Division of IT

Shibboleth SP Simple Installation Guide For LINUX

University of Missouri

Revision History

AM	July 2012	Created
AM	July 26, 2012	Changed links to SP download
AM	August 29, 2012	Updated for 2.5 client (no changes)
AM	September 6, 2012	Added special note for those upgrading from a previous release to 2.5. Other miscellaneous changes.
AM	September 7, 2012	Added section on configuring multiple host names on one SP

October 2012

1. Background

1.1.What is a Service Provider?

To put it simply, a service provider is the website you are trying to secure. Users authenticate to your site and you provide a service to them or send them away if they are not authorized to use your site.

For Linux distributions, the service provider software has both a daemon that runs in the background and also a loadable module in your Apache(httpd) server. The two components work together with the Shibboleth IdP servers to authenticate your users and provide your application with information about those users.

For Windows, the service provider has a windows service and an ISAPI module that loads into IIS. Like Linux, on Windows these two components work together with the Shibboleth IdP to authenticate users and provide information to your IIS application.

1.2.How does my application integrate with the Shibboleth SP software?

The Shibboleth SP daemon and Apache or IIS modules provide your application with one or more environment variables that you may use in your code to determine which users are visiting your site. If you've written your application to use REMOTE_USER or a similar variable then you can probably use Shibboleth. There are issues when using REMOTE_USER in IIS, so the Shibboleth authors recommend picking a different variable name for a username, the name can be configured during the SP software setup.

Commercial applications that can use an environment variable for a username can often also be configured to use Shibboleth. At the University of Missouri, Shibboleth has been used to secure commercial applications like Oracle Apex, Blackboard, and the PeopleSoft portal. The key to making this work, is that the application must turn off its default authentication system and rely on authentication services from outside the application (Apache or IIS).

1.3.Can Shibboleth tell me if a visitor is enrolled in a specific course or whether they are enrolled as a full or part-time student?

Shibboleth is best used as an "authentication" service. That is to say it tells your application who a person is (his or her username) and to a limited extent what roles within the institution the person may have, e.g. "staff", "student", etc. If your application needs more explicit information about a visitor, it is best if the application takes the username information and queries other data sources to determine whether that user is authorized to access a particular resource.

1.4.What if my application doesn't run under Apache or IIS?

Most web applications use either Apache, or IIS as the engine that handles requests from users' browsers. Shibboleth doesn't run on anything but Apache or IIS. However, it is still possible to protect web applications that run in a Tomcat, WebSphere, JBoss or other application environment with Shibboleth. You simply have to run Apache (or IIS) as a "reverse proxy" server. This will put Apache "out

in front" of your application server to handle security, then Apache will pass user credentials through to your application. See Appendix A for further discussion of using Apache/Shibboleth as a "reverse proxy." This method of securing applications is very common, but requires a thorough understanding of the underlying application.

2. Installing a Simple Service Provider

This section describes the process of installing a simple Service Provider (SP) that communicates with the University of Missouri System Shibboleth Identity Provider (IdP) from the initial request to successful integration in the production environment.

Prerequisites for Installation:

- A basic understanding of web applications in general as well as your application and in particular how your application authenticates users.
- For Linux SPs: A secure shell (SSH) client, the ability to navigate around a Linux OS, run commands to install software, start and stop daemons, and edit text files using vi or another editor.
- For Windows SPs: A remote desktop client, the ability to navigate directories and use a text editor for windows.
- A way to install the SP software on your host either with root/administrator access or by having someone who can install the software for you.
- Permissions to modify the Shibboleth configuration files, your Apache configuration files or IIS configuration AND the ability to start and stop the Shibboleth daemon/service as well as Apache or IIS.
- You **must** have already configured your website with an SSL certificate and at the very least set up a sample application or directory that you can use to test that Shibboleth is working. SSL certificates that are recognized by nearly every browser are available free of charge for University domains via the Division of IT SSL certificates page: https://doit.missouri.edu/security/certificates/

2.1.Service Provider Request Process

Any University application owner who wishes to integrate with the University of Missouri System, Division of IT Shibboleth Identity Provider should complete the request form which can be found on the IT Security Shibboleth page (http://doit.missouri.edu/security/shibboleth/). The completed form will be automatically submitted to the **UM DoIT Shib Support** distribution list for processing.

2.2.Download Shibboleth SP Software for your OS

Shibboleth software is available in pre-built binaries or source code for Linux and pre-built installers for Windows. **You must install the version appropriate to your operating system and choose the correct architecture (32-bit or 64-bit). Use at least SP version 2.4.3 or higher. Earlier versions have a serious security hole in one of the accompanying software libraries, and are no longer supported by the University.**

For Linux users, if you do not see your OS in the pre-built RPM list, we strongly urge you to change your OS to one supported by the University and Shibboleth software. It is possible to build Shibboleth from source code, but this is not recommended.

For Windows users, your **entire** IIS, Shibboleth and application (.Net etc) application software environment must be the same hardware architecture (32-bit or 64-bit). You cannot mix architectures on Windows. Even though 32-bit applications will usually run on a 64-bit OS, this is NOT the case with Shibboleth. It will fail if you run in a mixed architecture environment; often without anything to indicate why. For more information about Windows SPs, see the Shibboleth SP Simple Installation Guide for Windows IIS.

Shibboleth Service Provider (SP) software can be found here:

http://shibboleth.net/products/download.html

2.3.Install Shibboleth SP Software

For Linux users, either download the latest RPMs (http://shibboleth.net/downloads/service-provider/latest/RPMS/) and use the rpm command to install Shibboleth. Or simply add the appropriate repository to your yum configuration and install with yum. Building from source is not necessary and is discouraged.

The installation software will automatically create several default configuration files and generate a private key and certificate that is used to encrypt/sign data with the Shibboleth IdP. We strongly encourage you to use the default key and certificate. You do **NOT** need to use the same key/cert for Shibboleth as you are using to protect your web site, and doing so will **NOT** improve your site security. Switching to a different key/cert pair will, in fact, make administering your site more complicated over time.

2.4. Special Note When Upgrading Rather than Installing Fresh

Usually, the Shibboleth SP software can be upgraded in place with the latest binary RPMs from the download site. Usually, configuration files do not change from major release to major release. So, upgrading from 2.3.1 to 2.4.3 probably doesn't require any change to the configuration.

However, there is one notable exception to this. The 2.5 release of the Shibboleth SP daemon no longer runs as the root user. If you are upgrading to 2.5 then it is quite likely that your SP's private key and certificate files (sp-key.pem and sp-cert.pem) are still owned by root and have restrictive permissions that don't allow the new non-root Shibboleth user to access them. So, you must change the ownership of these files to the userid that is running the shibd daemon. If you do not change ownership of these files, then the shibd daemon will still start, but the IdP will not accept the SP as a valid client. Basically, the SP won't be sending the correct certificate information because it can't read the key and cert owned by root. Make sure you change ownership, but not the permissions flags, on these two files.

2.5.Configure Service Provider Metadata

Metadata is the data used to configure and describe your Shibboleth SP, and there are seemingly an infinite number of configuration options. However, for most installations only two or three files need to be changed to get Shibboleth up and running.

2.4.1. Modifying shibboleth2.xml

The shibboleth2.xlm file contains the basic Shibboleth SP configuration. It contains a unique identifier for your SP, sets session timeout information and tells the SP software about the IdP (where people sign-in) it will communicate with. The shibboleth2.xml file is located in the main Shibboleth directory: /etc/shibboleth/shibboleth2.xml. Open your shibboleth2.xml file with your favorite text editor and make the following changes:

1) Search for the <ApplicationDefaults tag and set your entityID to the host name URL for your website followed by a "/" and the word "shibboleth". This entityID is completely arbitrary and does NOT point any browser or user to any specific location on your site. You could just as easily name your entityID something like "super.waycool.funzone". However, your machine's URLs are used because they are unique values on the internet and they make it easy for administrators to identify a particular SP configuration. Here is an example valid entityID:

   ```
   entityID="https://grid.umsystem.edu/shibboleth"
   ```

2) In the same <ApplicationDefaults> tag add the value "samaccountname" to the REMOTE_USER attribute. For example:

   ```
   REMOTE_USER="samaccountname eppn persistent-id targeted-id"
   ```

   This will attempt to set the REMOTE_USER web server environment variable to the first available attribute from the list: samaccountname, eppn, persistent-id, and finally targeted-id.

3) Search for the <Sessions tag and make it look like the code below. You can modify the numbers for lifetime and timeout, they are recorded as seconds:

   ```
   <Sessions lifetime="32400" timeout="32400" checkAddress="false"
        consistentAddress="true"
        relayState="ss:mem" handlerSSL="true"
        postData="ss:mem" postTemplate="postTemplate.html"
        cookieProps="; path=/; secure">
   ```

   The settings shown above will set your Shibboleth session lifetime to 9 hours (the maximum allowed by the University's IdP). It also sets the timeout, which is the maximum time allowed between visits to your site for a session to remain active, to 9 hours. Web form POST data will be saved in the Shibboleth memory cache rather than discarded when a user requires authentication after filling out a web form. Finally, it sets the Shibboleth cookie properties so that the cookie applies to the entire site. The cookie is only sent when the browser connects with https (SSL encryption), and tells the browser to delete the Shibboleth cookie when the browser is closed.

4) Search for the first <SSO tag and set it as follows:

```
<SSO entityID="https://shib-idp.umsystem.edu/idp/shibboleth"
     discoveryProtocol="SAMLDS"
     discoveryURL="https://shib-idp.umsystem.edu/DS/WAYF">
     SAML2 SAML1
</SSO>
```

The settings above point your SP to the University's only IdP for single-sign-on and "discovery". We don't actually use discovery, but if we did this is where your SP would send users.

5) Search for <Handler type="Status" and add your desktop workstation's IP address to the ACL list. This will allow you to view the Shibboleth "status" URL from your desktop. Add other IP addresses as needed (if you have monitoring software that can monitor a site URL, etc). Do not attempt to add whole subnets or open the status to the world. Your status handler line should look something like this:

```
<Handler type="Status" Location="/Status"
     acl="127.0.0.1 128.206.92.200"/>
```

6) Search for <Handler type="Session" and change the showAttributeValues to true. Later you can change this back to false, but for setup and debugging, it is really helpful to have Shibboleth be able to display attributes associated with logged in users. Your session handler should look like this:

```
<Handler type="Session" Location="/Session"
     showAttributeValues="true"/>
```

7) Search for <Errors supportContact and adjust the support contact email address and logo's to fit your site "look and feel". If you don't, your users will see the standard Shibboleth "griffin" when a Shibboleth error occurs. Adjust your settings to point at your version of these files, perhaps something like this:

```
<Errors supportContact="oscarthegrouch@missouri.edu"
     logoLocation="/images/logo.jpg"
     styleSheet="/css/main.css"/>
```

8) Search for <MetadataProvider and configure the IdP metadata as follows:

```
<MetadataProvider type="XML"
 uri="https://shib-idp.umsystem.edu/idp/profile/Metadata/SAML"
 backingFilePath="shib-idp.umsystem.edu.metadata.xml"
 reloadInterval="7200"/>
```

The settings above tell your SP to retrieve the IdP metadata dynamically every 2 hours from the IdP itself and to save it in a temporary file called shib-idp.umsystem.edu.metadata.xml. This is the **best way** to keep your IdP metadata file up to date. It is also possible to retrieve the

file manually and place a static copy of the IdP metadata on your server, but this is discouraged.

9) Save your changes.

2.4.2. Configure the attribute-map.xml file

The attribute-map.xml file sets up the user attributes that your site accepts from the IdP. For the purposes of this document, we'll only request the user's logon userID or samAccountName as it is called in ActiveDirectory. Make the changes shown below to your attribute-map.xml file:

1) At the top of the file insert the following:

```
<Attribute name="urn:mace:umsystem.edu:attribute-def:samAccountName"
    id="samaccountname">
   <AttributeDecoder xsi:type="StringAttributeDecoder"
    caseSensitive="false"/>
</Attribute>
```

This will allow the SP to receive the samAccountName attribute if it is sent by the IdP. Recall that we configured the samaccountname in the shibboleth2.xml file in section 2.4.1 above. So if we receive samaccountname from the IdP, the SP software will attempt to set the REMOTE_USER variable to this value. Notice the difference in capitalization between the two instances of samAccountName and samaccountname above. Be very careful with capitalization it can sometimes bite you even when caseSensitive = false.

2) You may uncomment other attributes, but doing so DOES NOT mean that you will automatically receive them from the IdP, nor does it tell the IdP that you wish to receive them. The IdP configuration is solely responsible for the attributes sent to your SP and you **must** tell the Shibboleth support team which attributes you need. The attribute-map.xml file only tells your Shibboleth SP which attributes, of those it does receive, are to be passed on to applications.
3) Save your changes.

2.4.3. Test the SP Configuration

After the changes above, you should test your SP configuration. Run "shibd -t" (find the binary and set a path in your environment if necessary). You should see the words "Overall configuration is loadable". If you don't see this, and instead see some other error, then most likely you didn't close an XML tag or quotes in the configuration changes you made above. Shibd will start without error using the distributed copies of the xml configuration files. If it doesn't start after you've made changes, it is most likely due to one of those changes. Set a DEBUG level in the shibd.logger configuration file and test again until your configuration loads.

2.4.4. Generate your SP's Metadata File

The entire configuration up to this point has been necessary to make the Shibboleth SP daemon/service run on your server. Now information must be exchanged with the IdP so the two Shibboleth services can communicate. To do this, the IdP needs a copy of your "metadata". There are two methods to do this.

The first way is to make your metadata available on a URL that the IdP can download. The second (preferred) method is to create your own copy of the metadata file and e-mail it to shibsupport@umsystem.edu. The URL method seems convenient because you can make changes to your metadata and have it be automatically updated on the IdP. Unfortunately, if your changes cause your metadata to break then the IdP will simply not load the changed file. You may not know you've got an error, but your changes won't work either. It's best to coordinate changes with the Shib support team by sending them your metadata in email.

Either way, it is important to understand that your SP software does not use or reference your generated metadata file at all. If you save a copy of your metadata in the Shibboleth configuration folder, it is just an extra file. The SP gets its configuration from the files mentioned above, NOT the metadata file you save to disk or send to the Shibboleth team. The number one cause of mistakes in a Shibboleth SP configuration is for someone to change their generated metadata file, but not their actual configurations in shibboleth2.xml or attribute-map.xml.

To generate your SP's default metadata file (based on the configuration changes you've already made), do the following:

1. Start, or restart, the Shibboleth SP daemon/service. On Linux you should use the service command to start and stop shibd. Note: if you do NOT use the service command, it is very likely that the Shibboleth SP daemon will **not** release its open sockets and you will not be able to restart shibd after attempting to shut it down. The simplest way to clear "stuck" sockets is to reboot. Use the service command as a root or sudo user like this:

```
# service shibd stop
# service shibd start
```

2. After restarting the shibd daemon, download your metadata from the server's web site using a command like curl on Linux (the command below goes all on one line):

```
curl -o mysp.umsystem.edu-metadata.xml
-k https://mysp.umsystem.edu/Shibboleth.sso/Metadata
```

   The -o option specifies the output file name and -k tells curl to ignore any SSL errors.

3. The standard for naming metadata files is "hostname.domain.edu-metadata.xml." Rename your file to the fully qualified host name of your machine; even better would be to use the host name from the "entityID" section of your shibboleth2.xml file.

If you make configuration changes to your entity id or contact data in shibboleth2.xml or change your key or certificates, you may have to regenerate your metadata file and send it to the Shibboleth support team.

2.4.5. Modifying the Metadata

Unfortunately, there are some configuration options that simply cannot be set in the shibboleth2.xml file, and it is sometimes necessary to modify your metadata file directly. If, for example, you want to add

extra host names for your site or you want to customize the login message the user sees on the IdP then you must MANUALLY modify your metadata file. Remember, modifying the metadata file **will not** change how your SP functions, all of that is controlled by the shibboleth2.xml and attribute-map.xml files.

A common customization is to change the login logo and message users see when your site sends them to the IdP to authenticate. To do this you must manually update your metadata file before sending it to the Shibboleth support team. To do this:

1. Open your metadata file with a text editor. Search for <md:EntityDescriptor. It should be the very first line in the file.
2. In this EntityDescriptor tag you'll see an attribute like: xmlns:md="urn.....". Leave it there but move your cursor to a position after this attribute and paste the following additional xmlns attribute:

```
xmlns:mdui="urn:oasis:names:tc:SAML:metadata:ui"
```

3. You should now have two "xmlns:" attributes. All this did was add another shortcut description for the XML tags later in the file. When you add your logo and description information to the file, you will use this "mdui" shortcut, otherwise you would have to preface each tag with "urn:oasis:names:tc:SAML:metadata:ui".
4. Search for the <md:Extensions> tag. It is probably the third line in your metadata file. Add the following tags after the <md:Extensions> tag, but before the <DiscoveryResponse... tag.

```
<mdui:UIInfo>
     <mdui:DisplayName xml:lang="en">My Test Site</mdui:DisplayName>
     <mdui:Description xml:lang="en">A test site.</mdui:Description>
<mdui:Logo height="146"
width="148">https://mysp.umsystem.edu/icons/logo.png</mdui:Logo>
</mdui:UIInfo>
```

   Set the displayname, description and logo as appropriate for your site. Each of the tags is optional. If you specify a logo then, **your logo MUST be between 64 to 350 pixels wide and 64 to 146 pixels tall. More importantly, your logo must reside on an SSL encrypted site that is NOT protected by Shibboleth.** If you place your logo on an http (unencrypted) site, then your users' browsers will likely display a warning that the login page contains unencrypted and encrypted data. **This is not acceptable and will not be allowed in production because it undermines the perceived security on the login screen.** Your logo will not be placed on the Shibboleth IdP server; it is up to you to securely serve up this file to users.

2.4.6. Submit Metadata to ISAM

The metadata file must now be submitted to the Shibboleth support team.

1. If you are using the URL method to get your metadata to the Shibboleth support team, then place the metadata file in a location that your website can serve up to the IdP servers and open any necessary firewall holes so that those servers can reach your site. The production IdP servers sit behind a load balancer and have their own distinct IP addresses that MUST be put in your firewall "allow" list. Contact the Shibboleth support team for those addresses. Send the URL to the shibsupport@umsystem.edu email address.
   If you are mailing the file to the Shibboleth support team, then transfer the file to your mail client and attach it to an email and send it to shibsupport@umsystem.edu. Because Exchange/Webmail sometimes blocks attachments with .xml extensions, add a .txt extension to the metadata file just to make sure it gets through.
2. The Shibboleth support team will validate your metadata file for correct syntax and then upload it to the University's Shibboleth Quality Assurance server. **You will be informed when this is complete.** You **MUST** validate that your site is working against this QA server before the configuration will be migrated to production.

2.4.7. Test that the SP is Working

Your SP should now be ready to test in the QA environment. To do this you must configure a Shibboleth protected directory or location on your web server. By default, the Shibboleth SP software will create a shib.conf file in your Apache directory that will protect a location on your server. The location isn't necessarily a real directory, but rather a URL location. By default the location is: https://[yourhostname]/secure. It does not matter whether the "/secure" URL points to a real directory or not, Apache doesn't care and will still protect the URL. However, it is less confusing if you create a directory called "secure" in your web server's document root.

1. Navigate to your web servers document root directory. On RedHat Linux this is usually /var/www/html.
2. Create a directory named "secure".
3. Create a simple index.html file in the directory. Perhaps something like:

```
<html><body>Hello World</body></html>
```

4. The shib.conf (typically located in /etc/httpd/conf.d/) file should already contain any other settings necessary to load the shibd_mod into Apache.
5. Restart Apache by running the following as root or with sudo:
   # service httpd stop
   # service httpd start
6. Check the Apache logs for errors and correct them before moving on to step 7. On RedHat, Apache errors are usually located in /var/log/httpd and the Shibboleth daemon logs its errors to /var/log/shibboleth.
7. Assuming you have a Windows workstation, you must now edit your "hosts" file so that your workstation uses the University's Shib QA IdP server rather than the production server. Your SP will attempt to send you to production because that is what is configured in your shibboleth2.xml file, but the production Shib servers do not yet know anything about your

SP. The change to your hosts file will cause your browser to think it is talking to the production servers when it is, in fact, talking to the QA servers.

8. Open a command prompt on your workstation using "Run as Administrator".
9. Change directories to c:\windows\system32\drivers\etc
10. Edit the hosts file and add an entry for the Shib QA server IP address with the production name:

```
128.206.14.152   shib-idp.umsystem.edu shib-idp-qa.umsystem.edu
```

    The settings above will trick your browser into going to the QA server (128.206.14.152) whenever the host name of shib-idp.umsystem.edu is used. You do **not** have to change anything on your SP. Save the file. When you are finished testing simply comment out the line above in your hosts file by putting a "#" at the beginning of the line. Save the file.
11. Close all browser windows. This is necessary because your browser may have cached the host name/ip address for the production Shibboleth servers. Closing the browser down clears this cache.
12. Open your browser and point at your site's "secure" directory. For example:

```
https://mysite.umsystem.edu/secure
```

13. You should be immediately directed to the Shibboleth (QA) IdP. Check your Apache logs (/var/log/httpd) and Shibboleth logs in /var/log/shibboleth/ for errors if you are not directed to the IdP. Correct any errors, restart both Apache and shibd services, close your browser and try again.
14. On the QA IdP, sign in using your University ID and password.
15. You should be directed back to your SP's "secure" directory (assuming you've set one up in step 2 above).
16. To verify that the IdP is sending the correct user attributes to your site you can check the "session" debug URL on your SP. This URL is automatically generated by the Shib daemon on the SP. It is:

    ```
    https://[yourservername]/Shibboleth.sso/Session
    ```

    If you see:

    ```
    A valid session was not found.
    ```

    Then a Shibboleth credential wasn't created for your site. Check your Apache and Shibboleth logs for errors.

    Everything is working if you see something like this:

    ```
    Miscellaneous
    Client Address: 128.206.92.200
    Identity Provider: https://shib-idp.umsystem.edu/idp/shibboleth
    ```

```
SSO Protocol: urn:oasis:names:tc:SAML:2.0:protocol
Authentication Time: 2012-04-03T15:30:09.252Z
Authentication Context Class:
urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
Authentication Context Decl: (none)
Session Expiration (barring inactivity): 539 minute(s)

Attributes
SAM_ACCOUNT_NAME: 1 value(s)
eppn: 1 value(s)
persistent-id: 1 value(s)
```

Note that if you previously set 'showAttributeValues="true"' in your shibboleth2.xml file then you'll see the actual user attribute values, such as your username, passed to your SP rather than a message indicating the values were there.

17. After you finish testing, change your "hosts" file back to normal and let the Shibboleth support team know that everything is working (or not).

2.4.8. Protect Other Resources with Shibboleth

After you have the "secure" location working, you can now expand your Apache/Shibboleth configuration to include other URL "locations" on your web site. To do this you simply add or change the configuration of your Apache server by modifying either the /etc/httpd/conf/httpd.conf or /etc/httpd/conf.d/shib.conf files. Typically, you would add these changes to the httpd.conf file and leave the shib.conf file for settings specific to the Shib SP daemon, but you can make the changes in either location. It is also possible to use .htaccess files to protect physical directories, but they cannot protect locations in URLs so for consistency's sake, using .htaccess files is not recommended.

For example, if you want to protect your /cgi-bin directory, then add the following:

```
<Location /cgi-bin>
   AuthType shibboleth
   ShibRequireSession On
   ShibRequestSetting requireSession 1
   ShibUseHeaders On
   require valid-user
</Location>
```

A note about protecting your ENTIRE web site: While it sounds like a good idea to simply protect your entire site, this doesn't often work out as planned. There are certain locations on your site that must be available to users who have not authenticated. One of them is /Shibboleth.sso. This location and everything under it is used by Shibboleth to handle the authentication process. Also, exclude any area of your site that might contain error pages or images you wish serve up to users who can't or haven't yet authenticated (don't forget to allow access to your site logo displayed on the IdP). Exclude these locations from your Apache configuration, or configure them to be explicitly open. Typically, Apache reads configurations from top-to-bottom and general-to-specific so if you specify your whole site "/" as

protected by Shibboleth, you would later in the same configuration file also specify the specific resources that are not protected. Like so:

```
<Location /images>
AuthType none
ShibRequireSession Off
</Location>
```

No changes are required to either your SP metadata or the IdP configuration for you to protect additional URL locations on your web server.

2.4.9. Protecting Additional Host Names

Sometimes a web server has multiple host names and runs multiple applications. This is usually accomplished by setting up VirtualHosts in the Apache configuration (see Apache.org for documentation on setting up multiple VirtualHosts). If your SP has multiple host names, it is possible to use Shibboleth to protect one or more of them at the same time. Protecting additional host names is easiest if all those hosts use the same user attributes from the IdP.

Configuring multiple applications that require different set of user attributes is beyond the scope of this "simple" install guide. Please consult the ApplicationOverride section of the Shibboleth wiki (https://wiki.shibboleth.net/confluence/display/SHIB2/NativeSPApplicationOverride) for more information on this complex process.

To configure the Shibboleth SP to protect multiple host names when they all receive the same set of attributes you must do two things.

The first step is to configure Apache and its SSL certificate to handle multiple host names. Modern browsers accept SSL certificates with alternative host names embedded in them. So, one certificate and IP address can be used with multiple host names. Multi-host name certificates are available through the Division of IT SSL certificate request process. Alternatively, you can configure Apache with multiple IP addresses, each with their own host name and certificate. See Apache.org for more information about configuring multiple hosts and certificates.

The second step is to modify your generated metadata file manually. Unfortunately it is not possible to set up multiple-host protection by modifying the shibboleth2.xml file.

At the bottom of the metadata file you saved from previous steps, there should be a block of multiple lines that start with "<md:AssertionConsumerService". Here is an example:

```
<md:AssertionConsumerService
Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
Location="https://mysp.umsystem.edu/Shibboleth.sso/SAML2/POST" index="1"/>
<md:AssertionConsumerService
Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST-SimpleSign"
Location="https://altid.umsystem.edu/Shibboleth.sso/SAML2/POST-SimpleSign"
index="2"/>
```

```
<md:AssertionConsumerService
Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Artifact"
Location="https://mysp.umsystem.edu/Shibboleth.sso/SAML2/Artifact"
index="3"/>
<md:AssertionConsumerService
Binding="urn:oasis:names:tc:SAML:2.0:bindings:PAOS"
Location="https://mysp.umsystem.edu/Shibboleth.sso/SAML2/ECP" index="4"/>
<md:AssertionConsumerService
Binding="urn:oasis:names:tc:SAML:1.0:profiles:browser-post"
Location="https://mysp.umsystem.edu/Shibboleth.sso/SAML/POST" index="5"/>
<md:AssertionConsumerService
Binding="urn:oasis:names:tc:SAML:1.0:profiles:artifact-01"
Location="https://mysp.umsystem.edu/Shibboleth.sso/SAML/Artifact"
index="6"/>
```

To add an additional host name simply copy this entire set of lines and create a new set below the first set. Then edit the "Location=" attribute and the "index=" attribute in the new set of lines. Change each host name in the location URL to the additional host name you want to use, and increment the index value on each new line so that it doesn't conflict with the previous set. Leave the blocks of AssertionConsumerService with similar host names together, do not attempt to create a set of "HTTP-POST" lines with different host names followed by a set of "HTTP-POST-SimpleSign" with different host names, etc.; breaking up the blocks and intermixing different host names will cause the Shibboleth SP daemon to fail even though it is still correct XML syntax.

Below is an example block of metadata configured to handle the host names mysp.umsystem.edu and mysp.missouri.edu:

```
<!-- UM System -->
<md:AssertionConsumerService
Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
Location="https://mysp.umsystem.edu/Shibboleth.sso/SAML2/POST" index="1"/>
<md:AssertionConsumerService
Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST-SimpleSign"
Location="https://mysp.umsystem.edu/Shibboleth.sso/SAML2/POST-SimpleSign"
index="2"/>
<md:AssertionConsumerService
Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Artifact"
Location="https://mysp.umsystem.edu/Shibboleth.sso/SAML2/Artifact"
index="3"/>
<md:AssertionConsumerService
Binding="urn:oasis:names:tc:SAML:2.0:bindings:PAOS"
Location="https://mysp.umsystem.edu/Shibboleth.sso/SAML2/ECP" index="4"/>
<md:AssertionConsumerService
Binding="urn:oasis:names:tc:SAML:1.0:profiles:browser-post"
Location="https://mysp.umsystem.edu/Shibboleth.sso/SAML/POST" index="5"/>
<md:AssertionConsumerService
Binding="urn:oasis:names:tc:SAML:1.0:profiles:artifact-01"
```

```
Location="https://mysp.umsystem.edu/Shibboleth.sso/SAML/Artifact"
index="6"/>

<!-- MU -->
<md:AssertionConsumerService
Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
Location="https://mysp.missouri.edu/Shibboleth.sso/SAML2/POST" index="7"/>
<md:AssertionConsumerService
Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST-SimpleSign"
Location="https://mysp.missouri.edu/Shibboleth.sso/SAML2/POST-SimpleSign"
index="8"/>
<md:AssertionConsumerService
Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Artifact"
Location="https://mysp.missouri.edu/Shibboleth.sso/SAML2/Artifact"
index="9"/>
<md:AssertionConsumerService
Binding="urn:oasis:names:tc:SAML:2.0:bindings:PAOS"
Location="https://mysp.missouri.edu/Shibboleth.sso/SAML2/ECP" index="10"/>
<md:AssertionConsumerService
Binding="urn:oasis:names:tc:SAML:1.0:profiles:browser-post"
Location="https://mysp.missouri.edu/Shibboleth.sso/SAML/POST" index="11"/>
<md:AssertionConsumerService
Binding="urn:oasis:names:tc:SAML:1.0:profiles:artifact-01"
Location="https://mysp.missouri.edu/Shibboleth.sso/SAML/Artifact"
index="12"/>
```

Notice that the indexes range from 1-6 and 7-12 and that the host names are in different domains. You can just as easily have host names like application1.umsystem.edu and application2.umsystem.edu.

After changing your metadata, send a new copy to the Shibboleth support team so you can test the changes on the QA system. You don't need to change anything in your shibboleth2.xml file and Apache (if configured correctly) will handle sending the different host names to the IdP.

Depending on your Apache VirtuaLHost settings, you may need to specify Shibboleth protection directives for locations and directories under each <VirtualHost></VirtualHost> configuration block in httpd.conf. If you simply use multiple host names, but have a universal <VirtualHost _default_:443> block, then the Shibboleth protection will apply to all the "locations" regardless of host name used by the client. See section 2.4.8 for an example of protecting a specific location.

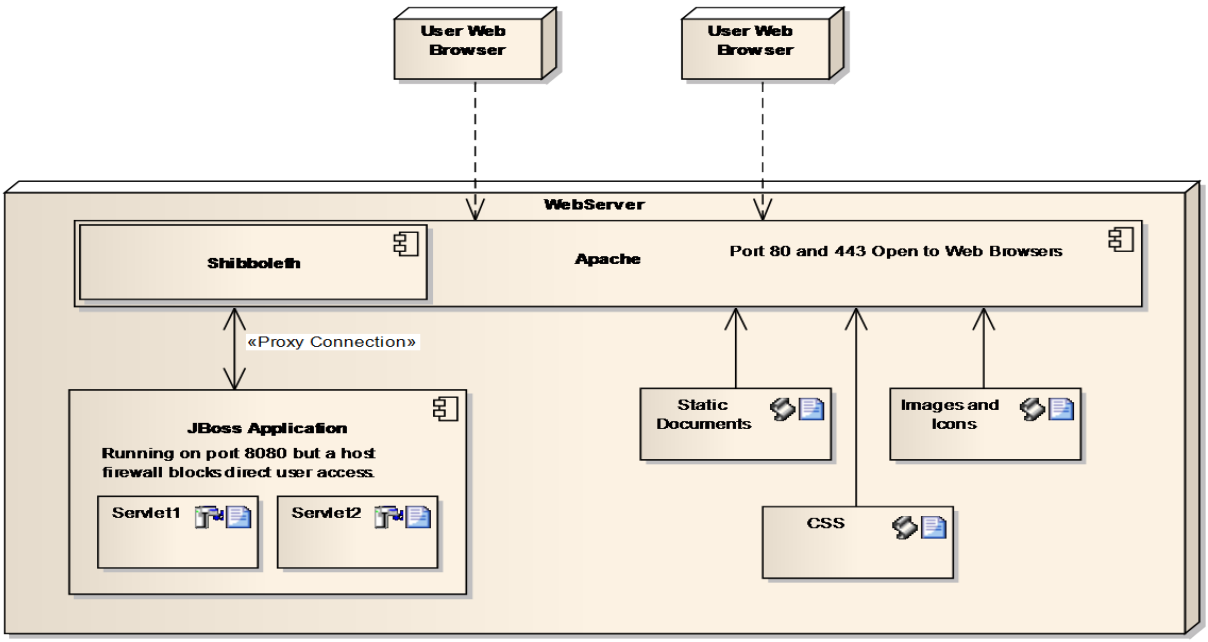
Appendix A: Setting up a Reverse Proxy to Protect non-Apache Applications

As mentioned in the first part of this document, not all web applications run as part of an Apache install, but Shibboleth integrates ONLY with Apache on Linux operating systems. Despite this, it is still possible to secure non-Apache web applications with Apache and Shibboleth using a "reverse proxy" technique. This technique will **not** work with non-web applications, but it will work with many applications that communicate using HTTP or HTTPS protocols.

This technique is used not only at the University of Missouri but at institutions all over the world to add Shibboleth protection to applications that do not support it natively. For this method to work, your application MUST be able to read header or environment variables, also known as CGI variables.

Using a reverse proxy is simple: First, you configure your non-Apache web application to run on a non-standard port like 8080 (HTTP) or 8443 (HTTPS). Then you configure Apache to run on the standard ports of 80 (HTTP) and 443 (HTTPS) in proxy mode. When Apache receives a request from a user's web browser, it checks to see if the user is authenticated via Shibboleth, and if so, it passes the request on to the application running on 8080/8443. An authentication "header" is added to this request to tell your application about the user's credentials.

Typically, Apache, Shibboleth and your application are all installed on the same machine and both the user and application are completely unaware that the Apache/Shibboleth combination is running in the middle.



In the example configuration shown below the server is running Apache/Shibboleth on the standard HTTP and HTTPS ports of 80 and 443. It is also running a copy of Oracle's HTTP server that front-ends Oracle's Apex product on port 7779, and yet another Oracle HTTP server that runs Enterprise Manager GridControl on port 1159.

The Apex service cannot accept inbound connections directly from off the host because of host based firewall rules, so there is no danger of users accessing its URLs directly. HOWEVER if the firewall rules were turned off, users could access these apps by using the following URLs:

- https://grid.umsystem.edu:1159/em - This is Enterprise Manager application
- http://grid.umsystem.edu:7779/pls/f?p=101:1 - This is the Apex application URL

Notice that the host name is the same, but the port numbers are different for each of the two services. The /em and /pls parts of the URLs are called "locations". This is an important concept. These "locations" do not represent directories on disk. They instead route requests to programs running as modules in the Oracle HTTP servers. This is the same as the /Shibboleth.sso/Session locations built-in to the Apache/Shibboleth server. Anything referencing /Shibboleth.sso/* is fed to the Shibboleth module for it to handle rather than being served up by Apache out of the file system. The same is true of /em and /pls for Apex and Enterprise Manager.

The example configuration below is from Apache running on the standard HTTP/HTTPS ports. Most of the non-proxy-specific pieces of the standard http.conf are left out for the sake of brevity. Comments within the configuration are used to explain what is happening at each point. Order of the different configuration pieces is important and all of these settings are from the bottom of the server's default httpd.conf file.

```
# Add settings to configure Apache to proxy the other
# daemons running on this server.
# It looks as if we are turning off the Apache proxy here,
# but in fact this statement disables FORWARD proxying
# and not the reverse proxying we will be doing. THIS
# STATEMENT IS CRITICAL. It prevents hackers from using
# your server to attack other sites, making the attack
# look like it is coming from you.
ProxyRequests Off

# This location configuration is using Shibboleth to protect any URL
# starting with /i. The /i location is an alias for a real directory
# located on disk that is used by the Apex software to serve up
# static images, javascript, and CSS files.
<Location /i>
   AuthType shibboleth
   ShibRequireSession On
   require valid-user
</Location>
```

```
# This location configuration is using Shibboleth to protect any URL
# starting with /pls. The /pls location is not a real directory, but
# instead points at the software that connects Oracle's HTTP server
# to its built-in handler for the Apex application.
# Notice the addition of the "ShibUseHeaders On" directive. This
# will cause Apache to set request headers in the proxy requests
# and pass them to the server being proxied. The headers will
# carry the user's identity attributes to the underlying application.
<Location /pls/>
   AuthType shibboleth
   ShibRequireSession On
   require valid-user
   ShibUseHeaders On
</Location>

# This location configuration is using Shibboleth to protect any URL
# starting with /cgi-bin. The /cgi-bin location is a real directory
# containing perl scripts to help debug user environment variables.
# The directory itself will be proxied (see settings below) and is
# served up by the Oracle HTTP/Apex server and not the outward facing
# Apache/Shibboleth server.
<Location /cgi-bin/>
   AuthType shibboleth
   ShibRequireSession On
   require valid-user
   ShibUseHeaders On
</Location>

# This directive tells Apache/Shibboleth that /i/ is an alias
# that points to a directory located in the Oracle Apex software
# installation.
# By serving up the static Apex images and scripts directly
# from the standard Apache server, we save a round trip to and
# from the proxied Oracle/Apex server. This is more efficient.
Alias /i/ "/u01/app/oracle/product/10gR2HTMLDB/htmldb/images/"
# This configuration merely tells the Apache/Shibboleth server
# how to handle files in the images directory.
<Directory "/u01/app/oracle/product/10gR2HTMLDB/htmldb/images">
    Options Indexes MultiViews
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>

# This location configuration is using Shibboleth to protect any URL
# starting with /em. The /em location is not a real directory, but
# instead points at the software that connects Oracle's HTTP server
# to its built-in handler for the Enterprise Manager GridControl
# application.
```

```
# Notice that in this case we are not using the "ShibUseHeaders On"
# directive. This is because the Enterprise Manager application
# cannot be fully integrated with Shibboleth because there is no way
# to change the EM application to use external authentication.
# Despite this, the site owners still wanted an extra layer of
# authentication surrounding the Enterprise manager application.
# So in this case, users have to login TWICE, once to
# Shibboleth, and once to the EM application with a different
# ID/password pair.
<Location /em>
   AuthType shibboleth
   ShibRequireSession On
   require valid-user
</Location>


# This part of the configuration sets a default to proxy ALL
# requests regardless of URL.
<Proxy *>
Order deny,allow
Allow from all
</Proxy>

# The rewrite directives below tell Apache to rewrite all
# incoming requests that are not connecting to HTTPS port 443 to
# switch to port 443. This causes all web applications to communicate
# using only HTTPS on port 443. For example, if a user accidentally
# types http://grid.umsystem.edu instead of https://grid.umsystem.edu
# then their browser will be automatically told to replay the request
# but to do so on the https (SSL encrypted) port.
RewriteEngine On
RewriteCond %{SERVER_PORT} !^443$
RewriteRule "^/(.*)" "https://grid.umsystem.edu/$1"

# Above we set a default to proxy EVERYTHING but we don't actually
# want to do that. We cannot proxy the Shibboleth interface because
# it is controlling authentication. We also don't want to proxy
# the /secure/ directory as it is our test directory to make sure
# Shibboleth is working. Also, we set up an alias for the Apex
# images directory (/i/) and we want Apache to serve those up
# directly rather than proxying them. So below we must set several
# locations that are NOT to be proxied.
# Don't proxy the Shibboleth interface
ProxyPass /Shibboleth.sso/ !
# Don't proxy the /secure directory
ProxyPass /secure/ !
# Don't proxy the /i/ directory. It's just images.
# Let us serve it up with Apache instead of Apex
ProxyPass /i/ !
```

```
# This set of proxy directives does all the work for the /pls
# location. Anything that comes in to the Apache/Shibboleth server
# using the /pls URL, will be passed on to the Oracle HTTP/Apex
# server running on port 7779. Notice also that this server is
# NOT running with https (SSL Encryption) so the communication
# is in "clear text". In this case the Oracle HTTP/Apex server
# has been bound only to the internal IP address 127.0.0.1, it
# cannot be directly reached from off this host. The entire
# conversation communication between the outward facing
# Apache/Shibboelth server and the Oracle/Apex server
# happens in memory, and cannot be intercepted except by someone
# with privileged access on the server itself.
ProxyPass /pls http://127.0.0.1:7779/pls
ProxyPassReverse /pls http://127.0.0.1:7779/pls

# This set of proxy directives does all the work for the /cgi-bin
# location. Anything that comes in to the Apache/Shibboleth
# using the /cgi-bin URL, will be passed on to the Oracle HTTP/Apex
# server running on port 7779. As above, communication is in
# "clear text". This cgi-bin directory inside the Oracle/Apex
# installation contains a perl script that can be used to display
# all of a users session environment variables. This includes the
# authentication headers set by the proxy server.
ProxyPass /cgi-bin http://127.0.0.1:7779/cgi-bin
ProxyPassReverse /cgi-bin http://127.0.0.1:7779/cgi-bin

# This set of proxy directives does all the work for the
# Enterprise Manager /em location. Anything that comes in to
# the Apache/Shibboleth server using the /em URL, will be passed on
# to the Oracle HTTP Enterprise Manager server running on port 1159.
# Notice that this configuration is different from the previous
# proxy directives. In this case the Enterprise Manager server
# is running with SSL enabled. And because of this the
# Apache/Shibboleth server must be configured to recognize the
# enterprise manager SSL certificates and to use SSL in the proxy
# communications. Why was this done? This was done because
# Enterprise Manager has two components. One piece is for database
# administrators to view and manage their running servers. The other
# piece is used by the servers running a client to report their status
# to the EM software. The servers do this by connecting directly on
# the default port, 1159. Because of this, the EM port of 1159 is
# NOT being blocked by the host firewall. The servers reporting
# through EM can connect directly to the EM server on port 1159
# without needing the Apache/Shibboleth in between. This effectively
# by-passes the proxy and Shibboleth entirely, and is also
# why we are not using 127.0.0.1 as the destination for our
# proxy requests. BE VERY CAREFUL WHEN ALLOWING APPLICATIONS TO
# BY-PASS YOUR APACHE/SHIBBOLETH PROXY!!! THIS IS NOT NORMALLY NEEDED!
```

```
# Turn on the SSL proxy engine
SSLProxyEngine On
# Tell the SSL Engine where to look for the certificates used by
# the Enterprise manager software.
SSLProxyCACertificateFile /etc/httpd/conf/ssl.crt/ca-bundle.crt
# Pass all requests for /em to the EM server running on port 1159.
ProxyPass /em https://gridcontrol.umsystem.edu:1159/em
ProxyPassReverse /em https://gridcontrol.umsystem.edu:1159/em

# -- end of config --
```

The configuration above uses Apache/Shibboleth to protect all of the HTTP port 80 and HTTPS port 443 requests made to the grid server. In combination with local firewall rules, it prevents users from accessing the Apex application running on 7779 directly, but also allows some clients to by-pass Shibboleth and access the Enterprise Manager application directly on port 1159. Local firewall rules are used to limit the access to the Enterprise Manager ports to specific servers.

The Apex application configured above receives user authentication information via HTTP headers passed by the Apache/Shibboleth proxy. When configuring a proxy server, it is absolutely critical that there be some component or script available, during debugging, so that you can dump the session environment to a log or directly back to the user's browser. Without this, it is very hard to verify the names of the Shibboleth authentication headers passed to the proxied applications.

Below is perl code for a script named nph-printenv.pl that can be used to echo the user's headers (behind the proxy) so that they can be seen. Use this ONLY for debugging purposes and you should disable the script after you've finished. In the examples above the script sits in the Oracle HTTP/Apex cgi-bin directory (not the Apache/Shibboleth cgi-bin) and is named nph-printenv.pl.

```
#!/usr/bin/perl
##
##  printenv -- demo CGI program which just prints its environment
##

print "HTTP 1.0/200 OK\n";
print "Content-type: text/plain\n\n";
foreach $var (sort(keys(%ENV))) {
    $val = $ENV{$var};
    $val =~ s|\n|\\n|g;
    $val =~ s|"|\\"|g;
    print "${var}=\"${val}\"\n";
}
```

Your application may have a different mechanism to help with debugging. Use whatever is available. Knowing what headers are being set by Shibboleth and the proxy server is CRITICAL to a successful install.