b:37:67:0b:6b:
                    b2:df:57:33:2f:22:5c:cd:46:ee:8f:61:55:28:f8:
                    d2:f3:a4:fb:16:8a:2b:a3:e9:77:64:7a:a6:f4:b0:
                    ae:91:e0:2a:c0:04:f2:dd:77:eb:8d:85:71:f6:2c:
                    83:fb
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            1.3.6.1.4.1.311.20.2:
                ...C.A
            X509v3 Key Usage:
                Digital Signature, Certificate Sign, CRL Sign
            X509v3 Basic Constraints: critical
                CA:TRUE
```

```
            X509v3 Subject Key Identifier:
                31:DA:D9:73:7D:9C:AE:5B:FB:59:17:D8:4D:46:BB:E0:95:90:3B:79
            1.3.6.1.4.1.311.21.1:
                ...
    Signature Algorithm: sha256WithRSAEncryption
         22:54:93:6c:9d:8f:e8:d7:d7:c6:f4:1b:12:b6:3c:e4:87:45:
         c1:40:1b:5c:29:7d:c6:71:c8:91:6e:9f:f1:1d:d0:1a:17:10:
         29:6a:52:6b:7c:90:66:55:5c:e6:e8:66:96:b0:1c:89:10:0a:
         26:18:39:91:25:36:09:30:f1:db:b4:44:12:7a:df:80:1f:f8:
         c4:28:4d:ee:93:39:e2:e3:d0:ff:f8:ff:15:50:05:50:0e:5f:
         23:47:92:ae:58:ee:63:d0:82:2a:d4:76:7f:10:8e:9d:46:bc:
         83:d8:ff:8a:41:e7:f7:d7:4c:13:91:f9:55:06:71:18:31:51:
         3e:83:df:4d:f9:69:28:10:51:92:69:34:e6:f2:84:13:ca:02:
         28:9f:c0:b3:01:8d:1b:37:3a:e8:97:d5:4b:ec:89:3d:f5:ba:
         99:f9:52:1c:37:8a:08:76:6a:a5:9b:ae:2a:a6:a1:b3:6a:74:
         0d:e6:24:cc:de:78:c7:10:17:43:6f:2f:4b:78:0b:2c:93:a2:
         26:04:73:d1:2d:40:f7:7c:53:30:a8:ab:79:1a:f4:fc:c2:51:
         fa:1f:e4:7d:d1:ff:c9:9c:c5:44:9b:e2:0b:b3:2b:5d:2c:ff:
         f4:9d:52:f3:1f:9e:17:08:4b:45:a1:78:13:3e:f4:99:94:a8:
         51:d2:c7:62
```

コマンドラインの詳細は "man x509" 等で参照してください。

なお、Windows Server でのエクスポート方法によっては、X.509 ではなく PKCS7 の証明書の発行を受けることがあります。PEM 形式でエンコードされた PKCS7 は、例えば以下のような形になります。

```
MIIMHQYJKoZIhvcNAQcCoIIMDjCCDAoCAQMxDzANBglghkgBZQMEAgEFADCCAWcG
CCsGAQUFBwwDoIIBWQSCAVUwggFRMIIBSTCCAQECAQEGCCsGAQUFBwcBMYHxMIHu
AgEAMAMCAQEMgePnmbrooYzjgZXjgozjgabjgYTjgb7jgZkgIDB4ODAwOTQwMDQs
IOWFpeS8muiAhSAoQ049QWRtaW5pc3RyYXRvcixDTj1Vc2VycyxEQz10dW5lLERD
PWxvY2FsKSDjga8gQWN0aXZlIERpcmVjdG9yeSDjgavpm7vlrZDjg6Hjg7zjg6vl
kI3jgYznmbvpjLLjgZXjgozjgabjgYTjgb7jgZvjgpPjgILpm7vlrZDjg6Hjg7zj
g6vlkI3jga/oqLzmmI7mm7jjgavov73liqDjgZXjgozjgb7jgZvjgpPjgIINCjBC
AgECBgorBgEEAYI3CgoBMTEwLwIBADADAgEBMSUwIwYJKwYBBAGCNxURMRYEFE0P
GUiYTzv6j0du7G1MBrnVdpfDMAAwAKCCCLQwggN4MIICYKADAgECAhB4l/1jnCTO
pUg52Ol/cIfPMA0GCSqGSIb3DQEBCwUAMEQxFTATBgoJkiaJk/IsZAEZFgVsb2Nh
bDEUMBIGCgmSJomT8ixkARkWBHR1bmUxFTATBgNVBAMTDHR1bmUtQUQwMS1DQTAe
Fw0xMzAzMDEwMTIwMjRaFw0xODAzMDEwMTMwMjNaMEQxFTATBgoJkiaJk/IsZAEZ
FgVsb2NhbDEUMBIGCgmSJomT8ixkARkWBHR1bmUxFTATBgNVBAMTDHR1bmUtQUQw
MS1DQTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAJwEbp4tphglI+Ak
EI/DpUzhYl5BeWvuyBvn2uAkq0q9kBo+BVg44FxjrdQ0TnVkKbvjCfIJxjOo05H3
fPioaCYdWUmwNfQmfOXqUMJb1vJ/KcSJYtvsCqRFHM/DwWY7+dVG65G0LglGCqEw
7SJAc6y6q+4A8+R5/kVxQyjEFI7zYUGMoT4IVSDTmDYUgrRr5qTvRNTrW2+Ow8Jx
orNBKTmfGN1hwor5MrNg5QBdk0HkXhUGFoMKY9HUhZwNm0TdR+dAjldtnq9cSnRi
TZs3Zwtrst9XMy8iXM1G7o9hVSj40vOk+xaKK6Ppd2R6pvSwrpHgKsAE8t13642F
cfYsg/sCAwEAAaNmMGQwEwYJKwYBBAGCNxQCBAYeBABDAEEwCwYDVR0PBAQDAgGG
MA8GA1UdEwEB/wQFMAMBAf8wHQYDVR0OBBYEFDHa2XN9nK5b+1kX2E1Gu+CVkDt5
MBAGCSsGAQQBgjcVAQQDAgEAMA0GCSqGSIb3DQEBCwUAA4IBAQAiVJNsnY/o19fG
9BsStjzkh0XBQBtcKX3GcciRbp/xHdAaFxApalJrfJBmVVzm6GaWsByJEAomGDmR
JTYJMPHbtEQSet+AH/jEKE3ukzni49D/+P8VUAVQDl8jR5KuWO5j0IIq1HZ/EI6d
RryD2P+KQef310wTkflVBnEYMVE+g99N+WkoEFGSaTTm8oQTygIon8CzAY0bNzro
l9VL7Ik99bqZ+VIcN4oIdmqlm64qpqGzanQN5iTM3njHEBdDby9LeAssk6ImBHPR
LUD3fFMwqKt5GvT8wlH6H+R90f/JnMVEm+ILsytdLP/0nVLzH54XCEtFoXgTPvSZ
lKhR0sdiMIIFNDCCBBygAwIBAgIKcZ/HSQAAAAAADjANBgkqhkiG9w0BAQsFADBE
```

```
MRUwEwYKCZImiZPyLGQBGRYFbG9jYWwxFDASBgoJkiaJk/IsZAEZFgR0dW5lMRUw
EwYDVQQDEwx0dW5lLUFEMDEtQ0EwHhcNMTMwMzE4MDYyMzI3WhcNMTQwMzE4MDYy
MzI3WjBVMRUwEwYKCZImiZPyLGQBGRYFbG9jYWwxFDASBgoJkiaJk/IsZAEZFgR0
dW5lMQ4wDAYDVQQDEwVVc2VyczEWMBQGA1UEAxMNQWRtaW5pc3RyYXRvcjCBnzAN
BgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEAmw70c24sxT18fZylb4Xdic5opUf6vrca
Vy9JUHZjtMN6pXRVqwPYke0qqvsUCPrKajLSs2lJCcs3dZbXQnlf2jsHQpdjBQML
qkXL/rN7oyYVRSzmpt3TbUfzN+WOJEGVZkbEpLURDEOrrf63XdTOUu93BIrcGsQr
rQxfrapxTWMCAwEAAaOCApkwggKVMBcGCSsGAQQBgjcUAgQKHggAVQBzAGUAcjAd
BgNVHQ4EFgQUgAN0Fx+W/V2BqNmJbFSXNvv0cWYwCwYDVR0PBAQDAgWgMB8GA1Ud
IwQYMBaAFDHa2XN9nK5b+1kX2E1Gu+CVkDt5MIHGBgNVHR8Egb4wgbswgbiggbWg
gbKGga9sZGFwOi8vL0NOPXR1bmUtQUQwMS1DQSxDTj1BRDAxLENOPUNEUCxDTj1Q
dWJsaWMlMjBLZXklMjBTZXJ2aWNlcyxDTj1TZXJ2aWNlcyxDTj1Db25maWd1cmF0
aW9uLERDPXR1bmUsREM9bG9jYWw/Y2VydGlmaWNhdGVSZXZvY2F0aW9uTGlzdD9i
YXNlP29iamVjdENsYXNzPWNSTERpc3RyaWJ1dGlvblBvaW50MIG9BggrBgEFBQcB
AQSBsDCBrTCBqgYIKwYBBQUHMAKGgZ1sZGFwOi8vL0NOPXR1bmUtQUQwMS1DQSxD
Tj1BSUEsQ049UHVibGljJTIwS2V5JTIwU2VydmljZXMsQ049U2VydmljZXMsQ049
Q29uZmlndXJhdGlvbixEQz10dW5lLERDPWxvY2FsP2NBQ2VydGlmaWNhdGU/YmFz
ZT9vYmplY3RDbGFzcz1jZXJ0aWZpY2F0aW9uQXV0aG9yaXR5MCkGA1UdJQQiMCAG
CisGAQQBgjcKAwQGCCsGAQUFBwMEBggrBgEFBQcDAjAzBgNVHREELDAqoCgGCisG
AQQBgjcUAgOgGgwYYWRtaW5pc3RyYXRvckB0dW5lLmxvY2FsMEQGCSqGSIb3DQEJ
DwQ3MDUwDgYIKoZIhvcNAwICAgCAMA4GCCqGSIb3DQMEAgIAgDAHBgUrDgMCBzAK
BggqhkiG9w0DBzANBgkqhkiG9w0BAQsFAAOCAQEAg14G5O/K46Yhyz1QK5+oRX4z
Muz5wmWmSwPT06sF33PpMift4wd07GvDNvadzadDeFyTtopYd/ngrVr5mrYl3XiC
PinatOZ9WyJpFcB7naUHJV2FhuoS9ait/NiaOVuBmUdhnmjssPmhc4nn5Xmk2QE5
jD8Nu+L5km8Uz2CCYK6qW52AKAd4bc04oSKt2lulyx6p6U4KGIa6A892dEQpu2PO
PjZCWQYVExtLyXf1vf+iDmMIr+fKJew9o6pFoDVJJ1XeW+NSUFf53HiCTtoulh1l
eCI18KjWRMuvuz2JtoSkzTN/lVIV1GaOb+wyfHpZeiBAsUe6s9+lSiYAaRPjQjGC
Ac8wggHLAgEBMFgwRDEVMBMGCgmSJomT8ixkARkWBWxvY2FsMRQwEgYKCZImiZPy
LGQBGRYEdHVuZTEVMBMGA1UEAxMMdHVuZS1BRDAxLUNBAhB4l/1jnCTOpUg52Ol/
cIfPMA0GCWCGSAFlAwQCAQUAoEowFwYJKoZIhvcNAQkDMQoGCCsGAQUFBwwDMC8G
CSqGSIb3DQEJBDEiBCCjOUFzwpcVe8+SrSFA9JXESKD2gz+7WivL6LLZxERmYzAN
BgkqhkiG9w0BAQEFAASCAQAr1Vb5xVxKN1TRWV9UA8dRJZxDyOTfXPTIPo0ShIvV
zoVsN+ir4l1OzlqJAUJUkwxEvU4DPVYge0zrZpCpnc/6OJngvGwQn/XlOAg57Zg9
gOWgTlhBK40Wlyszo9DMFIYfQZhzZna0A0bXGsPLix+bP8r+KmkAbaBz1w3Vhj8X
qmNx99kk16+UfSGqxptvuS9ePWRGC17Z35MizLl9zlRjchXakykLRJDYLx+rsBQu
1wC2TrZOJ6pZ7upLWX+wshXY2ALUCaBY0j18fixwzSpvmDIqacJJ8mNbpS5IqF8V
Sx91+pErs3yUHN77vf+07V+5D40UzBS1L6Wy7b5JskZu
```

PKCS7 と思われる証明書を渡された際には、openssl コマンドで X.509 の証明書を取得できる場合があります。上記の PKCS7 の証明書に、次の通りヘッダとフッタを追加し:

```
-----BEGIN PKCS7-----
MIIMHQYJKoZIhvcNAQcCoIIMDjCCDAoCAQMxDzANBglghkgBZQMEAgEFADCCAWcG
CCsGAQUFBwwDoIIBWQSCAVUwggFRMIIBSTCCAQECAQEGCCsGAQUFBwcBMYHxMIHu
AgEAMAMCAQEMgePnmbrooYzjgZXjgozjgabjgYTjgb7jgZkgIDB4ODAwOTQwMDQs
..(省略)..
1wC2TrZOJ6pZ7upLWX+wshXY2ALUCaBY0j18fixwzSpvmDIqacJJ8mNbpS5IqF8V
Sx91+pErs3yUHN77vf+07V+5D40UzBS1L6Wy7b5JskZu
-----END PKCS7-----
```

下記のような openssl コマンドによって X.509 証明書の取得を試みます。

```
# openssl pkcs7 -in pkcs7.txt -print_certs
```

Subject (身元を証明したい対象) が自分自身である証明書が、今回必要となる CA 証明書です。

```
subject=/DC=local/DC=tune/CN=tune-AD01-CA
issuer=/DC=local/DC=tune/CN=tune-AD01-CA
-----BEGIN CERTIFICATE-----
MIIDeDCCAmCgAwIBAgIQeJf9Y5wkzqVIOdjpf3CHzzANBgkqhkiG9w0BAQsFADBE
MRUwEwYKCZImiZPyLGQBGRYFbG9jYWwxFDASBgoJkiaJk/IsZAEZFgR0dW5lMRUw
EwYDVQQDEwx0dW5lLUFEMDEtQ0EwHhcNMTMwMzAxMDEyMDI0WhcNMTgwMzAxMDEz
MDIzWjBEMRUwEwYKCZImiZPyLGQBGRYFbG9jYWwxFDASBgoJkiaJk/IsZAEZFgR0
dW5lMRUwEwYDVQQDEwx0dW5lLUFEMDEtQ0EwggEiMA0GCSqGSIb3DQEBAQUAA4IB
DwAwggEKAoIBAQCcBG6eLaYYJSPgJBCPw6VM4WJeQXlr7sgb59rgJKtKvZAaPgVY
OOBcY63UNE51ZCm74wnyCcYzqNOR93z4qGgmHVlJsDX0Jnzl6lDCW9byfynEiWLb
7AqkRRzPw8FmO/nVRuuRtC4JRgqhMO0iQHOsuqvuAPPkef5FcUMoxBSO82FBjKE+
CFUg05g2FIK0a+ak70TU61tvjsPCcaKzQSk5nxjdYcKK+TKzYOUAXZNB5F4VBhaD
CmPR1IWcDZtE3UfnQI5XbZ6vXEp0Yk2bN2cLa7LfVzMvIlzNRu6PYVUo+NLzpPsW
iiuj6Xdkeqb0sK6R4CrABPLdd+uNhXH2LIP7AgMBAAGjZjBkMBMGCSsGAQQBgjcU
AgQGHgQAQwBBMAsGA1UdDwQEAwIBhjAPBgNVHRMBAf8EBTADAQH/MB0GA1UdDgQW
BBQx2tlzfZyuW/tZF9hNRrvglZA7eTAQBgkrBgEEAYI3FQEEAwIBADANBgkqhkiG
9w0BAQsFAAOCAQEAIlSTbJ2P6NfXxvQbErY85IdFwUAbXCl9xnHIkW6f8R3QGhcQ
KWpSa3yQZlVc5uhmlrAciRAKJhg5kSU2CTDx27REEnrfgB/4xChN7pM54uPQ//j/
FVAFUA5fI0eSrljuY9CCKtR2fxCOnUa8g9j/ikHn99dME5H5VQZxGDFRPoPfTflp
KBBRkmk05vKEE8oCKJ/AswGNGzc66JfVS+yJPfW6mflSHDeKCHZqpZuuKqahs2p0
DeYkzN54xxAXQ28vS3gLLJOiJgRz0S1A93xTMKireRr0/MJR+h/kfdH/yZzFRJvi
C7MrXSz/9J1S8x+eFwhLRaF4Ez70mZSoUdLHYg==
-----END CERTIFICATE-----
```

規格が分かっている証明書については、openssl コマンドラインツール等によって、最終的に x509 の証明書に変換できる場合があります。しかし Windows Server で起こり得る状況は多様であるため、証明書の詳細については Windows Server の管理者に問い合わせておく必要がある点は変わりません。

AD の用いるサーバ証明書の確認

openssl には SSL 接続を試す s_client サブコマンドが存在します (詳細は "man s_client" で確認してください)。例えば IdP から XXX.XXX.XXX.XXX という IP アドレスの 636 番ポートへ接続テストを行う場合には以下のようにします。

```
# openssl s_client -connect XXX.XXX.XXX.XXX:636
(終了するには Ctrl-C を入力してください)
```

s_client サブコマンドは指定されたホストへ接続し、SSL 接続時に得られるサーバ証明書を検証します。telnet コマンド同様実際の通信を行うこともできますが、本試験では利用しませんので、接続時の情報を取得した時点で Ctrl-C で接続を切断します。

接続先の提供するサーバ証明書が適切でない場合、例えば下記のようなエラーメッセージが表示されます。

```
Verify return code: 21 (unable to verify the first certificate)
```

AD が提供するサーバ証明書も表示されるので、openssl x509 サブコマンド等で内容を確認します。

本節の仮定と一致する場合、ここで提供される AD のサーバ証明書の発行者が、自己署名の CA 証明書のそれと同じであるはずです。

```
issuer=/DC=local/DC=tune/CN=tune-AD01-CA
```

この場合、上記の CA 証明書を openssl が信用すれば、接続は成功することが予想されます。CA 証明書を ca.crt とした場合、コマンドラインを以下のように変更して接続を試みてください。

```
# openssl s_client -CAfile ca.crt    -connect 160.244.10.100:636
```

これでなお接続が失敗する場合には、証明書の指定するホスト名でアクセスを試みます。たとえばここでは ad01.tune.local へ接続するものと仮定すると、/etc/hosts にその(本来正しくない)ホスト名を書くことによって、名前解決を行わせた上で、以下のように入力します。

```
# openssl s_client -CAfile ca.crt    -connect ad01.tune.local:636
```

サーバ証明書が信頼された発行機関からのものだと判定された場合、openssl s_client は下記のように返答します。

```
Verify return code: 0 (ok)
```

以上で CA 証明書とその CA から発行されたサーバ証明書を信頼する方法を示しました。この時点でもまだエラーが発生する場合には、サーバ管理者と協議して、前もって問題を解決してから次に移行してください。

参考: 運用中に接続エラーが発生しはじめた、という場合にも、この節と同様の接続テストを行い、証明書が変更されていないかを確認してください。

Shibboleth IdP での設定変更

openssl 等のコマンドラインによって当該ホストから AD へのアクセスが成功した状態で、Shibboleth IdP で同等の機能を有する設定を行います。

参考: /etc/hosts に書かれたホスト名に関する名前解決の情報は Shibboleth IdP (Java)でも有効です。

JDK に自己署名された CA 証明書を信頼させるためには、例えば下記のように入力しま

す。今回利用している環境においては、keystore のパスワードは変更されていない状態では"changeit"です。**この作業は JDK をアップデートした際には毎回行う必要があります。**

```
# keytool -import -trustcacerts -alias "tune-local-ca" -file /etc/openldap/certs/tune-ca.crt -keystore
/usr/java/jdk1.7.0_67jre/lib/security/cacerts
(keystore のパスワードは標準では  changeit)
```

その上で、正しいサーバ証明書を持つ LDAPS と同様の設定を行います。login.config は例えば以下のようになるでしょう (bindDN や bindCredential も環境毎に変わります)。

参考: Microsoft Global Catalog ポートを利用する場合、636 番ポートではなく 3269 番ポートを利用します。

```
edu.vt.middleware.ldap.jaas.LdapLoginModule required
    ldapUrl="ldaps://ad01.tune.local:636"
    bindDn="admin@tune.local"
    bindCredential="admin"
    ssl="true"
    baseDn="dc=tune,dc=local"
    userFilter="sAMAccountName={0}"
    subtreeSearch="true";
```

併せて、attribute-resolver.xml の設定も行います。

```
<resolver:DataConnector id="myLDAP"
      xsi:type="dc:LDAPDirectory"
      ldapURL="ldaps://ad01.tune.local:636"
      baseDN="dc=tune,dc=local"
      principal="admin@tune.local"
      principalCredential="admin">
      <dc:FilterTemplate>
            <![CDATA[
                  (sAMAccountName=$requestContext.principalName)
            ]]>
      </dc:FilterTemplate>
</resolver:DataConnector>
```