



SERAPIS Project

Installation & Configuration of a Shibboleth 1.3 Identity Provider on SuSE Linux with CAS SSO

**Arts and Humanities Data Service
King's College London**

Author: Sanjay Vivek

Contact: Mark Hedges

7 March 2007

Table Of Contents

1	Introduction	4
2	Shibboleth IdP Installation	5
2.1	Prerequisites	5
2.2	CAS Server.....	6
2.3	Shibboleth IdP 1.3 and Java CAS client	7
2.3.1	Deploy Java CAS client as Shibboleth IdP extension	7
2.4	Customizing Existing Views.....	10
2.4.1	Customizing the Views for the AHDS	10
2.5	Shibboleth IdP 1.3.....	11
2.6	LDAP Configuration.....	11
3	PKI Configuration.....	13
3.1	Generate certificate	13
3.2	Updating Server Certificates for Shibboleth.....	14
3.3	Updating the CA Root certificates	15
3.4	Tomcat configuration	17
4	Configuring IdP to join the SDSS Federation.....	19
4.1	Application letter to join SDSS.....	19
4.2	Uploading SDSS metadata file.....	19
4.3	Editing idp.xml.....	20
4.4	Editing resolver.ldap.xml	21
5	Testing.....	23

1 Introduction

This guide describes the installation procedure followed at the AHDS to install a Shibboleth Identity Provider v1.3 and to configure it to join the SDDS Federation. It provides details on installing with Tomcat 5.5 and the CAS Single Sign On system. More detailed information can be found in the [Shibboleth Identity Deployment Guide](#) from Internet2.

Before joining the SDSS Federation, the Shibboleth IdP has to be installed and a X.509 certificate has to be obtained. Once these two tasks are completed, the IdP has to be configured to join the SDSS Federation. Consequently, this document consists of three parts:

1. Installing the Shibboleth IdP.
2. PKI Installation.
3. Configuring the IdP to join the SDSS Federation.

2 Shibboleth IdP Installation

The Shibboleth Identity Provider (IdP) 1.3 is implemented in Java and can be run as a Java web application in Tomcat. The IdP consists of the two servlets SSO (Single Sign On) and AA (Attribute Authority). The Shibboleth IdP however includes only rudimentary SSO functionality, so authentication is instead handled outside the IdP either by means of a separate SSO system (e.g. CAS for Tomcat or Pubcookie for Apache) or Tomcat authentication. CAS was chosen as the SSO system in this installation. One benefit of CAS is that the Apache (httpd) application is not used. Tomcat is configured to listen on ports 443 and 8443, thus avoiding the need to use mod_jk2 or Apache for the Shibboleth IdP. This further frees up Apache for use as the Shibboleth SP interface.

We carried out two separate installations of the IdP: the first IdP installation was for testing purposes while the second IdP was our live IdP. Our live IdP installation was carried out using the quickstart package from K.U.Leuven, which contains all the components required to install an IdP. This saved us considerable time and effort and is available at <http://shib.kuleuven.be/download/idp/1.3/>. The installation and configuration of this quickstart package is relatively straightforward.

However, our first (test) IdP installation was carried out manually. This report describes this procedure, as it provides a clearer picture of the process and the individual components installed.

2.1 Prerequisites

This guide applies to a setup in a SuSE Linux environment. Given below is a summary of the packages that should be installed on your system and where you can find them:

- jdk (1.5.0) <http://java.sun.com/j2se/1.5.0/download.jsp>
- Apache Tomcat (5.5) http://jakarta.apache.org/site/downloads/downloads_tomcat-5.cgi
- Apache commons-logging (1.0) http://jakarta.apache.org/site/downloads/downloads_commons-logging.cgi
- log4j (1.2) <http://logging.apache.org/site/binindex.cgi>
- Apache ant (1.6) <http://ant.apache.org/bindownload.cgi>
- esup-cas-server (2.0) <http://prdownloads.sourceforge.net/esup-casgeneric/>
- cas-client java filter (2.1) <http://jasigch.princeton.edu:9000/display/CAS/Java+CAS+Client+2.1.1>
- Shibboleth IdP (1.3) <http://shibboleth.internet2.edu/downloads/>

The firewall also has to be configured for inbound traffic and the following ports have to be opened:

- webserver: port **443** is used by any browser-user for both the IdP and CAS.
- every Shibboleth SP has to connect to the IdP on port **8443** to fetch attributes/dereference the artifact.

Installation details are provided for the esup-cas-server, cas-client, and Shibboleth IdP as described below.

2.2 CAS Server

The Shibboleth IdP does not include SSO functionality or a means of authenticating a person, so authentication has to be handled outside in a separate SSO system. The SSO systems that can work with Shibboleth include [CAS](#) and [Pubcookie](#). We chose to use CAS (Central Authentication Service) with Tomcat for this purpose. Using a plugin called [GenericHandler](#) from ESUP, users can authenticate themselves against various backends like LDAP, SQL databases, NIS, Kerberos, etc. LDAP was chosen as the backend for our purposes.

Note that [LDAP](#) and [server certificates](#) have to be preconfigured before commencing this part of the process.

1. Get the esup-cas-server from <http://prdownloads.sourceforge.net/esup-casgeneric/>
2. Extract the file esup-cas-server-2.0.6-1.zip in directory /opt:

```
$ cd /opt
$ unzip esup-cas-server-2.0.6-1.zip
```

The package content can be found in directory **/opt/esup-cas-server-2.0.6-1/**. The documentation in HTML format is in the subdirectory **/docs**.

```
#.....

#=====
# Complex (search, then bind) LDAP authentication
esup-casgeneric.auth=ldap-search
esup-casgeneric.auth.ldap-search.filter=uid=%u
esup-casgeneric.auth.ldap-search.search-
base=ou=People,dc=ahds,dc=ac,dc=uk
esup-casgeneric.auth.ldap-search.scope=sub
esup-casgeneric.auth.ldap-search.bind-
dn=cn=Manager,dc=ahds,dc=ac,dc=uk
esup-casgeneric.auth.ldap-search.bind-password=password
esup-casgeneric.auth.ldap-search.url=ldap://reto.ahds.ac.uk:389
```

```
#...
#=====
# CAS Generic Handler log (log4j) level
# Set level to DEBUG for testing purposes
#
esup-casgeneric.log.level=FINE
esup-casgeneric.log.path=C:/usr/local/shibboleth-idp/logs/esup-
casgeneric.log
```

3. Configure your authentication backend as documented in [docs/auth-high-level.html](#) or online on <http://esup-casgeneric.sourceforge.net/auth-high-level.html> This can be done in the file [properties/build.properties](#). An example for an LDAP backend configuration as shown above.
4. Deploy the CAS application:

```
$ cd /opt/esup-cas-server-2.0.5-2/
$ ant deploy
```

5. Customize the CAS layout in [/opt/tomcat/webapps/cas/](#) to meet your design needs.
6. The CAS login page should now be available at <http://localhost:8080/cas/login> You can then test the authentication against the configured backend, which in our case is LDAP. If you encounter problems connecting, it is likely to result from problems with communicating with the authentication backend.

2.3 Shibboleth IdP 1.3 and Java CAS client

This component passes the identity of the authenticated user to Shibboleth. It is implemented as a Java Server Filter and can be placed on any Java webapp. The Java CAS client can be downloaded from <http://www.jasig.org/products/cas/downloads/index.html>

In brief: deploy the CAS client as an extension to Shibboleth in its extension build directory, adjust the web.xml and start the Shibboleth installer (which is another ant build script).

2.3.1 Deploy Java CAS client as Shibboleth IdP extension

1. Create a new directory in the custom dir of of the Shibboleth install package, eg: **`$SHIB-IDP_SOURCE/custom/cas-client-java-2.1.1`**

- Copy everything from the CAS client source to the new directory (including subdirectories):

```
$CASCLIENT_SOURCE/* => $SHIB-IDP_SOURCE/custom/cas-client-java-2.1.1/
```

- Create a new file `$SHIB-IDP_SOURCE/custom/cas-client-java-2.1.1/build.properties` to include the CAS client in the Shibboleth build process, and that file should contain:

```
ext.name=casclient
gen.ext.docs=false
```

- Edit the file `$SHIB-IDP_SOURCE/webAppConfig/dist.idp.xml` so it uses the CAS client filter for the SSO servlet. This file needs to contain your hostnames as configuration parameters. In the example below you should only change `$CASHOSTNAME` and `$IDPHOSTNAME`. (`$IDP_HOME` is replaced by the Shibboleth build process) In most setups the hostname for the CAS server should be the same as the hostname of the Shibboleth IdP. In our case, both `$CASHOSTNAME` and `$IDPHOSTNAME` should be `xenophobe.ahds.ac.uk`. (`dist.idp.xml` will become the `web.xml` of the Shibboleth IdP webapp)

```
<web-app>
  <context-param>
    <param-name>IdPConfigFile</param-name>
    <param-value>$IDP_HOME$/etc/idp.xml</param-value>
  </context-param>

  <filter>
    <filter-name>CASFilter</filter-name>
    <filter-class>edu.yale.its.tp.cas.client.filter.CASFilter</filter-class>
    <!-- URL of login page of CAS Server -->
    <init-param>
      <param-name>edu.yale.its.tp.cas.client.filter.loginUrl</param-name>
      <param-value>https://idp.ahds.ac.uk/cas/login</param-value>
    </init-param>
    <!-- URL to validation URL of CAS Server -->
    <init-param>
      <param-name>edu.yale.its.tp.cas.client.filter.validateUrl</param-name>
      <param-value>https://idp.ahds.ac.uk/cas/proxyValidate</param-value>
    </init-param>
    <!-- Full hostname with port number to be filtered. The port
         number is not required for standard ports (80,443) -->
    <init-param>
      <param-name>edu.yale.its.tp.cas.client.filter.serverName</param-name>
      <param-value>idp.ahds.ac.uk</param-value>
```

```
</init-param>
<!-- expose REMOTE_USER (from CAS Client version 2.1.0) -->
<init-param>
  <param-name>edu.yale.its.tp.cas.client.filter.wrapRequest</param-
name>
  <param-value>>true</param-value>
</init-param>
</filter>

<filter-mapping>
  <filter-name>CASFilter</filter-name>
  <url-pattern>/SSO/*</url-pattern>
</filter-mapping>

<servlet>
  <servlet-name>IdP</servlet-name>
  <display-name>Shibboleth Identity Provider</display-name>
  <servlet-
class>edu.internet2.middleware.shibboleth.idp.IdPResponder</servlet-class>
  </servlet>

<servlet-mapping>
  <servlet-name>IdP</servlet-name>
  <url-pattern>/SSO</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name>IdP</servlet-name>
  <url-pattern>/AA</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name>IdP</servlet-name>
  <url-pattern>/Artifact</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name>IdP</servlet-name>
  <url-pattern>/Status</url-pattern>
</servlet-mapping>

<mime-mapping>
  <extension>css</extension>
  <mime-type>text/css</mime-type>
</mime-mapping>
```

5. Edit the file **`$$SHIB-IDP_SOURCE/webAppConfig/dist.idp.xml`** so that it uses the CAS client filter for the SSO servlet. This file needs to contain your hostnames as configuration parameters. In the example below you should only change **`$CASHOSTNAME`** and **`$IDPHOSTNAME`** (**`$IDP_HOME`** is replaced by the Shibboleth build process). In most setups the hostname for the CAS server should be the same as the hostname of the Shibboleth IdP. In our case, both **`$CASHOSTNAME`** and **`$IDPHOSTNAME`** should be `idp.ahds.ac.uk`. (`dist.idp.xml` will become the `web.xml` of the Shibboleth IdP webapp).

2.4 Customizing Existing Views

CAS provides two sample views, a default and simple view. The default skin is a more complex interface, utilizing CSS. The simple skin is the minimum view needed for CAS to work. The default view is located in `/var/lib/tomcat5.5/webapps/cas/WEB-INF/view/jsp`. There are four views that need to be implemented:

- `casLoginView.jsp` - The screen a user sees when they provide their credentials or if there is an error processing their credentials. A user will be redirected to this view from the WAYF.
- `casLogoutView.jsp` - The screen users see after they have ended a CAS Single Sign On Session.
- `casGenericSuccess.jsp` - The screen users will see when they successfully authenticate without coming from a service.
- `casConfirmView.jsp` - The Confirmation screen a user will see when they have chosen "warn".

2.4.1 Customizing the Views for the AHDS

The simplest way to customize an AHDS view is to make a copy of default ui and place it in `/var/lib/tomcat5.5/webapps/cas/WEB-INF/view/jsp`. For example, place the new AHDS views in `/var/lib/tomcat5.5/webapps/cas/WEB-INF/view/jsp/ahds`. The JSP pages shown above should then be edited accordingly.

Once these JSP pages have been edited, the following should be done to let CAS know of the existence of this new AHDS view:

1. Make a copy of `/var/lib/tomcat5.5/webapps/cas/WEB-INF/classes/default.properties` and update this file to reflect the new AHDS view at `/var/lib/tomcat5.5/webapps/cas/WEB-INF/classes/ahds_views.properties`.
2. Update `/var/lib/tomcat5.5/webapps/cas/WEB-INF/cas-servlet.xml`'s `ViewResolver` element to reflect the new `ahds_views.properties` file.

2.5 Shibboleth IdP 1.3

The Shibboleth IdP is software written as Servlets and JSP and more information can be found at: <http://shibboleth.internet2.edu/downloads/>.

1. Download Shibboleth Identity Provider from <http://shibboleth.internet2.edu/downloads/> and extract the package in directory `/opt`.
2. Run `$SHIB-IDP_SOURCE/ant` to deploy the IdP webapp to Tomcat (make sure `%JAVA_HOME%` is pointing to the home of your SDK).

```
Do you want to install the Shibboleth Identity Provider? [Y,n] Y
What name do you want to use for the Identity Provider web application?
[default: shibboleth-idp] shibboleth-idp

Deploying the java web application. Do you want to install it directly onto
the filesystem or use the tomcat manager application?
1) filesystem (default)
2) manager

1

Select a home directory for the Shibboleth Identity Provider [default:
/usr/local/shibboleth-idp] /opt/shibboleth-idp
Enter the tomcat home directory [default: /usr/local/tomcat]
/opt/tomcat
```

3. You can now try out the CAS client servlet filter by using this URL: <http://localhost/shibboleth-idp/SSO>. It should redirect you to the CAS login page. After logging in, you should see an error saying "no target URL received", which would be correct at this point. If not, check the Shibboleth logs (-our log4j config- `$CATALINA_HOME/logs/shibboleth-idp.log` or - the default- `$IDP_HOME/logs/shib-error.log`) and the Tomcat logs for deployment errors (`$CATALINA_HOME/logs/tomcat.log`).

2.6 LDAP Configuration

This section explains how to configure Tomcat to use an LDAP for Authentication. This is **optional** but once the IdP is fully functional, the same LDAP will be used to retrieve user attributes.

1. Comment out the existing Realm element in the server.xml (`$TOMCAT_ROOT/conf/server.xml`).
2. Insert a new Realm as the example given below:

```
<Realm className="org.apache.catalina.realm.JNDIRealm"
        debug="99"
        connectionURL="ldap://reto:389"
```

```
connectionName="cn=Manager,dc=ahds,dc=ac,dc=uk"  
connectionPassword="password"  
userBase="ou=People,dc=ahds,dc=ac,dc=uk"  
userSearch="(uid={0})"  
roleBase="ou=Groups,dc=ahds,dc=ac,dc=uk"  
roleSearch="(uniqueMember={0})"  
roleName="cn"/>
```

3 PKI Configuration

This section explains how to get a server certificate and how to configure Tomcat to use keystore and truststore. The certificate has the common name (CN) "idp.ahds.ac.uk". This section primarily describes how to get [GlobalSign](#) ServerSign certificates. A more detailed explanation can be found [here](#).

3.1 Generate certificate

1. Generate a 1024-bit RSA key in PEM format for Apache/Tomcat:

```
$ openssl genrsa -out idp.ahds.ac.uk.key 1024
```

2. Once you have a key pair, you need to send the public key to GlobalSign, along with the DNS name of the server machine to be certified and the name of your institution. You must also prove to GlobalSign that you are a legitimate user of those names. The names and public key are sent as a Certificate Signing Request (CSR) file, which can be generated by openssl:

Create a Certificate Signing Request (CSR).

```
$ openssl req -new -key idp.ahds.ac.uk.key > idp.ahds.ac.uk.csr
```

3. Once you have made the Certificate Signing Request file, it must be submitted to GlobalSign. The procedure to follow is given at <http://www.ja.net/cert/web/globalsign.html>. Once your application has been processed, GlobalSign will return a server certificate, **idp.ahds.ac.uk.pem**.
4. Get your server certificate signed with the full chain up to the root CA certificate. GlobalSign's root certificate can be found at http://support.globalsign.net/en/serversign/root_install_ap.cfm.
5. Check that your certificate contains the full chain up to the root CA certificate – the file must contain at least two **-----BEGIN CERTIFICATE-----**, **-----END CERTIFICATE-----** blocks. This can be done by using [KeyMan](#) or manually concatenating the Private Key and Certificates in the same file. The order of certificates in the file is critical: the domain (Server) Certificate, Intermediate Certificate Authority, and finally Root Certificate Authority. Save the file as **idp-bundle.crt**:

```
-----BEGIN RSA PRIVATE KEY-----  
BASE64-ENCODEDCHARACTERS  
-----END RSA PRIVATE KEY-----
```

```
-----BEGIN CERTIFICATE-----  
YOUR LOCAL DOMAIN (SERVER) CERTIFICATE  
-----END CERTIFICATE-----
```

```
----BEGIN CERTIFICATE-----  
THE INTERMEDIATE CA CERTIFICATE  
-----END CERTIFICATE-----  
  
-----BEGIN CERTIFICATE-----  
THE ROOT CA CERTIFICATE  
-----END CERTIFICATE-----
```

3.2 Updating Server Certificates for Shibboleth

The server certificates are primarily used for 2 components:

- Shibboleth IdP
- The Single Sign On system, which in our case is CAS

We will be using **keytool** to create a keystore in the JKS format. This program and format was chosen because keytool is included with the JDK. Note that the APR connectors for Tomcat DO NOT support the JKS format (the private key can be extracted from the JKS if needed later). It is also possible to use openssl in combination with the PKCS12 format (which is the most supported format, but not by keytool). The steps detailed below are for joining the SDSS test federation.

1. Check if **keytool** is in your path, the program is located in **\$JAVA_HOME/bin/**.
2. Create an empty keystore. A default, simple password is given below for test purposes. A proper password should be used for production purposes.

```
$ keytool -v -genkey -v -alias abc -keystore idp.ahds.ac.uk.jks
```

Enter keystore password: **changeme**

What is your first and last name?

[Unknown]:

What is the name of your organizational unit?

[Unknown]:

What is the name of your organization?

[Unknown]:

What is the name of your City or Locality?

[Unknown]:

What is the name of your State or Province?

[Unknown]:

What is the two-letter country code for this unit?

[Unknown]:

Is CN=Unknown, OU=Unknown, O=Unknown, L=Unknown,
ST=Unknown, C=Unknown correct?

```
[no]: yes
```

```
Generating 1,024 bit DSA key pair and self-signed certificate  
(SHA1WithDSA)
```

```
for: CN=Unknown, OU=Unknown, O=Unknown,
```

```
L=Unknown, ST=Unknown, C=Unknown
```

```
Enter key password for <abc>
```

```
(RETURN if same as keystore password):
```

```
[Storing idp.ahds.ac.uk.jks]
```

```
$ keytool -v -delete -alias abc -keystore idp.ahds.ac.uk.jks
```

3. Convert PEM formatted key to PKCS8 format:

```
$ openssl pkcs8 -in idp.ahds.ac.uk.key -topk8 -nocrypt -outform DER -out  
idp.ahds.ac.uk.key.pkcs8
```

4. Import key/certificate into keystore using the extkeytool from the Shibboleth IdP installation:

```
/opt/shibboleth-idp-1.3/bin/extkeytool -importkey \  
-alias idp \  
-keyfile idp.ahds.ac.uk.key.pkcs8 -certfile idp-bundle.crt \  
-keystore idp.ahds.ac.uk.jks -storepass changeme \  
-provider org.bouncycastle.jce.provider.BouncyCastleProvider
```

5. Make sure the certificate has been imported correctly:

```
keytool -v -list -keystore idp.ahds.ac.uk.jks
```

3.3 Updating the CA Root certificates

When a Service Provider fetches the attributes from a user's Identity Provider, the CA root certificates enable the Attribute Authority (AA) to perform client authentication, which is required to do proper ARP processing. In order to trust the certificates offered by other parties (Service providers & LDAPS-server), the Identity Provider must trust the certificate authority (CA) that signed those certificates. The CAS-client must also trust the certificate offered by the CAS-server (in most cases it's the same server). The CAS-client connects to the CAS-server when it validates the Service Ticket it has just received together with a redirect from the login. At validation time, it checks the identity of the CAS-server using the CN in the certificate (that it trusts to be correct). You can do so by importing the right CA certificates in the Java trust keystore. The default password for the Java keystore is “**changeit**”. The default Java keystore does not contain GlobalSign CA cert (required to trust the SDSS CA certification chain) and can

be downloaded [here](#). These commands create the trust for a CA by importing the CA certificates into the *default Java keystore, cacerts*.

In Tomcat, there are atleast 2 options to use the CA root certificates:

- a. The **cacerts** file that comes with your Java installation, a Java keystore in which the most common CA root certificates can be found.
- b. Use a special Java keystore which can be set as a JVM option
-Djavax.net.ssl.truststore=/path/to/truststore.jks

In both cases, new cacerts can be added with Java's keytool utility as shown below:

1. Create an empty keystore, truststore.jks:

```
$ keytool -genkey -alias foo -keystore truststore.jks
$ keytool -delete -alias foo -keystore truststore.jks
```

2. Import CA root certificates into keystore (truststore). The CA root certificates for GlobalSign, Thawte, and SDSS are imported into the truststore.jks. The CA roots for GlobalSign, Thawte, and SDSS can be found at http://support.globalsign.net/en/serversign/secureroots_new.crt, <http://www.thawte.com/roots/>, and <http://www.sdss.ac.uk/ca/sdss-ca.crt> respectively.

```
$ wget http://support.globalsign.net/en/serversign/secureroots_new.crt
$ wget http://www.thawte.com/roots/
$ wget http://www.sdss.ac.uk/ca/sdss-ca.crt

$ keytool -import -v -trustcacerts -alias globalsignca \
  -file ./globalsign.crt -keystore ./truststore.jks

$ keytool -import -v -trustcacerts -alias thawteca \
  -file ./thawte.crt -keystore ./truststore.jks

$ keytool -import -v -trustcacerts -alias sdssca \
  -file ./thawte.crt -keystore ./truststore.jks
```

3. In order to trust the certificates offered by other parties (Service providers & LDAPS-server), the Identity Provider must trust the certificate authority (CA) that signed those certificates. The CAS-client must also trust the certificate offered by the CAS-server (in most cases it's the same server). The CAS-client connects to the CAS-server when it validates the Service Ticket it has just received together with a redirect from the login. At validation time, it checks the

identity of the CAS-server using the CN in the certificate (that it trusts to be correct). You can do so by importing the right CA certificates in the Java trust keystore. The default password for the Java keystore is “changeit”. The GlobalSign CA and SDSS CA have to be imported into the Java trust keystore. The default Java keystore does not contain GlobalSign CA cert (required to trust the SDSS CA certification chain) and can be downloaded [here](#). The GlobalSign CA certs should be copied into the default Java keystore, truststore.jks and xenophobe.jks. The command below should be run for EACH certificate:

4. In **/opt/shib-keystore** (or wherever the SDSS CA and GlobalSign CA certificates have been saved):

```
keytool -import -trustcacerts -keystore $JAVA_HOME/jre/lib/security/cacerts -file  
sdss-ca.crt -alias sdssca
```

```
keytool -import -trustcacerts -keystore $JAVA_HOME/jre/lib/security/cacerts -file  
globalsignca.cer -alias globalsignca
```

3.4 Tomcat configuration

Configure the HTTP connectors for Tomcat in \$CATALINA_HOME/conf/server.xml:

- one that supports SSL without client authentication (used for encrypted connection to SSO servlet and CAS server; browser users)
- one that supports SSL with client authentication (used for attribute queries on Artifact and AA servlet; machines)

Disable httpd (Apache) from responding to port 443 and 8443 by editing **/etc/httpd/conf.d/ssl.conf**: (THIS IS OPTIONAL)

```
# Listen 443  
# Listen 8443
```

Enable Tomcat to listen to these ports in \$CATALINA_HOME/conf/server.xml:

```
<!-- Define a SSL HTTP/1.1 Connector on port 443  
without client auth -->  
<Connector port="443" maxHttpHeaderSize="8192"  
maxThreads="150" minSpareThreads="25" maxSpareThreads="75"  
enableLookups="false" disableUploadTimeout="true"
```

```
acceptCount="100" scheme="https" secure="true"  
clientAuth="false" sslProtocol="TLS"  
keystoreType="jks"  
keystoreFile="/etc/pki/idp.ahds.ac.uk.jks"  
keystorePass="changeme"  
</>  
  
<!-- Define a SSL HTTP/1.1 Connector on port 8443  
with client authentication -->  
<Connector port="8443" maxHttpHeaderSize="8192"  
maxThreads="150" minSpareThreads="25" maxSpareThreads="75"  
enableLookups="false" disableUploadTimeout="true"  
acceptCount="100" scheme="https" secure="true"  
clientAuth="want" sslProtocol="TLS"  
keystoreType="jks"  
keystoreFile="/etc/pki/idp.ahds.ac.uk.jks"  
keystorePass="changeme"  
truststoreFile="/etc/pki/truststore.jks"  
truststorePassword="changeme" />  
</>
```

Other connectors (like the AJP 1.3 connector on port 8009) are not needed for a Tomcat-only installation and can be commented out.

It should be noted that `truststorePassword` defaults to `keystorePass` even when the values are totally different. The only solution as of yet is to have the same password for both `truststorePassword` and `keystorePass`. This seems to be a bug with Tomcat and has yet to be fixed.

Restart Tomcat 5.5 to load the new configuration.

```
$ /etc/init.d/tomcat restart
```

To check if your server is correctly configured, you can use **OpenSSL's s_client** command:

```
openssl s_client -connect idp.ahds.ac.uk:443 -showcerts  
  
openssl s_client -connect idp.ahds.ac.uk:8443 -showcerts
```

4 Configuring IdP to join the SDSS Federation

4.1 Application letter to join SDSS

Once the IdP was installed and a X.509 certificate was obtained, an application letter was sent to the SDSS Federation with the following details:

Provider type: Identity Provider.

Policy: The Arts and Humanities Data Service (AHDS) team will:

Observe best practice in the handling and use of your digital certificates and private keys. Make reasonable attempts to ensure that only members of the Arts and Humanities Data Service (AHDS) are provided with credentials permitting authentication to our handle server, and that the assertions made to service providers by our attribute authority are correct.

Alias: The Arts and Humanities Data Service (AHDS), King's College London.

Technical contact: Sanjay Vivek (sanjay.vivekanandan_at_ahds.ac.uk)

Admin contact: Mark Hedges (mark.hedges_at_ahds.ac.uk)

Provider Name: urn:mace:ac.uk:sdss.ac.uk:provider:identity:ahds.ac.uk

HS Location: <https://idp.ahds.ac.uk/shibboleth-idp/SSO>

HS Name: idp.ahds.ac.uk

AA Location: <https://idp.ahds.ac.uk:8443/shibboleth-idp/AA>

AA Name: idp.ahds.ac.uk

Domain: idp.ahds.ac.uk

4.2 Uploading SDSS metadata file

Download the Java keystore sdss.jks from <http://www.sdss.ac.uk/fed/sdss.jks> into **/opt/shib-idp/etc**. This keystore contains the Federation's signing certificate.

Download the Federation metadata from <http://www.sdss.ac.uk/fed/sdss-metadata.xml> into **/opt/shib-idp/etc**.

Metadatatool should be run as a regular scheduled job, perhaps daily, to keep your site's locally cached copy of the Federation metadata (sdss-metadata.xml) up to date.

4.3 Editing idp.xml

Edit idp.xml as follows. Modify the AAUrl, defaultRelyingParty and providerId attributes of <IdPConfig> to:

```
<IdPConfig
-----
AAUrl="https://idp.ahds.ac.uk:8443/shibboleth-idp/AA"
resolverConfig="file:/opt/shib-idp/etc/resolver ldap.xml"
defaultRelyingParty="urn:mace:ac.uk:sdss.ac.uk:federation:sdss"
providerId="https://idp.ahds.ac.uk/shibboleth">
```

The defaultRelyingParty is the URN of the SDSS Federation, which is fixed. AAUrl and providerId **must** match the 'AA Location' and 'Entity ID' values respectively which you gave when joining the SDSS Federation, and which are now embedded in your <EntityDescriptor> entry in the sdss-metadata.xml file (see <http://www.sdss.ac.uk/fed/sdss-metadata.xml>). Note in particular that the above suggested value for providerId might not match the value you gave when you joined the Federation: check the value in the sdss-metadata.xml file, which is definitive.

Change the <MetadataProvider> URI to refer to sdss-metadata.xml:

```
<MetadataProvider
type="edu.internet2.middleware.shibboleth.metadata.provider.XMLMetadata"
uri="file:/opt/shib-idp/etc/sdss-metadata.xml"/>
```

Within <IdPConfig>, modify the clause identifying the certificate to use when signing identity assertions for this Federation.

Note on schemaHack="true": Early versions of the Shibboleth SP rely on an XML parser that has significant bugs. Prior to Shibboleth 1.3, all versions of the IdP accommodated these bugs directly. From release 1.3 onwards, however, deployers must specifically add the attribute schemaHack="true" if they expect to be issuing attributes to older SP versions. It is recommended that this attribute be added, as specified below, to enable interoperability with older SPs.

```
<RelyingParty name="urn:mace:ac.uk:sdss.ac.uk:federation:sdss"
signingCredential="sdssCred" schemaHack="true">
<NameID nameMapping="shm"/>
</RelyingParty>
```

In the <Credentials> section, add a <FileResolver> clause defining the key and certificate files to use for the "sdssCred" signing credential:

```
<FileResolver Id="sdssCred">
  <Key>
    <Path>file:/opt/shib-keystore/idp.ahds.ac.uk.key</Path>
  </Key>
  <Certificate>
    <Path>file:/opt/shib-keystore/idp-bundle.crt</Path>
  </Certificate>
</FileResolver>
```

4.4 Editing resolver.ldap.xml

Because SDSS Service Provider sites rely on the eduPersonScopedAffiliation attribute, you must edit the resolver.ldap.xml file for your IdP (in `/opt/shib-idp/etc`). Uncomment the <SimpleAttributeDefinition> for eduPersonScopedAffiliation and change smartScope="idp.ahds.ac.uk" to one of the domains you specified when applying to join the SDSS Federation:

```
<SimpleAttributeDefinition id="urn:mace:dir:attribute-def:eduPersonScopedAffiliation"
smartScope="idp.ahds.ac.uk">
  <AttributeDependency
    requires="urn:mace:dir:attributedef:eduPersonAffiliation"/>
</SimpleAttributeDefinition>

<SimpleAttributeDefinition id="urn:mace:dir:attribute-def:eduPersonPrincipalName"
smartScope="idp.ahds.ac.uk">
  <DataConnectorDependency requires="directory"/>
</SimpleAttributeDefinition>
```

The file resolver.ldap.xml should also be edited to point at the right ldap directory as shown below:

```
<JNDIDirectoryDataConnector id="directory">
  <Search filter="uid=%PRINCIPAL%">
    <Controls searchScope="SUBTREE_SCOPE"
      returningObjects="false" />
  </Search>
  <Property name="java.naming.factory.initial"
    value="com.sun.jndi.ldap.LdapCtxFactory" />
  <Property name="java.naming.provider.url"
    value="ldap://xenophobe.ahds.ac.uk:389/ou=People,dc=ahds,dc=ac,dc=uk" />
```

```
<Property name="java.naming.security.principal"  
value="cn=Manager,dc=ahds,dc=ac,dc=uk" />  
<Property name="java.naming.security.credentials"  
value="secret" />  
</JNDIDirectoryDataConnector>
```

5 Testing

You can test your IdP setup by accessing the SDSS test page shown below:

<https://target.sdss.ac.uk/secure/index.html>

Select AHDS, King's College from the WAYF and authenticate yourself. You should then see "**Secured Page, You made it!**".

Two other test pages list target CGI environment variables, thus showing eduPersonPrincipalName and eduPersonScopedAffiliation attribute values of the authenticated user:

<http://nevis.ed.ac.uk:8885/cgi-bin/printenv>
<http://target.iay.org.uk/secure/printenv.cgi>