**Using IBM HTTP Server with SSL to a Liberty server**

Contents

[Lab 1 Using IBM HTTP Server with SSL to a Liberty server 3](#_Toc462388992)

[1.1 Prerequisites 3](#_Toc462388993)

[1.2 Create the server SSLServer 4](#_Toc462388994)

[1.3 Deploy the SimpleSSL sample 4](#_Toc462388995)

[1.4 Configure Liberty to use HTTPS/SSL 6](#_Toc462388996)

[1.5 Using keytool to display content of keystore 8](#_Toc462388997)

[1.6 Install and Configure IHS and the WebSphere plug-in 8](#_Toc462388998)

[1.7 Configure IHS to use HTTPS/SSL 10](#_Toc462388999)

[1.8 Configure the plug-in between httpd and Liberty 12](#_Toc462389000)

[1.9 About plugin-cfg.xml 13](#_Toc462389001)

[1.10 Testing the plug-in 14](#_Toc462389002)

[1.11 Configure SSL between the Plug-in and Liberty 15](#_Toc462389003)

[1.12 Cleanup 18](#_Toc462389004)

[Appendix A. Configuration files, key files, and log files 19](#_Toc462389005)

[Appendix B. Notices 20](#_Toc462389006)

[Appendix C. Trademarks and copyrights 22](#_Toc462389007)

# Using IBM HTTP Server with SSL to a Liberty server

In this lab we will configure IBM HTTP Server (IHS, a web server based on Apache) to act as a front-end web server for a Liberty server. We will configure Liberty, IHS, and the plug-in that connects them to use SSL, so the entire communications channel from browser to IHS to plug-in to Liberty is secure. The steps will be as follows:

1. Deploy a simple application that displays protocol and port information
2. Configure Liberty to use HTTPS / SSL as well as HTTP, and test via the JSP
3. Install and configure IHS and WebSphere plug-in
4. Configure IHS to use HTTPS / SSL
5. Configure the web server plug-in between IHS and Liberty, and test via the JSP
6. Configure SSL between the plug-in and Liberty, and test via the JSP

Please refer to the following table for file and resource location references on different operating systems.

This lab runs only on the Windows and Linux platforms as the WebSphere plugin does not work on Mac OSX.

|  |  |  |
| --- | --- | --- |
| Location Ref. | OS | Absolute Path |
| *{LAB\_HOME}* | Windows | C:/*WLP\_<version>* |
| Linux | ~/*WLP\_<version>* |
| Mac OSX | not applicable |

## Prerequisites

The following preparation must be completed prior to beginning this lab:

1. Complete the Getting Started lab to set up the lab environment, including JRE, and Liberty runtime. You do not need to install the WDT developer tools (steps 0.5 and 0.6 in the Setup lab).

2. Note that this lab runs the IHS web server on ports 9180 and 8443. To check whether these ports are available on your system, you can use the netstat command as follows:

Linux: **netstat -a | grep 9180** and **netstat -a | grep 8443**

Windows: **netstat -na | find "9180"** and **netstat -na | find "8443"**

If ports 9180 or 8443 are not available or are restricted on your host, select different (available) ports while you do this lab. Remember to use your chosen port numbers instead of port 9180 and 8443 for the rest of this lab.

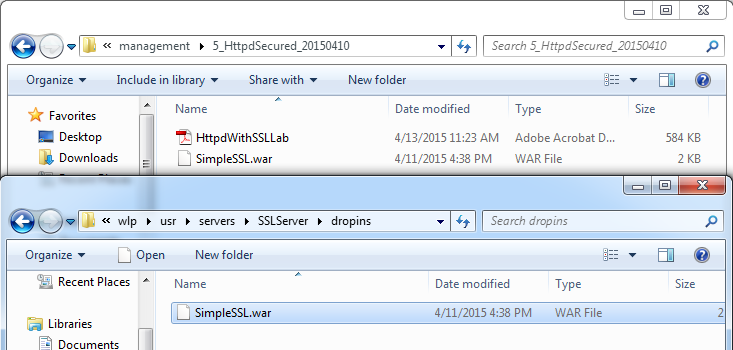
## Create the server SSLServer

1. Change directory to *{LAB\_HOME}*\wlp\bin
2. Run the command: **server create SSLServer**



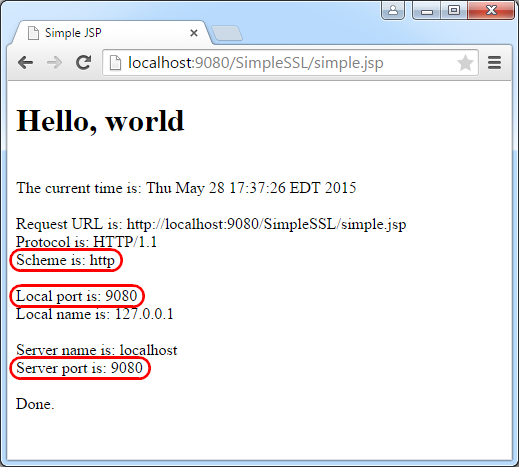
## Deploy the SimpleSSL sample

1. Copy *{LAB\_HOME}*\labs\management\1\_IHS\_SSLEnabled\_*<timestamp>*\SimpleSSL.war to *{LAB\_HOME}*\wlp\usr\servers\SSLServer\dropins directory



1. Change directory in the command-line window to: *{LAB\_HOME}*\wlp\bin
2. Run the command: **server start SSLServer**
3. Point your web browser to: <http://localhost:9080/SimpleSSL/simple.jsp>

Note the sample application prints the URL scheme (http), the local port the application is using (9080), and the server port in the URL, which is the same (9080).



1. (Optional) Import the application into Eclipse and browse the source code for **simple.jsp**. For your convenience, the source code is displayed below:

<HTML>

<HEAD>

<TITLE>Simple JSP</TITLE>

</HEAD>

<BODY>

<h1>Hello, world</h1>

<br>The current time is: <%= **new** java.util.Date().toString() %>

<br>

<br>Request URL is: <%=request.getRequestURL() %>

<br>Protocol is: <%=request.getProtocol() %>

<br>Scheme is: <%=request.getScheme() %>

<br>

<br>Local port is: <%=request.getLocalPort() %>

<br>Local name is: <%=request.getLocalName() %>

<br>

<br>Server name is: <%=request.getServerName() %>

<br>Server port is: <%=request.getServerPort() %>

<br>

<br>Done.

</BODY>

</HTML>

## Configure Liberty to use HTTPS/SSL

1. Edit *{LAB\_HOME}*\wlp\usr\servers\SSLServer\server.xml and add the following:
   1. Add the **transportSecurity-1.0** feature:

<featureManager>

<feature>jsp-2.2</feature>

**<feature>transportSecurity-1.0</feature>**

</featureManager>

* 1. Add a new key store definition. Note the password is **“labPassword”**:

<keyStore password="{xor}Mz49Dz4sLCgwLTs="></keyStore>

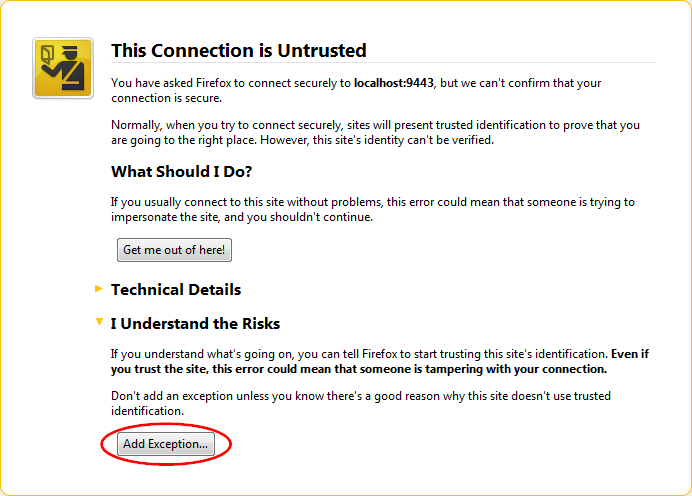
* 1. Save the **server.xml** file.

1. Ensure that you see the following message in the server’s **console.log** at *{LAB\_HOME}*\wlp\usr\servers\SSLServer\logs\console.log, and there are no error messages in the log:

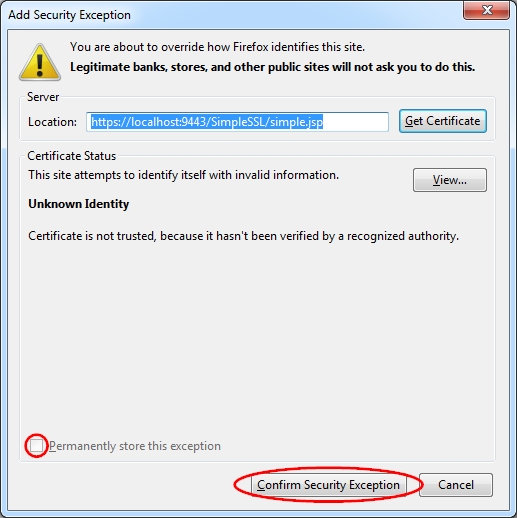
[AUDIT ] CWWKF0012I: The server installed the following features: [ssl-1.0, transportSecurity-1.0]

1. Point your browser to: <https://localhost:9443/SimpleSSL/simple.jsp>
2. You will be asked whether you want to continue despite the fact that the browser does not recognize Liberty’s certificate. Each browser asks you about this differently.

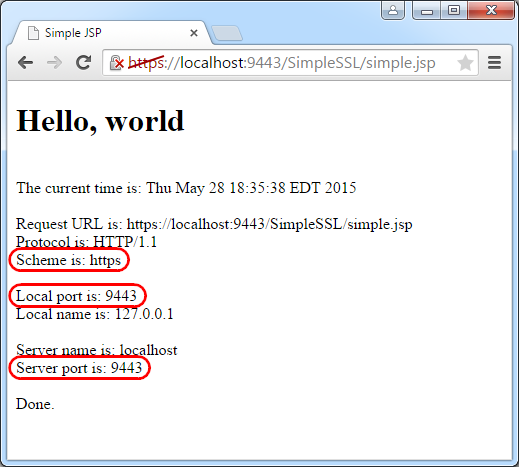
For Mozilla, you need to click the **Add Exception** button



Uncheck the **Permanently store this exception** option and then click the **Confirm security exception**:



1. Note the scheme is now **https**, and the local and server ports are both **9443**:



This shows that the Liberty server can execute our simple JSP using either HTTP (non-secure) via port 9080, or using HTTPS / SSL (secure) via port 9443.

## Using keytool to display content of keystore

Note that we have created a JKS-style keystore for Liberty, and the keystore lives in a file named *<path\_to\_Liberty>*\usr\servers\SSLServer\resources\security\key.jks. This keystore file is protected with a password of **“labPassword”**.

You can display information about the contents of the keystore using the keytool command from your Java JRE.

For example, the follow command:

keytool -list -keystore *{LAB\_HOME}*\wlp\usr\servers\SSLServer\resources\security\key.jks -storepass labPassword

will result in:



## Install and Configure IHS and the WebSphere plug-in

In the following steps we will add a separate IHS server. If you have already installed IHS and the plug-in for another lab you do not need to repeat that.

We will first configure IHS to use both HTTP and HTTPS. We will then configure the web server plug-in so that IHS server acts as a front-end for Liberty, receiving requests via either HTTP on port 9180 or HTTPS on port 8443, and forwarding them to Liberty for processing.

This lab includes a copy of IHS and the WebSphere plug-in, which can be found under *{LAB\_HOME}*.

1. Unzip the {LAB\_HOME}\IHS-<version>-<platform>.zip for your system into the ${LAB\_HOME} directory. This will create a {LAB\_HOME}\IHS directory where IHS is now installed. This contains a copy of IBM HTTP Server, which is the Apache web server that is provided with WebSphere. The zip also has the WebSphere web server plug-in pre-installed in IHS.

Note: On Windows, the command shell may need to be “Run as administrator”.

1. Run the IHS postinstall script to perform some host-specific setup of IHS. Note this script creates the directory that we will use later for the plug-in log files: {LAB\_HOME}\IHS\plugin\logs\webserver1.

Linux: {LAB\_HOME}/IHS/postinstall.sh

Windows: {LAB\_HOME}\IHS\postinstall.bat

1. Windows only: install IHS as a service using this command:

**{LAB\_HOME}\IHS\bin\httpd.exe -k install -n "IBM HTTP Server V9.0"**

The output should be similar to this:

****

1. Change the default HTTP port number for IHS by editing the main configuration file, *{LAB\_HOME}\IHS\conf\httpd.conf* as follows

change the line

Listen 0.0.0.0:80

to use port 9180 (or an alternative port, if 9180 is not available on your system):

Listen 9180

1. Change the directory to *{LAB\_HOME}*\IHS\bin

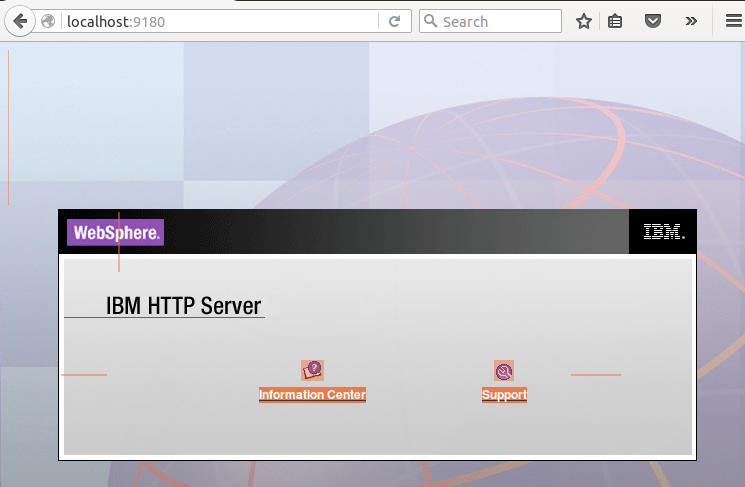
**cd *{LAB\_HOME}*\IHS\bin**

and start IHS with this command

Linux: **apachectl -k start -f conf/httpd.conf**

Windows: **httpd -k start -f {LAB\_HOME}\IHS\conf\httpd.conf**

1. Check IHS is running: point your browser to <http://localhost:9180>. It should show the screen below which proves that IHS server is up and running. This screen comes from *{LAB\_HOME}*\IHS\htdocs\index.html. (This htdocs directory is the “document root” for the Apache Httpd server.) Notice that <https://localhost:8443> fails, since we have not yet configured httpd for SSL.



## Configure IHS to use HTTPS/SSL

The following steps will enable the IHS to use SSL.

1. Stop the web server using the following command

Linux: **apachectl -k stop**

Windows: **httpd -k stop**

1. Edit *{LAB\_HOME}*\IHS\conf\httpd.conf using any text editor, and un-comment the following section by removing the hash character ('#') from the beginning of the lines. Leave IPv6 support with # marks

**# LoadModule ibm\_ssl\_module modules/mod\_ibm\_ssl.so**

**# Listen 0.0.0.0:443**

**## IPv6 support:**

**# Listen [..]:443**

**# SSLCheckCertificateExpiration 30**

**# <VirtualHost \*:443>**

**# SSLEnable**

**# Header always set Strict-Transport-Security "max-age=31536000; includeSubDomains; preload"**

**# </VirtualHost>**

**# KeyFile @@SERVERROOT@@/conf/ihsserverkey.kdb**

1. Change the SSL port number **443** to **8443** in three places that it occurs.
2. Add the following after the line **SSLCheckCertificateExpiration 30:**

**SSLServerCert libertylab.com**

The section should now look like this:

**LoadModule ibm\_ssl\_module modules/mod\_ibm\_ssl.so**

**Listen 0.0.0.0:8443**

**# IPv6 support:**

**#Listen [..]:8443**

**SSLCheckCertificateExpiration 30**

**SSLServerCert libertylab.com**

**<VirtualHost \*:8443>**

**SSLEnable**

**Header always set Strict-Transport-Security "max-age=31536000; includeSubDomains; preload"**

**</VirtualHost>**

**KeyFile {LAB\_HOME}/IHS/conf/ihsserverkey.kdb**

**SSLDisable**

**# End of example SSL configuration**

**SSLDisable**

Make sure “{LAB\_HOME}” points to the right directory

1. Save the **httpd.conf** file
2. Before you can configure IHS to accept SSL connections, you must create a keystore and certificate for your web server. In the command-line window, and use the IHS **gskcapicmd** command, which is included in the IHS installation, to create the server's private key and certificate as demonstrated below. To create the keystore, run this command in the IHS bin directory. Be sure to enter the command as one continuous string at the command prompt.

Change to the IHS/bin directory:

cd *{LAB\_HOME}\IHS\bin\*

run this command:

*gskcapicmd -keydb -create -db {LAB\_HOME}\IHS\conf\ihsserverkey.kdb -pw labPassword -stash*

Notice the ihsserverkey.\* files created in the {LAB\_HOME}/IHS/conf directory.

1. An SSL certificate authenticates your web server's identity to clients. To create the certificate, enter the following command:

*gskcapicmd -cert -create -db {LAB\_HOME}\IHS\conf\ihsserverkey.kdb -pw labPassword -dn "cn=www.libertylab.com" -label "libertylab.com" -size 2048*

Note that the label matches the SSLServerCert value (libertylab.com) that we added to the SSL section of the httpd.conf file in step 4.

Note that this certificate is good for development and test environments. For production environments you should use a certificate signed by a public certificate authority. This is beyond the scope of this lab.

You can examine the IHS server's certificate using the **gskcapicmd** command as demonstrated below. Again, be sure to enter the command as one continuous string.

gskcapicmd -cert -list -db {LAB\_HOME}\IHS\conf\ihsserverkey.kdb -pw labPassword

1. You should get an output similar to this:

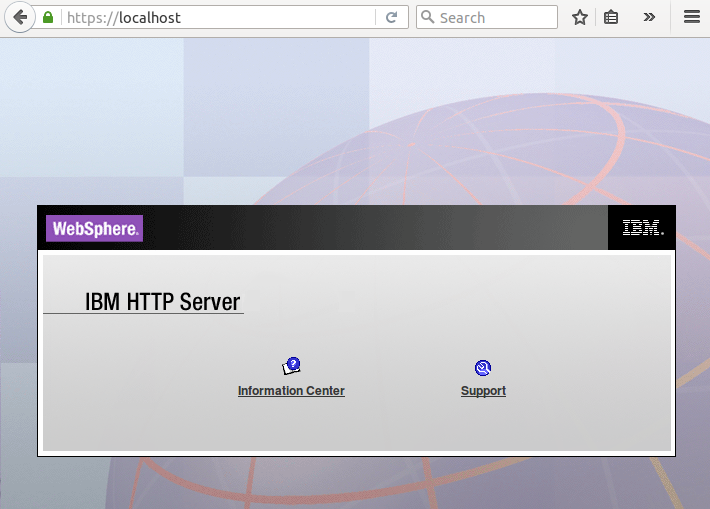


1. Start the web server. Change directory to *{LAB\_HOME}*\IHS\bin and run the command

Linux: **apachectl -k start -f conf/httpd.conf**

Windows: **httpd -k start -f conf\httpd.conf**

1. Now you should be able to type <http://localhost:9180> or <https://localhost:8443> in a browser, and you should get the welcome screen below (after a warning screen about the httpd server's certificate, similar to the warning that you got when you accessed the Liberty JSP via SSL, and for the same reason).



## Configure the plug-in between httpd and Liberty

We now have a running Liberty server that can be accessed via either HTTP or HTTPS / SSL, and a totally separate IHS server that is also accessible via either HTTP or via HTTPS/SSL. As things stand, the two servers have no awareness of each other whatsoever. The next step is to connect the two, by configuring the WebSphere web server plug-in so IHS can forward requests for Liberty applications. This will allow IHS to be used as a front-end for the Liberty server, which can be beneficial for a variety of reasons, including load-balancing and higher availability in cluster environments.

Since 16.0.0.3, the liberty server will automatically generate a configuration file that can be used to direct the IHS plug-in to direct HTTP requests to the liberty server. The file is generated when the server starts, and is regenerated if relevant configuration changes are made.

Your liberty server 'SSLServer' should still be running, if not then restart it now.

1. Add a <virtualHost> definition to your server.xml. This enable browser requests coming from any host to be directed towards the server’s ports of 9080, 9443, or the IHS server’s ports of 9180 or 8443.

<!-- Use virtual host default\_host for defaultHttpEndpoint -->

<virtualHost id="default\_host" allowFromEndpointRef="defaultHttpEndpoint">

<hostAlias>\*:9080</hostAlias>

<hostAlias>\*:9443</hostAlias>

<hostAlias>\*:9180</hostAlias>

<hostAlias>\*:8443</hostAlias>

</virtualHost>

2. Add a <pluginConfiguration> definition to your server.xml. This allows you to specify values to be used by the liberty server when generating the plugin configuration file.

<pluginConfiguration pluginInstallRoot="${LAB\_HOME}\IHS\plugin" logFileName="${LAB\_HOME}\IHS\plugin\logs\webserver1\http\_plugin.log" sslCertlabel="LibertyCert"/>

Be sure to replace ${LAB\_HOME} with the real location for your system, or define a variable for it in the server.xml file like this:

<variable name="LAB\_HOME" value="/home/liberty/wlp\_pot"/>

Note that the logFileName uses the directory that was created by the IHS postinstall script (this directory must exist or the plug-in will fail to start), and we will use the SSL certificate label later, when we generate the SSL certificate to use between the plug-in and the liberty server.

3. Examine the plug-in configuration file generated by the server in {LAB\_HOME}\wlp\usr\servers\SSLServer\logs\state\plugin-cfg.xml

## About plugin-cfg.xml

You can examine the plugin-cfg.xml with any text editor, but be careful not to modify it at this time. Notice the **<Route>** directive, which ties the **VirtualHostGroup**, **UriGroup**, and **ServerCluster** elements together.

The **<Route>** directive instructs the plug-in to forward requests for URL's that match the **UriGroup** (meaning, URL's that match **/SimpleSSL/\***) and requests that are sent to hosts in the **VirtualHostGroup** (meaning, any requests that arrive on port 9180 or port 8443); the requests should be forwarded to a WebSphere server in **ServerCluster** (meaning, the single server that listens on ports 9080 and 9443 – our Liberty server). In other words, the plug-in will cause httpd to forward any requests that belong to our SimpleSSL application to Liberty, and it will handle all other requests itself.

If we had an actual cluster instead of just one Liberty server, then the plugin-cfg.xml file could also include load-balancing directives and other directives related to handling Liberty servers that are temporarily unavailable.

The plugin-cfg.xml file is directly tied to your Liberty topology. If you make changes you to the topology, you have to update plugin-cfg.xml. If you were to add more applications to your Liberty server, you would have to copy the regenerated plug-in configuration file to the IHS install in order to access the new applications through IHS. The new plugin-cfg.xml file would include your new applications in **UriGroup**, so the plug-in would forward those requests to Liberty as well as requests for the SimpleSSL application. In the case of a Liberty cluster, you would also have to re-generate a new plug-in configuration file whenever you add or remove cluster members, to update the **ServerCluster** directives. If you manage your Liberty servers in a Collective you can use the Dynamic Routing feature to make those routing changes automatically; that is shown in Lab 3\_Dymanic Routing.

## Testing the plug-in

1. Stop the IHS server

Linux: **apachectl -k stop**

Windows: **httpd -k stop**

1. Copy the plug-in configuration file for the liberty SSLServer to the Plugin location:

**{LAB\_HOME}\wlp\usr\servers\SSLServer\logs\state\plugin-cfg.xml**

To this directory

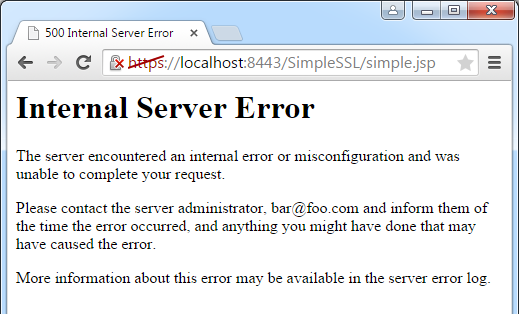
**{LAB\_HOME}\IHS\plugin\config\webserver1**

1. Start the http server

Linux: **apachectl -k start -f conf/httpd.conf**

Windows: **httpd -k start -f conf\httpd.conf**

1. Point your browser to <http://localhost:9180/SimpleSSL/simple.jsp>. You should see the JSP's output, where the local port (reported by the application) is 9080, and the server port is 9180:
2. Point your browser at <https://localhost:8443/SimpleSSL/simple.jsp>. You’ll see an internal server error because you have not yet configure SSL between the http server and SSL Server.



If you examine the http\_plugin.log file, you will see security errors when the server starts, because of the incomplete SSL configuration between the http server and SSLServer. However, you should see lines like this after the security errors, indicating that the plug-in still started successfully without SSL:

PLUGIN: Plugins loaded.

PLUGIN: --------------------System Information-----------------------

PLUGIN: Bld version: <version>

PLUGIN: Bld date: <date>

PLUGIN: Webserver: Apache/2.2.25 (Win32)

PLUGIN: OS : Microsoft Windows 32 bit

PLUGIN: Hostname = WIN7-X64

PLUGIN: OS version 6.1, build 7601, 'Service Pack 1'

PLUGIN: --------------------------------------------------------------

Notice that http://localhost:9180 still yields the IHS welcome page which is served by the Apache httpd server itself. The configuration instructions in plugin-cfg.xml tell the Apache server to forward requests for SimpleSSL to Liberty, but all other requests are still handled by the httpd server as usual, without any forwarding.

Also, you can bypass httpd and access the JSP directly at the Liberty server, with or without SSL, using <http://localhost:9080/SimpleSSL/simple.jsp> or <https://localhost:9443/SimpleSSL/simple.jsp>.

## Configure SSL between the Plug-in and Liberty

At this point, we have a simple JSP running in a web application in a Liberty server, and we can access that JSP directly via either HTTP or HTTPS. We have an IHS server and we can access its documents via either HTTP or HTTPS. Additionally, we have configured the web server plug-in so we can access the Liberty JSP through the IHS server, as long as we use only HTTP and not HTTPS. The final step is to finish configuring SSL between the plug-in and Liberty, so we can access the JSP via HTTPS through httpd using a URL like <https://localhost:8443/SimpleSSL/simple.jsp>.

The plugin-cfg.xml file is already prepared for SSL. It contains lines like these that specify a keyring for the plug-in to use when contacting Liberty via SSL:

<Transport Hostname="localhost" Port="9443" Protocol="https">

<Property Name="keyring" Value="{LAB\_HOME}/IHS/plugin/config/webserver1/plugin-key.kdb"/>

<Property Name="stashfile" Value="="{LAB\_HOME}/IHS/plugin/config/webserver1/plugin-key.sth"/>

<Property Name="certLabel" Value="LibertyCert"/>

</Transport>

Our task now is to create the proper plugin-key.kdb and plugin-key.sth files. The key-database file (.kdb) will contain the Liberty server's signing certificate, and the stash file (.sth) will hold the password for the .kdb file.

1. Change directory to the IHS bin directory:

cd {LAB\_HOME}\IHS\bin

1. Extract the Liberty server's signing certificate from its keyfile, which we can do via the **keytool** command as demonstrated here (type the command as one long line in a command window; you may have to adjust the path to **keytool** for your own Java JRE):

keytool -export –keystore *{LAB\_HOME}*\wlp\usr\servers\SSLServer\resources\security\key.jks -alias default -file {LAB\_HOME}\IHS\plugin\config\webserver1\libpub.cer -storepass labPassword

**On Windows** you may need to use **--keystore** in the above command

This produces a file named libpub.cer.

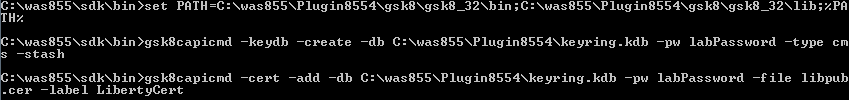


1. Create a separate keystore for the plug-in, then add the certificate contained in libpub.cer to the plug-in's new keystore. The keystore location needs to match what is specified in the plugin-cfg.xml file by the keyring property. We use the **gskcapicmd** to do this from a command-line window as demonstrated below. First to create the new keystore:

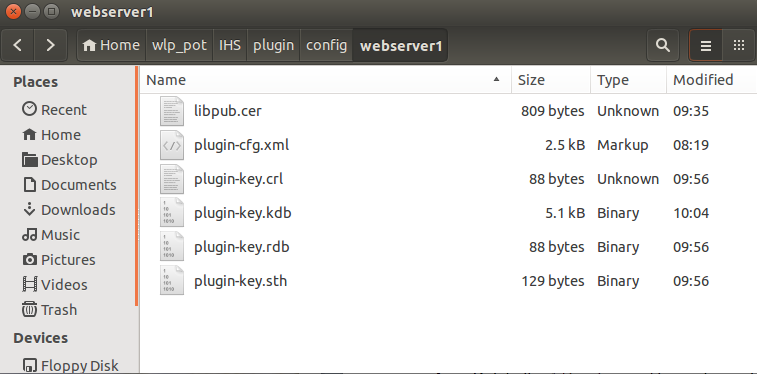
gskcapicmd -keydb -create -db {LAB\_HOME}\IHS\plugin\config\webserver1\plugin-key.kdb -pw labPassword -type cms -stash

1. Then add the certificate from libpub.cer to the keystore with label **LibertyCert**.

gskcapicmd -cert -add -db {LAB\_HOME}\IHS\plugin\config\webserver1\plugin-key.kdb -pw labPassword -file {LAB\_HOME}\IHS\plugin\config\webserver1\libpub.cer -label LibertyCert



After doing this, there should be several files in {LAB\_HOME}\IHS\plugin\config\webserver1 with the name of **plugin-key** and different suffixes

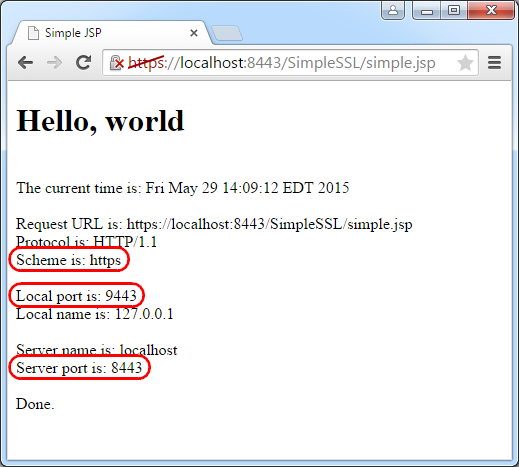


1. Delete the libpub.cer file, as it is no longer needed.
2. Restart IHS.

Linux: **apachectl -k restart**

Windows: **httpd -k stop** followed by**: httpd -k start -f conf\httpd.conf**

1. Point your browser to: <https://localhost:8443/SimpleSSL/simple.jsp>. Note that in the output the local (liberty server) port is 9443, and the (IHS) server port is 8443 of HTTP server. This shows we accessed the JSP using the HTTP server as the front end.



1. (Optional) Verify you can still access the IHS server's welcome page via either HTTP or HTTPS, and you still can access the Liberty JSP directly via either HTTP or HTTPS using ports 9080 and 9443 respectively.

If you have problems, verify your server.xml file against the example file provided in the lab (example\_SSLServer\_server.xml).

**Congratulations – it works!** This type of configuration, using a web server as a front end for a WebSphere Liberty server, is useful in several different situations. As you have seen, you can transparently serve static content (unchanging HTML files like the welcome page) from IHS, and serve dynamic content like JSP's and servlets from Liberty. In more complex environments, the plug-in can be configured to do load-balancing amongst multiple clustered Liberty servers, avoiding any Liberty servers that may be down, for higher availability and better scalability.

## Cleanup

Follow these steps to clean up

1. Stop the SSLServer

{LAB\_HOME}/wlp/bin/server stop SSLServer

1. Stop the HTTP server

Linux: **apachectl -k stop**

Windows: **httpd -k stop**

1. You can un-install the IHS server by deleting the *{LAB\_HOME}*\IHS directory tree if you wish.
2. Configuration files, key files, and log files

You have seen how you can access an application on the Liberty server via a web server and the associated plug-in. Each of these three components (web server, plug-in, and Liberty) has its own separate configuration files and key / certificate files, and each has its own log files. This section briefly summarizes the location and content of each of these files.

**Liberty server:** the Liberty server's home directory holds server.xml, which includes information about Liberty's keyfile (among other things). The keyfile lives in the server's resources/security directory, and it stores Liberty's private key and its public certificate. By default, Liberty's logs don't show when the JSP is accessed, but if you add a line like this to the JSP (just after the existing “Done” line), then you will see a line in Liberty's logs/message.log file every time the JSP is accessed:

<% System.out.println (request.getRequestURL() + " accessed at " + **new** java.util.Date().toString()); %>

**IHS web server:** the main configuration file is httpd.conf, which is in the *{LAB\_HOME}*\IHS\conf directory. The SSL configuration is defined in httpd.conf. Logs are in *{LAB\_HOME}*\IHS\logs, including error.log which shows general information about the server, and access.log which records all accesses made via HTTP.

Note that the httpd.conf file specifies the location of the plugin-cfg.xml. The plug-in configuration file must be in the location specified in httpd.conf, otherwise it will not be found and the plug-in will not work at all. Also note that the httpd.conf contains the locations of the KeyFile. This file must be in the location specified by httpd.conf, otherwise HTTPS / SSL will not work with the IHS web server.

**Plug-in:** the configuration file is plugin-cfg.xml and it resides in the plug-in configuration directory. The plug-in does not have its own private key or certificate, but it relies on the keyring files in *{LAB\_HOME}*\IHS\plugin\config\webserver1 to serve as a “trust store,” telling the plug-in which Liberty server certificates it should trust when establishing SSL. The locations of these keyring files are specified in plugin-cfg.xml, and the keyring files must be in the specified location, otherwise SSL will not work between the plug-in and Liberty. (HTTPS / SSL will still work for documents that are served directly by IHS, as long as HTTPS / SSL is properly configured for IHS itself).

1. Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing  
IBM Corporation  
North Castle Drive  
Armonk, NY 10504-1785  
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation  
Licensing  
2-31 Roppongi 3-chome, Minato-ku  
Tokyo 106-0032, Japan

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:** INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental. All references to fictitious companies or individuals are used for illustration purposes only.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

1. Trademarks and copyrights

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| IBM | AIX | CICS | ClearCase | ClearQuest | Cloudscape |  |
| Cube Views | DB2 | developerWorks | DRDA | IMS | IMS/ESA |  |
| Informix | Lotus | Lotus Workflow | MQSeries | OmniFind |  |  |
| Rational | Redbooks | Red Brick | RequisitePro | System i |  |  |
| *System z* | *Tivoli* | *WebSphere* | *Workplace* | *System p* |  |  |

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

ITIL is a registered trademark, and a registered community trademark of The Minister for the Cabinet Office, and is registered in the U.S. Patent and Trademark Office.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Linear Tape-Open, LTO, the LTO Logo, Ultrium, and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.



© Copyright IBM Corporation 2018.

The information contained in these materials is provided for informational purposes only, and is provided AS IS without warranty of any kind, express or implied. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, these materials. Nothing contained in these materials is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software. References in these materials to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. This information is based on current IBM product plans and strategy, which are subject to change by IBM without notice. Product release dates and/or capabilities referenced in these materials may change at any time at IBM’s sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way.

IBM, the IBM logo and ibm.com are trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at www.ibm.com/legal/copytrade.shtml.

