Shintau Architecture Installation Guide

Version	Date	Comments
0.1	06th April 2009	First draft by George Inman
0.2	17th April 2009	Comments by Stijn after trying it out.
0.3	1st May 2009	Updated by George after Adding LS and SP Installation Instructions
0.4	1st May 2009	Updated by George after creating beta 0.1 release packages

Contents

1. Introduction

This document describes the installation and configuration steps necessary to install and configure an implementation of the Shintau Identity Provider (IdP), Linking Service (LS) and ServiceProvider configured to work with the Shintau attribute aggregation framework. The conceptual model and protocol description used by the software described below are available from the Shintau project website (http://sec.cs.kent.ac.uk/shintau). We recommend that you proceed through this document in the way that it is written. e.g. Install the IdP first, then the LS and finally the SP.

As the software is currently in beta format please email the PERMIS users mailing list (permis-users@jiscmail.ac.uk) with any bugs or problems you may have installing the software.

Please note that this document has been written under the assumption that the reader is installing this software in a Linux based development environment and hence not all the documentation will be relevant to a Windows based installation.

2. Identity Provider Installation

Within the Shintau environment an Identity Provider entity consists of two entities, a modified Shibboleth 2.1.3 Identity Provider (http://shibboleth.internet2.edu/) and a modified IDWSF Liberty Alliance Discovery service based upon the IDWSF Server Toolkit (http://www.cahillfamily.com/OpenSource/) running under Axis 1.3. Both of these entities use a PostgreSQL database to store service information and both run inside a single Apache Tomcat server with an Apache HTTPD server frontend.

2.1 Prerequisites for the Identity Provider Installation

When installing the Identity Provider we assume that the following software has been installed:

1. The Sun java development kit, either release 1.5 or 6 (Please note that non Sun implementations of the Java JDK are not compatible with this software).
2. The Apache Ant java installation and testing suite.
3. A recent version of OpenSSL .

We also presume that the JAVA_HOME and ANT_HOME environment variables have been set to the root directory of your compatible JDK and ant installations respectively.

2.2 Installing Apache Tomcat

In order to install tomcat please download the latest binary distribution of Tomcat 5.5 from the Apache Site (http://tomcat.apache.org/).

1. Once the download is complete extract the archive file and move the entire folder it contains to `/usr/local`.
   e.g.

```
$ tar -xzf apache-tomcat-5.5.27.tar.gz
$ sudo mv apache-tomcat-5.5.27 /usr/local/
```

2. It may also be useful to create a symlink for /usr/local/tomcat that points towards the current version of tomcat. For the remainder of this document we will refer to this directory as $CATALINA_HOME
   e.g.

```
$ cd /usr/local/
$ sudo ln -s apache-tomcat-5.5.27/ tomcat
```

4. Set the CATALINA_HOME and JAVA_HOME environment variables in /usr/local/tomcat/bin/catalina.sh

```
export CATALINA_HOME=/usr/local/tomcat
export JAVA_HOME=/usr/lib/jvm/java-6-sun
```

5. Enable the tomcat Manager application by adding the following lines to $CATALINA_HOME/conf/tomcat-users.xml inside the <tomcat-users> element.
   e.g.

```
<role rolename ="manager"/>
<user username="manager" password="password" roles="manager"/>
```

`The username and password may be freely chosen.`

6. Once this has been accomplished the tomcat server should be installed and can be started and stopped using the startup.sh and shutdown.sh scripts contained in the installation's bin directory.

   e.g.

```
$ sudo /usr/local/tomcat/bin/startup.sh
$ sudo /usr/local/tomcat/bin/shutdown.sh
```

The installation can then be verified by viewing the tomcat welcome page available at http://localhost:8080/ whilst the server is running.

2.3 Installing Axis 1.3

Axis is a Java web services framework produced by the Apache foundation. The axis 1.3 release package is no longer available from the Axis website but can be downloaded from (http://www.cahillfamily.com/OpenSource/).

1. Once the download is complete extract the archive file and move the newly created folder to `/usr/local/axis`. For the remainder of this document we will refer to this directory as $AXIS_HOME.

```
$ tar -xzf axis-bin-1_3.tar.gz
$ sudo mv axis-1_3 /usr/local/axis
```

2. Set the AXIS_HOME variable to the root directory of your axis installation.

```
$ export AXIS_HOME=/usr/local/axis
```

3. Copy the webapps/axis directory from the xml-axis distribution to tomcat's webapps directory. It is probably safer to do this whilst the tomcat server isn't running.

```
$ sudo cp –r $AXIS_HOME/webapps/axis $CATALINA_HOME/webapps
```

4. Set the AXIS_CLASSPATH variable in $CATALINA_HOME/bin/catalina.sh so that the axis installation can find each of the JAR files that it requires to install the service. These JAR files are located inside the $AXIS_HOME/lib directory.

```
export AXIS_CLASSPATH=/usr/local/axis/lib/axis-ant.jar:/usr/local/axis/lib/axis-
schema.jar:/usr/local/axis/lib/commons-logging-1.0.4.jar:/usr/local/axis/lib/log4j-
1.2.8.jar:/usr/local/axis/lib/saaj.jar:/usr/local/axis/lib/axis.jar:/usr/local/axis/l
ib/commons-discovery-
0.2.jar:/usr/local/axis/lib/jaxrpc.jar:/usr/local/axis/lib/log4j.properties:
/usr/local/axis/lib/wsdl4j-1.5.1.jar:
```

5. Enable the Axis admin servlet, by default the axis admin application for web based application deployment is disabled. It can be re-enabled by un-commenting the relevant section in the web.xml configuration file in the $CATALINA_HOME/webapps/axis/WEB-INF/ directory.

   e.g

```
<!-- uncomment this if you want the admin servlet -->
 <!--
  <servlet-mapping>
    <servlet-name>AdminServlet</servlet-name>
    <url-pattern>/servlet/AdminServlet</url-pattern>
  </servlet-mapping>
 -->
```

   Becomes:

```
<!-- uncomment this if you want the admin servlet -->
  <servlet-mapping>
    <servlet-name>AdminServlet</servlet-name>
    <url-pattern>/servlet/AdminServlet</url-pattern>
  </servlet-mapping>
```

The Axis installation can now be validated after starting the tomcat server using the happy axis jsp application which can be viewed at http://localhost:8080/axis/happyaxis.jsp. Don't worry if the Optional Components section of the displayed page expresses warnings as the functionality provided by those components are not necessary to run this service.

2.4 Installing PostgreSQL

PostgreSQL is an open source SQL database which can be obtained from http://www.postgresql.org/. This document describes the process of installing the database from source. Please download the latest version of the server from the project website.

1. Create a postgres user and set the user's password

```
$ sudo useradd postgres
$ sudo passwd postgres
```

2. Once the download is complete please extract the archive file.

```
$ tar -xzf postgresql-8.3.7.tar.gz
```

3. Navigate into the newly created postgresql directory and run the ./configure script specifying the installation directory using the prefix variable(for the purposes of this document /usr/local/postgresql). If ./configure completes successfully then build and deploy the server using GNU make.

```
$ cd postgresql-8.3.7
$ ./configure --prefix=/usr/local/postgresql
$ make
$ sudo make install
```

4. Change the owner of the postgres installation to the postgres user. From now the installation directory will be referred to as $POSTGRES_HOME

```
$ sudo chown –R postgres.postgres /usr/local/postgresql
```

5. In order to initialise the database please run the following command

```
$ sudo  -u postgres /usr/local/postgresql/bin/initdb  –D /usr/local/postgresql/data
```

6. The database should have now been installed, in order to start and stop the server the following commands should be used.

   To start the server :

```
$ sudo –u postgres /usr/local/postgresql/bin/pg_ctl –D /usr/local/postgresql/data
start
```

   To stop the server:

```
$ sudo –u postgres /usr/local/postgresql/bin/pg_ctl –D /usr/local/postgresql/data stop
```

7. In order to configure the database for use with Shintau we must also configure the server to use passwords rather than allowing all users to access the server. First we must give the root postgres user a password in the database.

```
$ sudo –u postgres /usr/local/postgresql/bin/psql
Postgres=# ALTER USER postgres WITH PASSWORD ‘new_password’;
Postgres=# \q
```

   Where the string ‘new_password’ is the new password for the postgres user.

8. The file $POSTGRES_HOME/data/pg_hba.conf file contains the postgresql server’s trust configuration. As the postgres user, comment out the line :

```
local all all trust
```

   and replace it with:

```
local all all md5
```

to enable md5 passwords.

9. Each user that is to be used as part of the installation should now be initialised as a postgres user using the createuser command. In the example below we create two users. We have the issrg user who is our main user account and root which is the root user account. Follow the on screen commands and make each user account a super user account for the purposes of this installation.

```
$ sudo -u postgres /usr/local/postgresql/bin/createuser -d -R -P issrg
$ sudo –u postgres /usr/local/postgresql/bin/createuser -d -R -P root
```

The PostgreSQL server should now be ready to be configured for use with the Shintau databases at a later stage in the installation.

2.5 Installing Apache 2.2 with SSL support

The Apache HTTPD web server is available from the Apache foundation web site (http://httpd.apache.org/). This documentation describes the installation and configuration of an apache 2.2.* server with SSL and proxy support for tomcat using mod_proxy_ajp. Please note that the mod_proxy_ajp module is only available in apache 2.1 and later for earlier versions of apache mod_jk should be utilised.

1. Once the download is complete extract the archive file and navigate into the directory e.g.

```
$ tar -xzf httpd-2.2.11.tar.gz
$ cd httpd-2.2.11/
```

2. Run the ./configure server configuration script with the following parameters to configure the build script before using GNU make to install and deploy the server using the make and make install commands.

   --prefix : This parameter determines the directory where the server will be installed
   -- enable-ssl : This parameter enables the apache SSL module for compilation
   --enable-dso: This parameter enables Dynamic Shared Objects and allows additional modules to added after the serve r is installed
   --enable-proxy : This parameter enables the apache HTTP proxying module
   --enable-proxy-ajp: This parameter enables an AJP proxying module for use with tomcat
   --enable-mods-shared=all : This command builds all the remaining shared modules (Optional)
   e.g.

```
$ ./configure --prefix=/usr/local/apache --enable-ssl --enable-dso --enable-proxy --
enable-proxy-ajp --enable-mods-shared=all
$ make
$ sudo make install
```

   From this point on we will refer to the installation directory as $APACHE_HOME

3. The apache server is now installed and can be started and stopped using the following commands

```
$ sudo /usr/local/apache/bin/apachectl –k start
```

```
$ sudo /usr/local/apache/bin/apachectl –k stop
```

You can then validate that the server is running by viewing the page http://localhost/ where you should see an "It Works" page

4. In order to configure SSL we must first generate a private key and self-signed certificate for the server that can be used to identify the server. This will be done using OpenSSL. This step can be skipped if you already have a server certificate.

   $ openssl genrsa 1024 > server.key
   $ openssl req –new –x509 –nodes –sha1 –days 365 –key server.key > server.crt

   The first command creates a 1024 bit RSA key called server key and the second creates a new certificate called server.crt valid for a year and signed by the key created in the first command. The second command requires that certain information be filled in about the server. The most important of these is the Common Name field which should contain the hostname or IP address of the server to be used. The example below shows a configuration for a server with the hostname of issrg-testidp.kent.ac.uk.

   e.g.

```
$ openssl req -new -x509 -nodes -sha1 -days 365 -key server.key > server.cert

You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:UK
State or Province Name (full name) [Some-State]:Kent
Locality Name (eg, city) []:Canterbury
Organization Name (eg, company) [Internet Widgits Pty Ltd]:University of Kent
Organizational Unit Name (eg, section) []:ISSRG
Common Name (eg, YOUR name) []:issrg-testidp.kent.ac.uk
Email Address []:
```

   You now have created a certificate/key pair which can be used to identify your web host.

5. We will now configure Apache for SSL. The SSL module should have been enabled by Apache at installation so we will start by enabling the ssl configuration file which provides a new virtual host that listens on port 443 in the apache configuration file httpd.conf located in the conf directory of the apache installation location.

   In this file remove the comments around the line:

```
# Include conf/extra/httpd-ssl.conf
```

   So that it becomes:

```
Include conf/extra/httpd-ssl.conf
```

You should then change the SSLCertificateFile and SSLCertificateKeyFIle parameters in the newly enabled configuration (httpd-ssl.conf) so that they point to your newly created certificate and key.

e.g.

```
SSLCertificateFile "/usr/local/apache/conf/server.crt"
SSLCertificateKeyFile "/usr/local/apache/conf/server-dsa.key"
```

Becomes:

```
SSLCertificateFile "/path/to/new/certificate/server.crt"
SSLCertificateKeyFile "/path/to/new/key/server.key"
```

Please make sure that apache has sufficient privileges to read both the certificate and key files before restarting apache using the –D SSL switch which enables SSL support in the server.
e.g.

```
$ sudo /usr/local/apache/bin/apachectl –k start –D SSL
```

If SSL has been successfully configured then you should be able to view the https://localhost/ page. Do not worry if the browser throws a certificate error, this is displayed because we used a self signed certificate rather than one signed by an established CA and the browser does not trust the signer. Note that your users will have exactly the same experience on using the IdP.for the first time.

6. The final step is to configure apache to proxy requests for certain directory pages to the tomcat server setup in step 2.2. In order to do this we must ensure that mod_proxy_ajp is enabled which can be confirmed by viewing the apache configuration file $APACHE_HOME/conf/httpd.conf and searching for the Line "LoadModule proxy_ajp_module modules/mod_proxy_ajp.so". If it is present then we can proceed otherwise you will need to compile the module from source using the apxs command.

   In order to demonstrate mod_proxy_ajp working with our tomcat server we are going to enable access to the jsp-examples page installed by default on the tomcat server when a user attempts to access the directory examples on the apache server. To do this we must place the following line at the bottom of the httpd.conf file.

   ProxyPass /examples ajp://127.0.0.1:8009/jsp-examples

Restart the apache web server and attempt to access https://localhost/examples/. You should then be forwarded to the tomcat examples page. Make sure that tomcat is running before trying this or you will end up with a 'Service Temporarily Unavailable' error.

2.6 Install the Shibboleth Identity Provider

This guide will describe how to install a modified shibboleth Identity provider using the testshib test federation for testing purposes.

1. Extract the Shintau identity provider and navigate into the created folder

```
$ tar xzf shintau-identityprovider-2.1.3-bin.tar.gz
$ cd shintau-identityprovider-2.1.3
```

2. Make the automated installation script ./install executable and then run it as root and follow the on screen instructions to install the base identity provider installation. We recommend that you install the identity provider in /usr/local/idp. The hostname should match that of the server.

   e.g.

```
$ chmod 755./install.sh
$ sudo JAVA_HOME=/usr/lib/jvm/java-6-sun./install.sh

Buildfile: src/installer/resources/build.xml

install:
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Be sure you have read the installation/upgrade instructions on the Shibboleth website
before proceeding.
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Where should the Shibboleth Identity Provider software be installed? [/opt/shibboleth-
idp]
/usr/local/idp
What is the fully qualified hostname of the Shibboleth Identity Provider server?
[idp.example.org]
issrg-testidp.kent.ac.uk
A keystore is about to be generated for you. Please enter a password that will be used
to protect it.
password
Updating property file: /home/issrg/Desktop/Installation files/shibboleth-
identityprovider-2.1.3/src/installer/resources/install.properties
Created dir: /usr/local/idp
Created dir: /usr/local/idp/bin
Created dir: /usr/local/idp/conf
Created dir: /usr/local/idp/credentials
Created dir: /usr/local/idp/lib
Created dir: /usr/local/idp/lib/endorsed
Created dir: /usr/local/idp/logs
Created dir: /usr/local/idp/metadata
Created dir: /usr/local/idp/war
Generating signing and encryption key, certificate, and keystore.
Copying 5 files to /usr/local/idp/bin
Copying 9 files to /usr/local/idp/conf
Copying 1 file to /usr/local/idp/metadata
Copying 48 files to /usr/local/idp/lib
Copying 5 files to /usr/local/idp/lib/endorsed
Copying 1 file to /home/issrg/Desktop/Installation files/shibboleth-identityprovider-
2.1.3/src/installer
Building war: /home/issrg/Desktop/Installation files/shibboleth-identityprovider-
2.1.3/src/installer/idp.war
Copying 1 file to /usr/local/idp/war
Deleting: /home/issrg/Desktop/Installation files/shibboleth-identityprovider-
2.1.3/src/installer/web.xml
Deleting: /home/issrg/Desktop/Installation files/shibboleth-identityprovider-
2.1.3/src/installer/idp.war

BUILD SUCCESSFUL
Total time: 43 seconds
```

3. Endorse XALAN and XERCES by copying all the `.jar` files included in the IdP source endorsed directory into Tomcat's $CATALINA_HOME/common/endorsed directory. Some, but not all, versions of Tomcat ship with some XML related JARs in this directory. If you see JAR files

related to JAXP or XML please remove these as well as adding the aforementioned Xerces and Xalan JARs.

4. Install the Shibboleth Java security provider by copying the library shibboleth-jce-1.0.0.jar, located in the IdP source's lib directory into $JAVA_HOME/jre/lib/ext and editing the file java.security located in the JRE's lib/security directory ($JAVA_HOME/jre/lib/security) by adding the following line after the last security.provider entry:
`security.provider.#=edu.internet2.middleware.shibboleth.DelegateToApplicationProvider`
where # is replaced with the a number one more than the last provider in the list. e.g.

```
... 5 previous provider declarations ...
security.provider.6=sun.security.jgss.SunProvider
security.provider.7=com.sun.security.sasl.Provider
security.provider.8=edu.internet2.middleware.shibboleth.DelegateToApplicationProvider
```

5. Configure tomcat to use SSL and shibboleth java security provider by copying the following Connector definition into Tomcat's conf/server.xml configuration file. This definition should be placed either before the first, or after the last, connector already defined in the configuration file.

```
<Connector port="8443"
           maxHttpHeaderSize="8192"
           maxSpareThreads="75"
           scheme="https"
           secure="true"
           clientAuth="want"
           SSLEnabled="true"
           sslProtocol="TLS"
           keystoreFile="IDP_HOME/credentials/idp.jks"
           keystorePass="PASSWORD"
           truststoreFile="IDP_HOME/credentials/idp.jks"
           truststorePass="PASSWORD"
           truststoreAlgorithm="DelegateToApplication"/>
```

Replace IDP_HOME with the directory you installed the IdP into (e.g /usr/local/idp in the example above) and PASSWORD with the password you entered during your installation of the IdP for the keystore password.

6. This step configures tomcat to use the WAR file generated when the identity provider was installed by giving Tomcat a small piece of XML which tells it where to get the WAR and provides some properties used when Tomcat load the application.

   Create the file TOMCAT_HOME/conf/Catalina/localhost/idp.xml and place the following content in it:

```
<Context docBase="IDP_HOME/war/idp.war"
         privileged="true"
         antiResourceLocking="false"
         antiJARLocking="false"
         unpackWAR="false"
         swallowOutput="true" />
```

   Replacing IDP_HOME with your IdP's home directory (/usr/local/idp).

7. Export the following JAVA_OPTS parameters in $CATALINA_HOME/bin/catalina.sh to prevent the server crashing due to memory leak errors

-Xmx###m - this is the maximum amount of memory that Tomcat may use, at least 512M is recommended.

-XX:MaxPermSize=###m - (Sun JVM specific option) the maximum amount of memory allowed for the permanent generation object space. Set this to half of the maximum memory (specified above) or 512M, whichever is less.

e.g.

```
export JAVA_OPTS="-Xmx1024m -XX:MaxPermSize=512m"
```

8. Export the following IDP_HOME parameter in $CATALINA_HOME/bin/catalina.sh to allow the IdP to find the Database setting for Shintau.

```
export IDP_HOME=/usr/local/idp/
```

9. Configure the IdP to use LDAP based JAAS username and password authentication. This is accomplished by editing two files. The first determines the type of the authentication and is called IDP_HOME/conf/handler.xml and the second determines the LDAP server to use for authentication and is called IDP_HOME/conf/login.config. The first step here is to set the type of authentication to UsernamePassword by commenting out the original Remote user login handler and uncommenting the UsernamePassword definition in handler.xml.

e.g.

```
<!-- Login Handlers -->
    <LoginHandler xsi:type="RemoteUser">

<AuthenticationMethod>urn:oasis:names:tc:SAML:2.0:ac:classes:unspecified</Authenticati
onMethod>
    </LoginHandler>

    <!--  Username/password login handler -->
    <!--
    <LoginHandler xsi:type="UsernamePassword"
                  jaasConfigurationLocation="file:///usr/local/idp/conf/login.config">

<AuthenticationMethod>urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTranspor
t</AuthenticationMethod>
    </LoginHandler>
    -->
```

Becomes:

```
<!-- Login Handlers -->
<!--
   <LoginHandler xsi:type="RemoteUser">

<AuthenticationMethod>urn:oasis:names:tc:SAML:2.0:ac:classes:unspecified</Authenticati
onMethod>
    </LoginHandler>
-->
    <!--  Username/password login handler -->

    <LoginHandler xsi:type="UsernamePassword"
                  jaasConfigurationLocation="file:///usr/local/idp/conf/login.config">

<AuthenticationMethod>urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTranspor
</AuthenticationMethod>
    </LoginHandler>
```

We must then configure the login handler to connect to our LDAP server. In order to do this we comment out the existing LDAP example in the login.config file and modify it to connect to our LDAP server. The example configuration below connects to the University of Kent's publicly accessible test LDAP server which will be used for all future tests in this document.

e.g.

```
/*
   edu.vt.middleware.ldap.jaas.LdapLoginModule required
      host="ldap.example.org"
      base="ou=people,dc=example,dc=org"
      ssl="true"
      userField="uid";
*/
```

Becomes:

```
edu.vt.middleware.ldap.jaas.LdapLoginModule required
host="sec.cs.kent.ac.uk"
base="o=PERMISv5, c=gb"
ssl="false"
userField="uid";
```

10. The next step is to create a proxy link between tomcat's IdP directory and Apache's SSL virtual host so that the IdP is visible to outside users. This is accomplished by placing a new proxy pass directive in Apache's SSL configuration file $APACHE_HOME/conf/extra/httpd-ssl.conf.

```
ProxyPass /idp ajp://127.0.0.1:8009/idp
```

You can now validate the IdP installation by restarting both tomcat and the apache web server before navigating to https://localhost/idp/profile/Status if the page displays "OK" then the IdP has been installed correctly.

11. We will now configure the IdP for use with the testshib 2 testing federation. If you are a member of another federation you may wish to ignore this step and configure your own federation metadata here.

We will begin by changing the metadata provider in IDP_HOME/conf/relying-party.xml configuration so that it uses the testshib 2 metadata. To do this uncomment the URLMD <MetadataProvider> element and change the metadata URL to http://www.testshib.org/metadata/testshib-providers.xml and the backing file to $IDP_HOME/metadata/testhib.xml

Next comment out the entire <MetadataFilter> , from the ChainingFilter down as the metadata for testshib is insecure and therefore not signed. In a production environment these filters should be turned on.

e.g.

```
<!--
        <!-- Example metadata provider. -->
        <!-- Reads metadata from a URL and store a backup copy on the file system. -->
        <!-- Validates the signature of the metadata and filters out all by SP
entities in order to save memory -->
        <!-- To use: fill in 'metadataURL' and 'backingFile' properties on
MetadataResource element -->

<MetadataProvider id="URLMD" xsi:type="FileBackedHTTPMetadataProvider"
xmlns="urn:mace:shibboleth:2.0:metadata"
                          metadataURL="http://example.org/metadata.xml"
```

```
                        backingFile="/usr/local/idp/metadata/some-metadata.xml">
            <MetadataFilter xsi:type="ChainingFilter"
xmlns="urn:mace:shibboleth:2.0:metadata">
                <MetadataFilter xsi:type="RequiredValidUntil"
xmlns="urn:mace:shibboleth:2.0:metadata"
                                maxValidityInterval="604800" />
                <MetadataFilter xsi:type="SignatureValidation"
xmlns="urn:mace:shibboleth:2.0:metadata"
                                trustEngineRef="shibboleth.MetadataTrustEngine"
                                requireSignedMetadata="true" />
                    <MetadataFilter xsi:type="EntityRoleWhiteList"
xmlns="urn:mace:shibboleth:2.0:metadata">
                        <RetainedRole>samlmd:SPSSODescriptor</RetainedRole>
                    </MetadataFilter>
            </MetadataFilter>
        </MetadataProvider>
        -->
```

becomes

```
<MetadataProvider id="URLMD" xsi:type="FileBackedHTTPMetadataProvider"
xmlns="urn:mace:shibboleth:2.0:metadata"
                        metadataURL="http://www.testshib.org/metadata/testshib-
providers.xml"
                        backingFile="/usr/local/idp/metadata/testshib.xml">
<!--
  <MetadataFilter xsi:type="ChainingFilter" xmlns="urn:mace:shibboleth:2.0:metadata">
                <MetadataFilter xsi:type="RequiredValidUntil"
xmlns="urn:mace:shibboleth:2.0:metadata"
                                maxValidityInterval="604800" />
                <MetadataFilter xsi:type="SignatureValidation"
xmlns="urn:mace:shibboleth:2.0:metadata"
                                trustEngineRef="shibboleth.MetadataTrustEngine"
                                requireSignedMetadata="true" />
                    <MetadataFilter xsi:type="EntityRoleWhiteList"
xmlns="urn:mace:shibboleth:2.0:metadata">
                        <RetainedRole>samlmd:SPSSODescriptor</RetainedRole>
                    </MetadataFilter>
            </MetadataFilter>
-->
        </MetadataProvider>
```

12. Register the IdP with the testshib service at (http://testshib.org).

Restart the IdP (for instance through tomcat's manager interface) and validate your installation using the testshib SP (https://sp.testshib.org). This will allow you to authenticate to the IdP using one of three test accounts present in the University of Kent's LDAP server and return an SSO cookie to the testshib SP from your IdP. At this point your IdP works exactly as a normal Shibboleth IdP would. The following account details are available for testing with:

```
Username: User0
Password: User0

Username: User1
Password: User1

Username: User2
Password: User2
```

Please note: This IdP is not yet configured to use Shintau aggregation.

2.7 Install the IDWSF Liberty Alliance Discovery Service

1. Download and extract the JUnit java testing suite from http://junit.org extract the distribution and move it to /usr/local/junit.

```
$ tar –xzf  junit4-5.tar.gz
$ mv junit4-5 /usr/local/junit
```

2. Download the modified IDWSF liberty Alliance Server toolkit from the PERMIS website, extract it to a convenient directory and navigate into the newly created directory. This directory shall henceforth be referred to as $DISCO_HOME for the purposes of this installation guide.

```
$ tar -xzf shintau-LibertyIDWSF.tar.gz

$ cd shintau-libertyIDWSF
```

3. Copy the Postgresql JDBC Java connector from the root of the extracted directory to the axis common library so that axis can use it.
   e.g.

```
$ sudo cp ./postgresql-jdbc3-8.2.jar /usr/local/tomcat/webapps/axis/WEB-INF/lib/
```

4. Navigate into the $DISCO_HOME/install directory

```
$ cd install/
```

5. Edit the program defaults described in Constants.java. We must first edit the basic program definitions defined in $DISCO_HOME/install/idwsf/src/liberty/toolkit/constants.java. These definitions determine the URLs of the discovery service when installed. We only need edit the basic definitions shown below:

```
        /*
         * Basic Defs
         */

public static /*final*/ String
                strMyHostSSL= "https://issrg-testidp.kent.ac.uk:8443/";
public static /*final*/ String
                strMyHostNonSSL= "http://issrg-testidp.kent.ac.uk:8080/";
public static /*final*/ String
        strMyPIDBase= "https://issrg-testidp.kent.ac.uk/";
public static /*final*/ String
                strAxisPath= "axis/services/";
```

   The first of these settings strMyHostSSL defines the SSL endpoint used by tomcat and the second strMyHostNonSSL defines the normal http endpoint used by tomcat. The third setting defines the hostname of the service and the final details the path required to access the axis services installed on the tomcat server.

6. Edit the Shintau specific defaults described in SetupLinking.java. These are used to determine whether the server is to use Shintau aggregation and if so how to access the database. Open the file $DISCO_HOME/install/ds2/src/issrg/linking/SetupLinking.java.

```
public static final String pGSQLuser = "issrg";
public static final String pGSQLPassword = "password";
public static final String pGSQLDatabase = "lib-idwsf";
```

```
public static final boolean isIdP = true;
public static final boolean useShintau = true;
```

The first three variables are used to define the connection to the postgresql server and the username and password should correspond to one of the user's created when setting up posrtgresql. As this is an Identity Provider installation isIdP should be true as in the example above and the useShintau should also be true to enable Shintau Aggregation rather than the default behaviour.

7. Edit the DBSettings file found in the $DISCO_HOME/install/db/Setup directory. This file determines the database settings to be used when creating the service's PostgresSQL database. You should edit the LIBUSER and LIBPASS variables so that they represent the same user defined in the ShintauSetup file. The rest of the file should not be modified. In the same directory, change the MyHostName variable in the FillDB file.
8. Change the username/password combination in the file $DISCO_HOME/install/db/src/liberty/db/DBCore.java. This is the username/password combination that will be used for accessing the discovery service database.

9. In order to create the database the /var/lib/pgsql directory needs to be created for the database files to be stored in. This directory should also be owned by the root PostgreSQL user postgres so that the postgres database can write to the directory.

```
$ sudo mkdir /var/lib/pgsql
$ sudo chown postgres.postgres /var/lib/pgsql
```

10. Creating and initialising the database with an intial set of values is the next step and this can be accomplished by running the InitDB, FillDB and SetupDB scripts located in the $DISCO_HOME/install/db/Setup directory. Please note that when using the sudo command the postgresql bin directory may need to be added to the PATH variable in order to successfully run these commands.

```
$ cd $DISCO_HOME/db/Setup
$ sudo PATH=$PATH:/usr/local/postgresql/bin ./InitDB
$ sudo PATH=$PATH:/usr/local/postgresql/bin ./FillDB
$ sudo PATH=$PATH:/usr/local/postgresql/bin ./SetupDB
```

11. Run the ant installation script from the installation directory (install/). This requires your PATH environment variable to contain links to the java bin directory, the postgresql bin directory and the tomcat bin directory. An extensive CLASSPATH environment variable must also be set containing all the AXIS library JAR files, the JUnit JAR, the PostgreSQL JDBC connector JAR and servlet.jar from the tomcat common directory. When using sudo to run the ant script these variables should prefix the ant command to be run. The ant installation parameter is called install and an example command including the required classpath is shown below. Note; it might be easier to use an interactive 'root' session so that it is easier to set the different environment variables. If you would like to be on the safe side, you can run 'ant fullclean' first (making sure that all environment variables are set).

```
$ sudo ANT_HOME=/usr/share/ant/ PATH=$PATH:/usr/bin:/usr/lib/jvm/java-6-
sun/bin:/usr/local/tomcat/bin:/usr/local/postgresql/bin/
CLASSPATH=$CLASSPATH:/usr/local/axis/lib/axis.jar:/usr/local/axis/lib/saaj.jar:/usr/lo
cal/axis/lib/commons-logging-1.0.4.jar:/usr/local/axis/lib/commons-discovery-
0.2.jar:/usr/local/axis/lib/wsdl4j-1.5.1.jar:/usr/local/axis/lib/log4j-
1.2.8.jar:/usr/local/tomcat/common/lib/servlet.jar:/usr/local/axis/
lib/jaxrpc.jar:/usr/local/tomcat/common/endorsed/xalan-
2.7.1.jar:/usr/local/tomcat/common/endorsed/serializer-
2.9.1.jar:/usr/local/tomcat/common/endorsed/xercesImpl-
2.9.1.jar:/usr/local/tomcat/common/endorsed/xml-apis-
2.9.1.jar:/usr/local/tomcat/common/lib/postgresql-jdbc3.jar:/usr/local/junit/junit-
4.5.jar ant install
```

Please note that there may be an error running the installation script due to compilation problems when compiling downloaded wsdl files to java. If this occurs then you should overwrite the install/wsdl/build directory with the schema files located in old-wsdl/build and rerun the script e.g.

```
$ sudo cp old-wsdl/build install/wsdl/
$ sudo ANT_HOME=/usr/share/ant/ PATH=$PATH:/usr/bin:/usr/lib/jvm/java-6-
sun/bin:/usr/local/tomcat/bin:/usr/local/postgresql/bin/
CLASSPATH=$CLASSPATH:/usr/local/axis/lib/axis.jar:/usr/local/axis/lib/saaj.jar:/usr/lo
cal/axis/lib/commons-logging-1.0.4.jar:/usr/local/axis/lib/commons-discovery-
0.2.jar:/usr/local/axis/lib/wsdl4j-1.5.1.jar:/usr/local/axis/lib/log4j-
1.2.8.jar:/usr/local/tomcat/common/lib/servlet.jar:/usr/local/axis/
lib/jaxrpc.jar:/usr/local/tomcat/common/endorsed/xalan-
2.7.1.jar:/usr/local/tomcat/common/endorsed/serializer-
2.9.1.jar:/usr/local/tomcat/common/endorsed/xercesImpl-
2.9.1.jar:/usr/local/tomcat/common/endorsed/xml-apis-
2.9.1.jar:/usr/local/tomcat/common/lib/postgresql-jdbc3.jar:/usr/local/junit/junit-
4.5.jar ant install
```

12. Install the service in axis and tomcat, this is accomplished by using the ant "config" command. In order to successfully complete this step the tomcat server must be running and the axis administration servlet must have been enabled.
    e.g.

```
$ sudo ANT_HOME=/usr/share/ant/ PATH=$PATH:/usr/bin:/usr/lib/jvm/java-6-
sun/bin:/usr/local/tomcat/bin:/usr/local/postgresql/bin/
CLASSPATH=$CLASSPATH:/usr/local/axis/lib/axis.jar:/usr/local/axis/lib/saaj.jar:/usr/lo
cal/axis/lib/commons-logging-1.0.4.jar:/usr/local/axis/lib/commons-discovery-
0.2.jar:/usr/local/axis/lib/wsdl4j-1.5.1.jar:/usr/local/axis/lib/log4j-
1.2.8.jar:/usr/local/tomcat/common/lib/servlet.jar:/usr/local/axis/
lib/jaxrpc.jar:/usr/local/tomcat/common/endorsed/xalan-
2.7.1.jar:/usr/local/tomcat/common/endorsed/serializer-
2.9.1.jar:/usr/local/tomcat/common/endorsed/xercesImpl-
2.9.1.jar:/usr/local/tomcat/common/endorsed/xml-apis-
2.9.1.jar:/usr/local/tomcat/common/lib/postgresql-jdbc3.jar:/usr/local/junit/junit-
4.5.jar ant config
```

    You should now restart tomcat and you should be able to validate that the discovery service has been installed by checking the list of deployed axis services (http://localhost:8080/axis/services). Note that there is no '/' at the end of this URL!

2.8 IdP configuration

At this point all the components necessary to install the IdP have been installed but in order for them to work together correctly they now need to be configured to correctly release attributes and referral information.

1. The first step is to configure the discovery service's entity Identifier so that it associates the IdPs entity identifier with its SAML 2.0 AttributeQuery endpoint. This is accomplished by running the SQL script below which will create a new service definition for the endpoint. You

should run the script while being connect to the postgres database, for instance through the graphical user interface pgAdmin III.

```
INSERT INTO providers (pr_id, pr_abstract)
VALUES ('PROVIDER-ID', 'SAML Attribute Service V2.0');

INSERT INTO svc_metadata (smd_id, smd_abstract, smd_pr_id , smd_sequence,
smd_lifetime)
VALUES ('PROVIDER-ID', ' SAML Attribute Service V2.0', 'PROVIDER-ID', '0', '3600');

INSERT INTO svc_context (sc_id, sc_smd_id, sc_sequence)
VALUES ('PROVIDER-ID','PROVIDER-ID', '1');

INSERT INTO ep_context (epc_id, epc_sc_id)
VALUES ('PROVIDER-ID','PROVIDER-ID');

INSERT INTO versions (vs_epc_id, vs_version, vs_sequence)
VALUES ('PROVIDER-ID','2.0', '0');

INSERT INTO addresses (ad_epc_id, ad_address, ad_sequence)
VALUES ('PROVIDER-ID','SAML-ADDRESS', '0');

INSERT INTO secmechs (sm_epc_id, sm_mech_uri, sm_sequence)
VALUES ('PROVIDER-ID','urn:oasis:names:tc:SAML:2.0:assertion', '0');

INSERT INTO svc_types (st_sc_id, st_svc_type, st_sequence)
VALUES ('PROVIDER-ID','urn:oasis:names:tc:SAML:2.0:metadata:AttributeService', '0');
```

Where all occurrences of PROVIDER-ID are replaced with the Shibboleth entity ID of the service and all occurrences of SAML-ADDRESS should be replaced with the endpoint address of the IdPs Attribute service.

e.g.

```
INSERT INTO providers (pr_id, pr_abstract)
VALUES ('https://issrg-testidp.kent.ac.uk/idp/shibboleth', 'SAML Attribute Service
V2.0');

INSERT INTO svc_metadata (smd_id, smd_abstract, smd_pr_id , smd_sequence,
smd_lifetime)
VALUES ('https://issrg-testidp.kent.ac.uk/idp/shibboleth', ' SAML Attribute Service
V2.0', 'https://issrg-testidp.kent.ac.uk/idp/shibboleth', '0', '3600');

INSERT INTO svc_context (sc_id, sc_smd_id, sc_sequence)
VALUES ('https://issrg-testidp.kent.ac.uk/idp/shibboleth','https://issrg-
testidp.kent.ac.uk/idp/shibboleth', '1');

INSERT INTO ep_context (epc_id, epc_sc_id)
VALUES ('https://issrg-testidp.kent.ac.uk/idp/shibboleth','https://issrg-
testidp.kent.ac.uk/idp/shibboleth');


INSERT INTO versions (vs_epc_id, vs_version, vs_sequence)
VALUES ('https://issrg-testidp.kent.ac.uk/idp/shibboleth','2.0', '0');


INSERT INTO addresses (ad_epc_id, ad_address, ad_sequence)
VALUES ('https://issrg-testidp.kent.ac.uk/idp/shibboleth','https://issrg-
testidp.kent.ac.uk/idp/profile/SAML2/SOAP/AttributeQuery', '0');

INSERT INTO secmechs (sm_epc_id, sm_mech_uri, sm_sequence)
VALUES ('https://issrg-
testidp.kent.ac.uk/idp/shibboleth','urn:oasis:names:tc:SAML:2.0:assertion', '0');

INSERT INTO svc_types (st_sc_id, st_svc_type, st_sequence)
VALUES ('https://issrg-testidp.kent.ac.uk/idp/shibboleth',
'urn:oasis:names:tc:SAML:2.0:metadata:AttributeService', '0');
```

A slightly modified version of the same script should also be used to add the discovery service addresses of each linking service and IdP that will be used with the service.

```
INSERT INTO providers (pr_id, pr_abstract)
VALUES ('PROVIDER-ID', 'Liberty Discovery Service V2.0');

INSERT INTO svc_metadata (smd_id, smd_abstract, smd_pr_id , smd_sequence,
smd_lifetime)
VALUES ('PROVIDER-ID', 'Liberty Discovery Service V2.0', 'PROVIDER-ID', '0', '3600');

INSERT INTO svc_context (sc_id, sc_smd_id, sc_sequence)
VALUES ('PROVIDER-ID','PROVIDER-ID', '1');

INSERT INTO ep_context (epc_id, epc_sc_id)
VALUES ('PROVIDER-ID','PROVIDER-ID');


INSERT INTO versions (vs_epc_id, vs_version, vs_sequence)
VALUES ('PROVIDER-ID','2.0', '0');


INSERT INTO addresses (ad_epc_id, ad_address, ad_sequence)
VALUES ('PROVIDER-ID','LIBERTY-ADDRESS', '0');

INSERT INTO secmechs (sm_epc_id, sm_mech_uri, sm_sequence)
VALUES ('PROVIDER-ID','urn:oasis:names:tc:SAML:2.0:assertion', '0');

INSERT INTO svc_types (st_sc_id, st_svc_type, st_sequence)
VALUES ('PROVIDER-ID','urn:liberty:disco:2006-08', '0');
```

Where PROVIDER-ID is the entity ID of the linking service or IdP and LIBERTY-ADDRESS is the endpoint address of the service. If you have been following this guide then LIBERTY-ADDRESS will be of the following form: https://<HOSTNAME>:8443/axis/services/LibertyDS2, with <HOSTNAME> being the domain name of your server.

2. The next step is to edit the IdPs Shintau configuration file found in the $IDP_HOME/conf/shintau.xml file. This file contains configuration details that the IdP then uses to make its connection to the PostgreSQL databse created by the discovery service. You should change the username, password and database variables to match those used by the discovery service.
3. The next step is to edit the Shibboleth IdP's federation metadata to take into account the Liberty Alliance Discovery Service endpoint that is provided by the Discovery Service. If you have registered your IdP with the test shib federation then you should be able to change your IdPs metadata by logging into the testhib 2 site clicking register and editing your existing IdP entry. If however you have registered you Idp with another federation you will need to contact the owner of your federation for instructions on updating your metadata entry.

   Two changes are required to the existing metadata in order to support Shintau aggregation the first of these states that the IdP supports the Liberty Alliance IDWSF Discovery Service protocol and the second states the endpoint at which the Discovery service can be reached. Please make the following change to your entities XML metadata:

   Add "*urn:liberty:disco:2006-08*" to the *IDPSSODescriptor* element's *protocolSupportEnumeration* attribute.

Add a new *SingleSignOnService* element with a binding of "urn:liberty:disco:2006-08" and an address of https://<HOSTNAME>:8443/axis/services/LibertyDS2 where <HOSTNAME> is the domain name of your server.
Make the same changes to the metadata describing your own service on your local hard drive. This metadata can be found at $IDP_HOME/metadata/idp-metadata.xml.

4. Configure the IdP to release persistent Identifiers when requested by editing the attribute release policy of the shibboleth service located at $IDP_HOME/conf/attribute-resolver.xml.

We first add the attribute definition of a persistent Id below:

```
    <resolver:AttributeDefinition id="persistentId" xsi:type="TransientId"
xmlns="urn:mace:shibboleth:2.0:resolver:ad">
        <resolver:AttributeEncoder xsi:type="SAML1StringNameIdentifier"
xmlns="urn:mace:shibboleth:2.0:attribute:encoder"
            nameFormat="urn:mace:shibboleth:1.0:nameIdentifier" />

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

    </resolver:AttributeDefinition>
```

In order for this definition to work me must also add the following definition to the principal connectors section of the file:

```
    <resolver:PrincipalConnector xsi:type="Transient"
xmlns="urn:mace:shibboleth:2.0:resolver:pc" id="saml2Persistent"
        nameIDFormat="urn:oasis:names:tc:SAML:2.0:nameid-format:persistent" />
```

Now that the persistent Id has been defined we need to add a release policy for its attribute so that Shibboleth releases the attribute when queried by a service provider entity. Please copy the attribute release policy for the persistent ID below into $IDP_HOME/conf/attribute-filter.xml :

```
    <!--  Release the Persistent ID to anyone -->
    <AttributeFilterPolicy id="releasePersistentIdToAnyone">
        <PolicyRequirementRule xsi:type="basic:ANY" />

        <AttributeRule attributeID="persistentId">
            <PermitValueRule xsi:type="basic:ANY" />
        </AttributeRule>

    </AttributeFilterPolicy>
```

Please note that if you wish to release any other attributes for authorisation they will also need to be defined in attribute-resolver.xml and attribute-filter.xml. The sample definition provided below will allow permisRole and attributeCertificateAttributes to be released from the publically accessible university of Kent LDAP server when added to the appropriate files.

$IDP_HOME/conf/attribute-resolver.xml :

```
<!-- permisROle definition -->
<resolver:AttributeDefinition id="permisRole" xsi:type="Scoped"
xmlns="urn:mace:shibboleth:2.0:resolver:ad"
```

```
<!-- Make sure to use your own hostname as the scope -->
        scope="issrg-testidp.kent.ac.uk" sourceAttributeID="permisRole">

        <resolver:Dependency ref="permisLDAP" />

         <resolver:AttributeEncoder xsi:type="SAML1ScopedString"
         xmlns="urn:mace:shibboleth:2.0:attribute:encoder"
         name="permisRole" />

        <resolver:AttributeEncoder xsi:type="SAML2ScopedString"
       xmlns="urn:mace:shibboleth:2.0:attribute:encoder"
         name="permisRole" friendlyName="permisRole" />
</resolver:AttributeDefinition>

    <!-- X.509 AC definition -->
<resolver:AttributeDefinition id="attributeCertificateAttribute" xsi:type="Simple"
xmlns="urn:mace:shibboleth:2.0:resolver:ad"
sourceAttributeID="attributeCertificateAttribute">

       <resolver:Dependency ref="permisLDAP"/>

       <resolver:AttributeEncoder xsi:type="SAML1String"
       xmlns="urn:mace:shibboleth:2.0:attribute:encoder"
       name="urn:oid:2.5.4.58"/>

       <resolver:AttributeEncoder xsi:type="SAML2String"
       xmlns="urn:mace:shibboleth:2.0:attribute:encoder"
       name="urn:oid:2.5.4.58" friendlyName="attributeCertificateAttribute" />

</resolver:AttributeDefinition>

<!-- LDAP data connector for Kent LDAP -->

<resolver:DataConnector id="permisLDAP" xsi:type="LDAPDirectory"
xmlns="urn:mace:shibboleth:2.0:resolver:dc"
ldapURL="ldap://sec.cs.kent.ac.uk" baseDN="o=permisv5,c=gb" principal=""
principalCredential="">

       <FilterTemplate>
              <![CDATA[
                     (uid=$requestContext.principalName)
              ]]>
       </FilterTemplate>

       <ReturnAttributes>permisRole uid dn cn attributeCertificateAttribute
       </ReturnAttributes>


       <!-- tell that attributeCertificateAttribute is a binary blob -->
       <LDAPProperty name="java.naming.ldap.attributes.binary"
       value="attributeCertificateAttribute"/>
</resolver:DataConnector>
```

$IDP_HOME/conf/attribute-filter.xml:

```
<!--  Release the permisRole to anyone -->
    <AttributeFilterPolicy id="releasePermisRoleToAnyone">
        <PolicyRequirementRule xsi:type="basic:ANY" />

        <AttributeRule attributeID="permisRole">
            <PermitValueRule xsi:type="basic:ANY" />
        </AttributeRule>

    </AttributeFilterPolicy>

<!-- release attributeCertificateAttribute to anyone -->
     <AttributeFilterPolicy id="releaseX509ACToAnyone">
        <PolicyRequirementRule xsi:type="basic:ANY" />

        <AttributeRule attributeID="attributeCertificateAttribute">
            <PermitValueRule xsi:type="basic:ANY" />
        </AttributeRule>

    </AttributeFilterPolicy>
```

3. Linking Service (LS) Installation

Within the Shintau environment an Identity Provider entity consists of two entities, a standard Shibboleth 2 Service Provider (http://shibboleth.internet2.edu/) protecting a PHP based frontend used to Link user accounts together and a modified IDWSF Liberty Alliance Discovery service based upon the IDWSF Server Toolkit (http://www.cahillfamily.com/OpenSource/) running under Axis 1.3. Both of these entities use a PostgreSQL database to store service information. The SP and PHP frontend both run under an Apache HTTPD Server and the IDWSF Liberty Alliance Discovery Service runs inside an Apache Tomcat Server.

2.1 Prerequisites for the Linking Service Installation

When installing the Identity Provider we assume that the following software has been installed:

1. The Sun java development kit, either release 1.5 or 6 (Please note that non Sun implementations of the Java JDK are not compatible with this software).
2. The Apache Ant java installation and testing suite.
3. A recent version of OpenSSL .

We also presume that the JAVA_HOME and ANT_HOME environment variables have been set to the root directory of your compatible JDK and ant installations respectively.

The following applications should also be installed using the instructions provided in the above section detailing the Identity Provider installation before continuiong with this guide:

1. Apache Tomcat 5.5.* (Section 2.2)
2. Apache Axis 1.3 (Section 2.3)
3. PostgreSQL (Section 2.4)
4. Apache HTTPD with SSL support (Section 2.5)

2.2 PHP 5 Installation

In order to allow the Apache HTTPD server to display dynamically generated pages the PHP server side processing language must be installed on the machine hosting the Apache server and then Apache configured to recognise PHP pages. The PHP source distribution can be downloaded from the PHP project website (http://www.php.net).

1. Once the download is complete extract the tar file to a directory of your choosing and navigate into the newly created directory:

   $ tar –xzf php-5.2.9.tar.gz

   $ cd php-5.2.9

2. Run the ./configure script setting the following options before using GNU make to install and deploy the server using the make and make install commands:

   --prefix : This option should be set to /usr/local/php to specify the location to install new files
   --with-apxs2: This option should be set to the path to your apache server's apxs application (/usr/local/apache/bin/apxs)
   --with-openssl: This option should be set in order to enable PHP to use OpenSSL.

--enable-soap: This option should be set to allow PHP to use SOAP messaging.
--with-pgsql: This option should be set to the base directory of your PostgreSQL installation (/usr/local/postgresql) to allow PHP to access the PostgreSQL database.
e.g.

```
$ ./configure --prefix=/usr/local/php --with-apxs2=/usr/local/apache/bin/apxs --with-
openssl --enable-soap --with-pgsql=/usr/local/postgresql
$ make
$ make test
$ sudo make install
```

3. Configure Apache to load and utilise the PHP 5 module. First check to ensure that the PHP 5 module has been included in Apache's httpd.conf configuration file (/usr/local/apache/conf/httpd.conf) by the installation script by ensuring that the Line below is present (add it if it is not):

   LoadModule php5_module modules/libphp.so

   Tell Apache to serve index.php files as well as index.html files when a directory is requested by Modifying the DirectoryIndex parameter

   ```
   DirectoryIndex index.html index.php
   ```

   Tell Apache to parse certain extensions as PHP. By adding the following lines to the httpd.conf file

   ```
   AddType application/x-httpd-php .html .php .foo

   AddType application/x-httpd-php-source .phps .phtmlss
   ```

4. Create a new file test.php in the root html directory of your apache installation (/usr/local/apache/htdocs/) and add the following text:

   ```
   <?php phpinfo() ?>
   ```

   Save the file and exit.

5. Restart Apache and attempt to navigate to http://localhost/test.php in your browser. If a list of php information is displayed then PHP 5 has been installed correctly.

2.3 Shibboleth 2.0 Service Provider (SP) Installation

There are several different methods of installing a Shibboleth 2 service provider entity and the correct one to use is dependent on the architecture of the system. We cannot document all of them here and instead we provide a link to the relevant installation documentation provided by internet2:

https://spaces.internet2.edu/display/SHIB2/Installation

Please follow the installation guide for the Shibboleth SP and once the server is installed please follow the steps below to configure your SP for use with the Shintau architecture.

For the purposes of this document we assume that you have registered your LS with the test shib 2 federation provided by internet2 available from http://www.testshib.org/testshib-two/index.jsp if

you are using a different federation please configure your metadata and federation details as needed.

2.4 Liberty Alliance IDWSF Discovery Service Installation

The Liberty Alliance IDWSF Discovery Service is used to provide EPR attributes to SP services that query for additional IdP accounts when aggregation. It generates these based on the contents of the database filled by the PHP frontend. It should be installed in the following manner:

1. Download and extract the JUnit java testing suite from http://junit.org extract the distribution and move it to /usr/local/junit.

```
$ tar –xzf  junit4-5.tar.gz
$ mv junit4-5 /usr/local/junit
```

2. Download the modified IDWSF liberty Alliance Server toolkit from the PERMIS website, extract it to a convenient directory and navigate into the newly created directory. This directory shall henceforth be referred to as $DISCO_HOME for the purposes of this installation guide.

```
$ tar -xzf shintau-LibertyIDWSF.tar.gz

$ cd shintau-libertyIDWSF
```

3. Copy the Postgresql JDBC Java connector from the root of the extracted directory to the axis common library so that axis can use it.
   e.g.

```
$ sudo cp ./postgresql-jdbc-3.5.jar /usr/local/tomcat/webapps/axis/WEB-INF/lib/
```

4. Edit the program defaults described in Constants.java. We must first edit the basic program definitions defined in $DISCO_HOME/install/idwsf/src/liberty/toolkit/constants.java. These definitions determine the URLs of the discovery service when installed. We only need edit the basic definitions shown below:

```
        /*
         * Basic Defs
         */

public static /*final*/ String
               strMyHostSSL= "https://issrg-testls.kent.ac.uk:8443/";
public static /*final*/ String
               strMyHostNonSSL= "http://issrg-testls.kent.ac.uk:8080/";
public static /*final*/ String
       strMyPIDBase= "https://issrg-testls.kent.ac.uk/";
public static /*final*/ String
               strAxisPath= "axis/services/";
```

   The first of these settings strMyHostSSL defines the SSL endpoint used by tomcat and the second strMyHostNonSSL defines the normal http endpoint used by tomcat. The third setting defines the hostname of the service and the final details the path required to access the axis services installed on the tomcat server.

5. Edit the Shintau specific defaults described in SetupLinking.java. These are used to determine whether the server is to use Shintau aggregation and if so how to access the database. Open the file $DISCO_HOME/install/ds2/src/issrg/linking/SetupLinking.java.

```
public static final String pGSQLuser = "issrg";
public static final String pGSQLPassword = "password";
public static final String pGSQLDatabase = "lib-idwsf";

public static final boolean isIdP = false;
public static final boolean useShintau = true;
```

   The first three variables are used to define the connection to the postgresql server and the username and password should correspond to one of the user's created when setting up postgresql. As this is a Linking Service installation isIdP should be false as in the example above and the useShintau should also be true to enable Shintau Aggregation rather than the default behaviour.

6. Edit the DBSettings file found in the $DISCO_HOME/install/db/Setup directory. This file determines the database settings to be used when creating the service's PostgresSQL database. You should edit the LIBUSER and LIBPASS variables so that they represent the same user defined in the ShintauSetup file. The rest of the file should not be modified. In the same directory, change the MyHostName variable in the FillDB file.
7. Change the username/password combination in the file $DISCO_HOME/install/db/src/liberty/db/DBCore.java. This is the username/password combination that will be used for accessing the discovery service database.

8. In order to create the database the /var/lib/pgsql directory needs to be created for the database files to be stored in. This directory should also be owned by the root PostgreSQL user postgres so that the postgres database can write to the directory.

```
$ sudo mkdir /var/lib/pgsql
$ sudo chown postgres.postgres /var/lib/pgsql
```

9. Creating and initialising the databse with an intial set of values is the next step and this can be accomplished by running the InitDB, FillDB and SetupDB scripts located in the $DISCO_HOME/install/db/Setup directory. Please note that when using the sudo command the postgresql bin directory may need to be added to the PATH variable in order to successfully run these commands.

```
$ cd $DISCO_HOME/db/Setup
$ sudo PATH=$PATH:/usr/local/postgresql/bin ./InitDB
$ sudo PATH=$PATH:/usr/local/postgresql/bin ./FillDB
$ sudo PATH=$PATH:/usr/local/postgresql/bin ./SetupDB
```

10. Run the ant installation script from the installation directory (install/). This requires your PATH environment variable to contain links to the java bin directory, the postgresql bin directory and the tomcat bin directory. An extensive CLASSPATH environment variable must also be set containing all the AXIS library JAR files, the JUnit JAR, the PostgreSQL JDBC

connector JAR and servlet.jar from the tomcat common directory. When using sudo to run the ant script these variables should prefix the ant command to be run. The ant installation parameter is called install and an example command including the required classpath is shown below. Note; it might be easier to use an interactive 'root' session so that it is easier to set the different environment variables. If you would like to be on the safe side, you can run 'ant fullclean' first (making sure that all environment variables are set).

```
$ sudo ANT_HOME=/usr/share/ant/ PATH=$PATH:/usr/bin:/usr/lib/jvm/java-6-
sun/bin:/usr/local/tomcat/bin:/usr/local/postgresql/bin/
CLASSPATH=$CLASSPATH:/usr/local/axis/lib/axis.jar:/usr/local/axis/lib/saaj.jar:/usr/lo
cal/axis/lib/commons-logging-1.0.4.jar:/usr/local/axis/lib/commons-discovery-
0.2.jar:/usr/local/axis/lib/wsdl4j-1.5.1.jar:/usr/local/axis/lib/log4j-
1.2.8.jar:/usr/local/tomcat/common/lib/servlet.jar:/usr/local/axis/
lib/jaxrpc.jar:/usr/local/tomcat/common/endorsed/xalan-
2.7.1.jar:/usr/local/tomcat/common/endorsed/serializer-
2.9.1.jar:/usr/local/tomcat/common/endorsed/xercesImpl-
2.9.1.jar:/usr/local/tomcat/common/endorsed/xml-apis-
2.9.1.jar:/usr/local/tomcat/common/lib/postgresql-jdbc3.jar:/usr/local/junit/junit-
4.5.jar ant install
```

Please note that there may be an error running the installation script due to compilation problems when compiling downloaded wsdl files to java. If this occurs then you should overwrite the wsdl/build directory with the schema files located in old-wsdl/build and rerun the script e.g.

```
$ sudo cp old-wsdl/build install/wsdl/
$ sudo ANT_HOME=/usr/share/ant/ PATH=$PATH:/usr/bin:/usr/lib/jvm/java-6-
sun/bin:/usr/local/tomcat/bin:/usr/local/postgresql/bin/
CLASSPATH=$CLASSPATH:/usr/local/axis/lib/axis.jar:/usr/local/axis/lib/saaj.jar:/usr/lo
cal/axis/lib/commons-logging-1.0.4.jar:/usr/local/axis/lib/commons-discovery-
0.2.jar:/usr/local/axis/lib/wsdl4j-1.5.1.jar:/usr/local/axis/lib/log4j-
1.2.8.jar:/usr/local/tomcat/common/lib/servlet.jar:/usr/local/axis/
lib/jaxrpc.jar:/usr/local/tomcat/common/endorsed/xalan-
2.7.1.jar:/usr/local/tomcat/common/endorsed/serializer-
2.9.1.jar:/usr/local/tomcat/common/endorsed/xercesImpl-
2.9.1.jar:/usr/local/tomcat/common/endorsed/xml-apis-
2.9.1.jar:/usr/local/tomcat/common/lib/postgresql-jdbc3.jar:/usr/local/junit/junit-
4.5.jar ant install
```

11. Install the service in axis and tomcat, this is accomplished by using the ant "config" command. In order to successfully complete this step the tomcat server must be running and the axis administration servlet must have been enabled.
    e.g.

```
$ sudo ANT_HOME=/usr/share/ant/ PATH=$PATH:/usr/bin:/usr/lib/jvm/java-6-
sun/bin:/usr/local/tomcat/bin:/usr/local/postgresql/bin/
CLASSPATH=$CLASSPATH:/usr/local/axis/lib/axis.jar:/usr/local/axis/lib/saaj.jar:/usr/lo
cal/axis/lib/commons-logging-1.0.4.jar:/usr/local/axis/lib/commons-discovery-
0.2.jar:/usr/local/axis/lib/wsdl4j-1.5.1.jar:/usr/local/axis/lib/log4j-
1.2.8.jar:/usr/local/tomcat/common/lib/servlet.jar:/usr/local/axis/
lib/jaxrpc.jar:/usr/local/tomcat/common/endorsed/xalan-
2.7.1.jar:/usr/local/tomcat/common/endorsed/serializer-
2.9.1.jar:/usr/local/tomcat/common/endorsed/xercesImpl-
2.9.1.jar:/usr/local/tomcat/common/endorsed/xml-apis-
2.9.1.jar:/usr/local/tomcat/common/lib/postgresql-jdbc3.jar:/usr/local/junit/junit-
4.5.jar ant config
```

    You should now restart tomcat and you should be able to validate that the discovery service has been installed by checking the list of deployed axis services (http://localhost:8080/axis/services). Note that there is no '/' at the end of this URL!

12. Configure the discovery service's entity Identifier so that it associates the SPs entity identifier with the Service's Discovery Service Endpoint. This is accomplished by running the SQL script below which will create a new service definition for the endpoint. You should run the script while being connect to the postgres database, for instance through the graphical user interface pgAdmin III.

```
INSERT INTO providers (pr_id, pr_abstract)
VALUES ('PROVIDER-ID', 'Liberty Discovery Service V2.0');

INSERT INTO svc_metadata (smd_id, smd_abstract, smd_pr_id , smd_sequence,
smd_lifetime)
VALUES ('PROVIDER-ID', 'Liberty Discovery Service V2.0', 'PROVIDER-ID', '0', '3600');

INSERT INTO svc_context (sc_id, sc_smd_id, sc_sequence)
VALUES ('PROVIDER-ID','PROVIDER-ID', '1');

INSERT INTO ep_context (epc_id, epc_sc_id)
VALUES ('PROVIDER-ID','PROVIDER-ID');

INSERT INTO versions (vs_epc_id, vs_version, vs_sequence)
VALUES ('PROVIDER-ID','2.0', '0');

INSERT INTO addresses (ad_epc_id, ad_address, ad_sequence)
VALUES ('PROVIDER-ID','LIBERTY-ADDRESS', '0');

INSERT INTO secmechs (sm_epc_id, sm_mech_uri, sm_sequence)
VALUES ('PROVIDER-ID','urn:oasis:names:tc:SAML:2.0:assertion', '0');

INSERT INTO svc_types (st_sc_id, st_svc_type, st_sequence)
VALUES ('PROVIDER-ID','urn:liberty:disco:2006-08', '0');
```

Where all occurrences of PROVIDER-ID are replaced with the Shibboleth entity ID of the service (https://<HOSTNAME>/shibboleth-sp) and all occurrences of Liberty-ADDRESS should be replaced with the endpoint address of the SPs new Discovery Service endpoint (https://<HOSTNAME>:8443/axis/services/LibertyDS2).

e.g.

```
INSERT INTO providers (pr_id, pr_abstract)
VALUES ('https://issrg-testls.kent.ac.uk/shibboleth-sp', 'Liberty Discovery Service
V2.0');

INSERT INTO svc_metadata (smd_id, smd_abstract, smd_pr_id , smd_sequence,
smd_lifetime)
VALUES ('https://issrg-testls.kent.ac.uk/shibboleth-sp', ' SAML Attribute Service
V2.0', 'https://issrg-testls.kent.ac.uk/shibboleth-sp', '0', '3600');

INSERT INTO svc_context (sc_id, sc_smd_id, sc_sequence)
VALUES ('https://issrg-testls.kent.ac.uk/shibboleth-sp','https://issrg-
testls.kent.ac.uk/shibboleth-sp', '1');

INSERT INTO ep_context (epc_id, epc_sc_id)
VALUES ('https://issrg-testls.kent.ac.uk/shibboleth-sp','https://issrg-
testls.kent.ac.uk/shibboleth-sp');


INSERT INTO versions (vs_epc_id, vs_version, vs_sequence)
VALUES ('https://issrg-testls.kent.ac.uk/shibboleth-sp','2.0', '0');


INSERT INTO addresses (ad_epc_id, ad_address, ad_sequence)
VALUES ('https://issrg-testls.kent.ac.uk/shibboleth-sp','https://issrg-
testls.kent.ac.uk:8443/axis/services/LibertyDS2', '0');

INSERT INTO secmechs (sm_epc_id, sm_mech_uri, sm_sequence)
VALUES ('https://issrg-testls.kent.ac.uk/shibboleth-
sp','urn:oasis:names:tc:SAML:2.0:assertion', '0');
```

```
INSERT INTO svc_types (st_sc_id, st_svc_type, st_sequence)
VALUES ('https://issrg-testls.kent.ac.uk/shibboleth-sp',
‘urn:oasis:names:tc:SAML:2.0:metadata:AttributeService’, '0');
```

You should also use the same script to add the Discovery Service endpoint of each of your Shintau enabled IdPs, replacing all instances of PROVIDER-ID with the IdP’s shibboleth entity id (https://<HOSTNAME>/idp/shibboleth) and the LIBERTY-ADDRESS with the IdPs discovery service endpoint.

13. The next step is to edit the Shibboleth SP’s federation metadata to take into account the Liberty Alliance Discovery Service endpoint that is provided by the Discovery Service. If you have registered your IdP with the test shib federation then you should be able to change your IdPs metadata by logging into the testhib 2 site clicking register and editing your existing SP entry. If however you have registered you Idp with another federation you will need to contact the owner of your federation for instructions on updating your metadata entry.

    The following changes should be made to the metadata, after the SPSSODescriptor element the following template should be copied into the file:

```
       <md:IDPSSODescriptor protocolSupportEnumeration=" urn:liberty:disco:2006-08"
xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata">
            <md:Extensions>
                <saml1md:Scope
xmlns:saml1md="urn:mace:shibboleth:metadata:1.0">scope</saml1md:Scope>
            </md:Extensions>
            <md:KeyDescriptor>
                <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                    <ds:X509Data xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                        <ds:X509Certificate
xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                    <!—Place SP Public Key Cert Here -->
                    </ds:X509Data>
                </ds:KeyInfo>
            </md:KeyDescriptor>
       <md:SingleSignOnService Binding="urn:liberty:disco:2006-08"
Location="https://HOSTNAME:8443/axis/services/LibertyDS2"/>
        </md:IDPSSODescriptor>
```

    You should then change the sections in bold to mach your service. The scope attribute should become the domain name of your service e.g. “kent.ac.uk”. You should copy the public key certificate used by the SP in to the <ds:X509Certificate> element and you should change the discovery service address of the service to match the hostname of your service.

14. You should now restart both the Apache Tomcat service and the Shibboleth SP (by restarting the shibd process).

2.5 Complete the Linking Service Installation

Extract the Linking Service distribution, inside you should find three folders shib_config, htdocs and apache_config.

1. The first folder should hold two files linkingServiceTemplate.html and shibboleth2.xml.example copy these files to your shibboleth sp configuration directory (/opt/shibboleth-sp/etc/shibboleth).

```
$ sudo cp shib_config/* /opt/shibboileth-sp/etc/shibboleth/
```

2. The second folder contains the PHP files that make up the Apache/PHP frontend and should be copied into the DocumentRoot directory of your Apache SSL installation.

```
$ sudo cp  -R htdocs/* /usr/local/apache/htdocs/www/
```

3. The third folder contains the linking.conf apache configuration file which when included by Apache protects the login1 directory of the service with Shibboleth this should be copied into the $APACHE_HOME/conf directory.

```
$ sudo cp apache_config/linking.conf $APACHE_HOME/conf/

$ cd $APACHE_HOME/conf/
```

   You should then edit your $APACHE_HOME/conf/extra/httpd-ssl.conf file and add the following line inside the SSL VirtualHost element to include the linking service configurations.

```
Include conf/linking.conf
```

4. Navigate into the shibboleth configuration directory(/opt/shibboleth-sp/etc/shibboleth) and add the following lines to shibboleth2.xml, an example configuration file is provided in shibboleth2.xml.example :

   First edit your logout initiator by commenting out the existing logout initiatr and replacing it with a new one that redirects the user to <HOSTNAME>/login1/index.php

   e.g.

```
   <!-- LogoutInitiators enable SP-initiated local or global/single logout of
sessions. -->
            <LogoutInitiator type="Chaining" Location="/Logout" relayState="cookie">
<!--
<LogoutInitiator type="SAML2" template="bindingTemplate.html"
return="https://<HOSTNAME>/login1/index.php"/>
-->
                <LogoutInitiator type="Local"
        return="https://<HOSTNAME>/login1/index.php"/>
          </LogoutInitiator>
```

   Replacing <HOSTNAME> with the hostname of the service.

   If you are utilising the testshib federation you should then replace the default testshib login handler with the one below and make sure that the handler below is the default handler for this SP:

```
<SessionInitiator type="Chaining" Location="/DS"
                id="DS" isDefault="true" relayState="cookie">
        <SessionInitiator type="Transform">
                <Subst>https://testidp.$entityID/idp/shibboleth</Subst>
                <Regex match=".+@(.+)">https://testidp.$1/idp/shibboleth</Regex>
        </SessionInitiator>
        <SessionInitiator type="SAML2" defaultACSIndex="1"
acsByIndex="false" template="bindingTemplate.html">
```

```
        <samlp:AuthnRequest xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
        AssertionConsumerServiceIndex="1"
        Destination="https://hostname.kent.ac.uk/idp/profile/SAML2/Redirect/SSO"
        ID="_010493171056268e60828d4db4613df8"
        IssueInstant="2009-03-22T12:43:33Z" Version="2.0"
        ForceAuthn="1">
                <saml:Issuer
                        xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
                        https://<HOSTNAME>/idp/shibboleth
                </saml:Issuer>
                <samlp:NameIDPolicy
                Format="urn:oasis:names:tc:SAML:2.0:nameid-format:persistent"
AllowCreate="1"/>
                </samlp:AuthnRequest>
</SessionInitiator>
        <SessionInitiator type="Shib1" defaultACSIndex="5"/>
        <SessionInitiator type="Form" template= "discoveryTemplate.html"/>
</SessionInitiator>
```

Replacing <HOSTNAME> above with the full hostname of the server on which the SP is running. This login handler provides a template for all authentication request made from the SP and if a federation WAYF file is used instead of the discovery template above this authnRequest must still be used to make the authentication request.

5. Edit the linkingServiceTemplate.html which you copied into the shibboleth configuration directory in the following ways:

    a. Replace the hostname in all the page's image URL paths, with the fully qualified hostname of the service.

    b. Manually edit the IdP entity entries in the drop down box to refer to your IdP entities e.g. To specify a new IdP with the entity ID https://issrg-testidp.kent.ac.uk/shibboleth/idp you would add the following entry inside the <select> element:

```
<select name="entityID" style="width:80%">
        <option value="https://issrg-testidp.kent.ac.uk/idp/shibboleth">
                issrg-testidp.kent.ac.uk
        </option>
</select>
```

6. Edit the PHP database settings found in the $APACHE_HOME/htdocs/php/idwsf-db.php file, to allow the PHP service to connect to the identity mapping database:

```
$username = "db-user";
$password = "password";
$dbname = "lib-idwsf";
```

The above variables should be edited to ensure that they are correct for your system.

7. The Linking Service should now be configured for use. To access the service please restart Apache, Tomcat and the shibd service before navigating to https://localhost/ and following the online instructions to link accounts at IdPs together.

Please Note: Before accounts can be linked together the IdP will need to add the LS service to its PostgreSQL database and described in Section 2.8 Step 1.

4. Installation of a Shintau enabled Service Provider

This guide describes the steps necessary to allow Shintau aggregation of attributes to be used when protecting a web based resource. We assume that you are working in a Linux environment and that you are reasonably confident working on its command line. Broadly, you will need to do the following:

- Install an Apache web server.

- Install a Shibboleth Service Provider

- Install the Shibboleth Apache Authorization Module (SAAM)

- Install the PERMIS standalone server.

- Configuration of the different components to work together.

4.1 Apache Web Server Installation

Important note: the SAAM software, which you will need later on, has not yet been confirmed to work with Apache's 2.2 web server. Therefore, you will need to install a 2.0.x webserver. It is recommended to install the latest web server software in the 2.0 release. At the time of writing the latest applicable version is 2.0.63.

1. First, download the software from the Apache site:

   ```
   $ wget http://mirror.public-internet.co.uk/ftp/apache/httpd/httpd-2.0.63.tar.gz
   ```

2. Next, unpack it:

   ```
   $ tar -xzvf httpd-2.0.63.tar.gz
   ```

   This will create a directory, called httpd-2.0.63, which holds all the files necessary to compile the web server.

3. Set the APACHE_HOME variable to where you want to install the web server to.

   ```
   $ export APACHE_HOME=/usr/local/apache2.0
   ```

4. Next, configure your apache installation for compilation, by moving into the directory created by the tar command and running the ./configure script. Once the ./configure script has completed use GNU make to compile and install the server :

   ```
   $ cd httpd-2.0.63
   $ ./configure --prefix=$APACHE_HOME --enable-so  --enable-mods-shared=all --enable-
   proxy --enable-ssl
   $ make
   $ sudo make install
   ```

5. The next step is to install your server certificate so that the web server uses it. We assume that your server certificate and server key are called server.crt and server.key respectively and that they are both located in $APACHE_HOME/conf. Change the $APACHE_HOME/conf/ssl.conf file such that the SSLCertificateFile and SSLCertificateKeyFile point to these files:

```
SSLCertificateFile /usr/local/apache2.0/conf/server.crt

SSLCertificateKeyFile /usr/local/apache2.0/conf/server.key
```

6. Start the apache web server:

```
$APACHE_HOME/bin/apachectl  -D SSL start
```

and check that it is running by visiting: http://localhost/ and https://localhost/ .

Both locations should display Apache's "It works" page.

4.2 Installing a Standard Shibboleth SP

Since the Shibboleth documentation on the Internet2 website is very comprehensive and easy to follow we don't repeat its contents here. You should install a native Shibboleth2 Service provider on Linux. The relevant document can be found at https://spaces.internet2.edu/display/SHIB2/NativeSPLinuxInstall

Make sure that your service provider is working correctly, for instance by using the TestShib federation.

4.3. Install the Shibboleth Apache Authorization Module

First, you will need to download the SAAM software from the PERMIS website. Make sure you download version 5.1.x of the SAAM software. The 5.0.x versions will not work if you want to use Shintau authorization.

In order to download PERMIS software you need to register. You can do this by visiting this page: http://sec.cs.kent.ac.uk/permis/essentials/register.shtml. You will then receive a username/password combination that can be used to download the software:

$ wget --user=the_user_name --password=the_password \
http://sec.cs.kent.ac.uk/permis/private/saam/saam_5_1_0.zip

Unzip the file into a directory called 'saam'

$ unzip saam_5_1_0.zip -d saam

4.3.1 Compilation and Installation of the mod_permis Module

In order for the compilation of the mod_permis module to succeed, you need to have a Sun JDK implementation installed on your system. Enter the saam/mod_permis_src directory:

In order to compile the mod_permis module we have to update the variable LD_LIBRARY_PATH. This is done in reference to the JAVA_HOME variable:

```
$ export JAVA_HOME=/usr/lib/jvm/java-6-sun

$ export
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$JAVA_HOME/jre/lib/i386/client:$JAVA_HOME/jre/lib/i386
```

Please Note: The LD_LIBRARY_PATH variable is also required when running the SAAM module in Apache and without it being set the module will fail to work.

Ensure that the APACHE_HOME variable is still set:

```
$ export APACHE_HOME=/usr/local/apache2.0
```

Next, make sure that the file make20.sh is executable and execute it:

```
$ chmod 755 make20.sh ; ./make20.sh
```

Note: if the compilation fails saying something like

"permisJNI.c:21:17: error: jni.h: No such file or directory" then this probably means that you don't have a JDK installed or you LD_LIBRARY_PATH variable is improperly set. In particular, your $JAVA_HOME directory should contain a directory called 'include'.

The next step is to copy the module to the Apache modules directory:

```
$ sudo cp mod_permis_20.so $APACHE_HOME/modules
```

4.2 Copy the libraries

We suggest that you create a new directory on your system called usr/local/saam . Copy the saam.jar and the saam_include directory found in the original saam directory into the new one. Then copy the log4j.properties file also included in the release to your apache server's conf directory.

4.4. Install the PERMIS Standalone Server

Installing the standalone server is easy: download the standalone software and unpack it into the directory of your choice. Navigate into this directory and make the file standalone.sh executable.

```
$ chmod 755 standalone.sh
```

Starting the standalone server is done by running the standalone.sh script:

```
$ ./standalone.sh
```

The main configuration file for the standalone server is called permis.xml and is located in the same directory as the standalone.sh file. An example file is provided to get you started.

The configuration file starts by providing meta information like the port the service is going to listen on and the number of threads that may be used by the service. This is done in the <TCPConfiguration> element like this:

```
<TCPConfiguration>
  <ServerPort>1104</ServerPort>
  <ThreadCount>10</ThreadCount>
</TCPConfiguration>
```

Next are one or more <PermisConfiguration> elements each of which describe a different PERMIS Policy Decision Point, loaded with a different policy and with different configuration parameters. There are two main modes of operation:

The policy can either be located on the local hard drive (in an XML file) or it can be stored in a repository (typically an LDAP repository) stored in an X.509 Attribute Certificate. When using Shintau attribute aggregation additional elements are needed as shown below:

Parameters common to all policy configurations:

```
<PERMISConfiguration>

  <!—Either describes where the policy can be found on the local hard drive. Or supplies a
Repository Location e.g. ldap://sec.cs.kent.ac.uk/c=gb -->

  <PolicyLocation>/home/issrg/policies/shib-policy.xml</PolicyLocation>

  <!-- Ignored when using text policies, but has to be there anyway -->

  <PolicyIssuer>cn=The Policy Issuer, c=gb</PolicyIssuer>

  <!-- For XML policies the Policy Identifier may have any unique value but MUST still be set
-->

  <PolicyIdentifier>shib-policy</PolicyIdentifier>

  <!-- Ignored when no LDAP repository is being used, but has to be set -->

  <LDAP_AC_Attribute>attributeCertificateAttribute</LDAP_AC_Attribute>

  <LDAP_PKC_Attribute>userCertificate</LDAP_PKC_Attribute>
```

Shintau specific parameters:

```
  <!-- The following needs to be set when using Shintau Attribute Aggregation -->

  <Shintau_Attribute>true</Shintau_Attribute>

  <!-- Path to a JKS keystore containing the private key of the service -->

  <Shintau_Private_Key>/home/issrg/software/new_standalone/keystore.jks</Shintau_Private_Key>

  <!-- Password to open up this key store -->

  <Shintau_Private_Key_Password>password</Shintau_Private_Key_Password>

<!-- Path to the public Key certificate of the service - ->

<Shintau_Public_Key>/home/issrg/issrg-shintau1.crt</Shintau_Public_Key>

 <!-- The trust store containing the CAs of the trusted certificates -->

  <Shintau_TrustStore>/home/issrg/software/new_standalone/truststore.jks</Shintau_TrustStore>

<!-- Password to open up this key store -->

 <Shintau_TrustStore_Password>changeit</Shintau_TrustStore_Password>

 <!-- The entity ID of the Service Provider as described in the metadata -->

 <Shintau_ProviderID>https://issrg-shintau1.kent.ac.uk/shibboleth-sp</Shintau_ProviderID>

</PERMISConfiguration>
```

The two JKS trust store objects should be configured in the following way:

1. The Key Store object should contain the private and public key pair of the SP service that the Shintau service is linked too, to allow the service to send and receive aggregation messages on behalf of the shibboleth SP.

2. The Trust Store object should contain the SSL certificates of each entity in the federation that the SP has a trust relationship with (in future releases this trust store will be generated automatically from the SP metadata). The alias under which the certificate is stored should match the hostname contained in the certificate e.g. for the public key certificate of issrg-testls.kent.ac.uk we would add the certificate to the trust store in the following manner:

   ```
   $ keytool -import -trustcacerts -alias "issrg-testls.kent.ac.uk" -file CAcert.crt -
   keystore truststore.jks
   ```

After any changes are made to the configuration file the server must be restarted to reflect the changes made.

Please Note: All path configuration parameters should provide the absolute path to the required Object.

4.5. Overall Configuration

Apache needs to be configured in the following way. From the main configuration file you should include a file with the following content to configure both Shibboleth and Permis:

```
######
## SHIB Config
######
#
# Load the PERMIS and SHIBBOLETH module. PERMIS module should come first
#
LoadModule mod_permis /usr/local/apache2.0/modules/mod_permis_20.so
LoadModule mod_shib /opt/shibboleth-sp/lib/shibboleth/mod_shib_20.so

# This must match the <PolicyIdentifier> element in the permis.xml configuration file
# of the standalone server
PermisPolicyIdentifier shib-policy

# The host the standalone server is running on
PermisServerHost localhost
# The port on which it is listening. This is specified in the <ServerPort> element of
# the permis.xml configuration file
PermisServerPort 1104
# The location of the saam.jar file
PermisJavaClass "/usr/local/saam/saam.jar"
# The location of a Log4J properties file for the Java code of SAAM
PermisLog4jLocation /usr/local/apache2.0/conf/permis.log4j.properties
# The log level for the C code of the saam module
PermisDebug debug
#
# Used for example logo and style sheet in error templates.
#
<IfModule mod_alias.c>
  <Location /shibboleth-sp>
    Allow from all
  </Location>
  Alias /shibboleth-sp/main.css /opt/shibboleth-sp/share/doc/shibboleth/main.css
  Alias /shibboleth-sp/logo.jpg /opt/shibboleth-sp/share/doc/shibboleth/logo.jpg
</IfModule>

#
# Configure the module for content
```

```
#
# You can now do most of this in shibboleth.xml using the RequestMap
# but you MUST enable AuthType shibboleth for the module to process
# any requests, and there MUST be a require command as well. To
# enable Shibboleth but not specify any session/access requirements
# use "require shibboleth".
#
<Location /secure>
  AuthType shibboleth
  ShibRequireSession On
  require valid-user
</Location>

## Protected by Shibboleth and PERMIS
<Location /permissecure>
  AuthType shibboleth
  PermisAuthorization
  ShibRequireSession On
  ShibUseHeaders On
  require valid-user
</Location>

## Protected by Shibboleth and PERMIS but using Shintau Aggregation
<Location /shintausecure>
  AuthType shibboleth
  PermisAuthorization
  ShibRequireSession On
  ShibUseHeaders On
  # following directive is now also required!
  ShibExportAssertion on
  require valid-user
</Location>
```

The relevant sections of the configuration provided above are marked in bold and a guide to configuring the various parameters is presented below.

System wide Configuration Parameters:
PermisPolicyIdentifier – This is the unique identifier of the policy that should be used to authorise the user it much match an Identifier used when configuring the standalone server.
PermisServerHost – This is the server host on which the Standalone server is running.
PermisServerPort – This is the server port on which the Standalone server is running.
PermisJavaClass – The path location of the SAAM JAR library used to construct and parse the SAML Query/Response messages sent to the standalone server.
PermisLog4jLocation – The path location of the log4j properties file used to configure the Java logging of the SAAM module
PermisDebug – One of three debug parameters (debug,info,off) that are used to configure the level of logging in the Apache C++ SAAM authorisation module

Directory Specific Configuration Parameters:

PermisAuthorization – This specifies the directory location is protected by PERMIS.
ShibUseHeaders – Whilst this us a Shibboleth parameter it must be included to allow the SAAM module to retrieve the attribute values from the authentication request in order to create the authorisation request to the standalone server.

ShibExportAssertion on – Whilst this is a Shibboleth parameter it must be included to allow the standalone server to retrieve the SAML authentication assertion when Shintau is used.

The configuration given above protects three directories; secure which is protected only by Shibboleth authentication, permissecure which when used with the example policy "Shintau.xml" provided in the standalone package requires a permisRole attribute of Role0 or Role1 to access and shintausecure which when used with the example policy "Shintau.xml" provided in the standalone package requires permisRole attributes of Role0 and Role1 to access.

5 A Simple Interoperability Test Scenario

An example policy "Shintau.xml" is included in the standalone release package. This policy protects two directories, permissecure/ and shintausecure/. The permissecure directory requires Role0 or Role1 only and the shintausecure directory requires both Role0 and Role0 to authorise.

If you wish to test this configuration you should create two user's at your IdP the first with a permisRole attribute of Role0 and the second with a permisRole attribute of Role1. Either of these accounts should then be sufficient to authorise you to access permissecure but only when the accounts are linked and a Link Release Policy set by the user should it be possible to access the shintausecure directory.