Dynix

Using the SSL protocol with HIP 3

Jeff Reed 8/1/2005

This document is for HIP system administrators who want to enable Secure Sockets Layer [SSL] on their HIP server. This is also commonly referred to as Hypertext Transfer Protocol Secure [HTTPS]. Administrators should have a solid working knowledge of the operating system of the HIP server and the HIP administration tools. They should also be familiar with HTTP, HTTPS and SSL.

For more information on your operating system, refer to the documentation and training provided by the manufacturer.

For more information on HIP administration, please refer to the *HIP System Administrator's Guide* or attend training offered by Dynix.

For more information on HTTPS and SSL, see the web sites for a local Certificate Authority such as Verisign or Thawte in the US, eSign in Australia, HiTRUST in Taiwan, Hong Kong, and Mainland China, Soltrus in Canada, and Firstream in Europe. This is not an exhaustive list, nor does it constitute a recommendation to use one of these CAs.

In order to deploy HIP 3 with SSL you must be running HIP version 3.0 or later. If you are running iPac 2.02 or above, please refer to the Tech Tip related to that release.

SSL involves the encryption and decryption of network traffic. This in turn requires additional CPU processing power and memory. Keep this in mind when sizing your HIP server and tuning it for performance.

Introduction

SSL provides two key security features. First, it provides assurance to the remote user that the web server they are communicating with is owned by the organization it claims to be provided by. Second, it encrypts the data passed between the user's web browser and the web server. It does not encrypt this data on the user's computer or on the web server. The security of SSL is facilitated by the use of digital certificates.

HIP is a Java based application and uses a Java Key Store to store digital certificates. A Java Key Store is an encrypted, password protected file in which digital certificates are stored. In order to support SSL, you must first create a Java Key Store and add a server digital certificate to the Key Store.

There are two options for obtaining a server certificate to use when enabling SSL on your HIP server. The first option is to create a self-signed certificate. The second is to obtain a certificate signed by a Certificate Authority.

Self-signed certificates are typically used for initial testing or for internal only use. A self-signed certificate **does not** provide assurance to the remote user that the web server is owned by the organization it claims to be provided by. When a user connects to HIP, their web browser will display a prompt window to allow the user to choose if they want to trust your web server. A user can decide to trust the HIP server for just this session or to install your self-signed certificate and trust your web server until the certificate expires. A self-signed certificate does enable encryption of the data between the web browser and the web server.

Certificates signed by a Certificate Authority [CA] establish a relationship of trust that is transparent to the user. The end users' web browser is configured by its author to trust the Root Certificate Authorities. By paying a CA to sign a certificate for you, the trust relationship is extended to include trust of your web server. Obtaining this certificate is not just an issue of paying the modest fee. You must also prove to the CA that you are who you claim to be. In many cases this is no small task. Plan ahead as a CA signed certificate normally takes two to three weeks to obtain. These certificates cost as much as $400.00 US and must be renewed annually.

Note: A CA signed certificate starts as a working self-signed certificate so start there in all cases.

The keytool utility for managing digital certificates in a Java keystore

The SUN Java SDK installed as part of the HIP installation includes the keytool utility. The keytool utility is a command line tool used to create the Java Key Store, add a self-signed certificate, create a Certificate Signature Request [CSR] and import a CA signed digital certificate.

The keytool executable is found in the bin directory of the Java SDK install.
For Windows systems, the default location of keytool is:
C:\j2sdk1.4.2\bin\keytool.exe

For Solaris servers using the SUN Java SDK, the default location of keytool is:
/opt/java/j2sdk1.4.2/bin/keytool

When running keytool, make sure you are using the version of keytool installed with HIP. This can be done by always specifying the full path to keytool such as the default location listed above or by adding the SDK bin directory to the front of your PATH.

Note: If you installed the Java SDK in another directory, you will need to adapt the instructions to match your installation.

To get a list of all the command line options available for keytool, run the following command:
keytool –help

Creating a self-signed certificate for SSL with Jetty/JBoss

Use keytool to create a keystore and generate the self-signed certificate. Specify the following options:

Option	Value	Meaning
-genkey		Generate a new keystore
-keystore	selfcert	Enter full path to HIP/JBoss/jetty/etc/selfcert
-alias	jetty	The alias for this certificate is "jetty"
-keyalg	rsa	Algorithm for key generation (default dsa)
-validity	365	Number of days certificate is valid (default 90)

Keytool will prompt you for the following information:

Prompt	Answer
Keystore password	Create a password for this keystore.
Your first and last name	The server's fully qualified domain name
Name of your organizational unit	Servers organizational unit
Name of your organization	The name of your organization
Name of your city or locality	City or locality
Name of your state or province	Enter the full name in all capitals.
What is the two- letter country code	Enter your two- letter country code in capitals.
Enter key password for <*alias*>	Create a password for the self-signed key.

Remember the passwords you set as you will need them later.

Note: If you plan to get a CA signed digital certificate, make sure all information is accurate or your signature request will be rejected and you will have to start over with the generation of a new keystore with the correct information. Contact your chosen CA for more information on properly answering these questions.

The following is an example of using keytool to create a java keystore and add a self-signed certificate named jetty.
On a Windows Server:
C:**>** C:\ j2sdk1.4.2\bin\keytool -genkey -keystore \dynix\hzapp\jboss\server\default\conf\selfcert -alias jetty -keyalg rsa –validity 365
On a Unix Server:
/opt/java/j2sdk1.4.2/bin/keytool -genkey -keystore /data/dynix/hzapp/jboss/server/default/conf/selfcert -alias jetty -keyalg rsa –validity 365

Enter keystore password: *1234567890* .
What is your first and last name?

[Unknown]: *HIP.library.org*
What is the name of your organizational unit?
[Unknown]: *web*
What is the name of your organization?
[Unknown]: *library*
What is the name of your City or Locality?
[Unknown]: *Provo*
What is the name of your State or Province?
[Unknown]: *UTAH*
What is the two- letter country code for this unit?
[Unknown]: *US*
Is <CN=HIP.library.org, OU=web, O=library, L=Provo, ST=UTAH, C=US> correct?
[no]: *y*
Enter key password for <jetty>
(RETURN if same as keystore password): *abcdefghijklmnop*

This should return you to a command prompt in 15-30 seconds.

You have now created a keystore with a self-signed digital certificate valid for 365 days. It will be located in the current location from which you are running the keytool command or in the path specified with the -keystore option.

Configure HIP to use SSL

Once you have a digital certificate, HIP needs to be configured to use it. This involves three steps. First, edit the jboss-service.xml and jboss-bindings.xml files to enable SSL support for the web server. Second, configure SSL support in the HIP admin tool. Third, restart HIP to apply your changes.

Edit jboss-service.xml and jboss-bindings.xml

Jetty needs to be told where to find the keystore and the passwords to open it. If you do not enter the passwords in the HIP configuration file, you will be prompted for them in the HIP console window. If you are running HIP as a Windows service or as a background process on Solaris, it will not come up unless you enter the passwords in the configuration file as it will be waiting in the background for these passwords.

IMPORTANT: If you are not running HIP in a console window, you MUST enter the passwords in jboss-service.xml and jboss-bindings.xml.

Edit dynix/hzapp/jboss/server/default/deploy/jbossweb-jetty.sar/META-INF/jboss-service.xml and configure it to use SSL using the following steps:
1. Remove the comments around the SSL section (labeled “Add a HTTPS SSL listener on port 8843”).
2. Change the port number to the standard https port of 443.

3. Set the Min and Max Threads to tune for your usage load. If you are not sure, use the numbers from the port 80 listener in the section above.
4. Change the name of the keystore file to match the one you created. If the keystore is not in the Dynix/hzapp/jboss/server/default/conf directory, copy it there first.
5. Set the Password to the keystore password entered earlier. (If you do not enter it here, HIP will prompt for it at every restart.)
6. Set the KeyPassword to the Key Password entered earlier. (If you do not enter it here, HIP will prompt for it at every restart.)

```
<!-- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -->
<!-- Add a HTTPS SSL listener on port 8843                          -->
<!-- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -->
<!-- UNCOMMENT TO ACTIVATE -->
<Call name="addListener">
  <Arg>
    <New class="org.mortbay.http.SunJsseListener">
     <Set name="Port">443</Set>
     <Set name="MinThreads">5</Set>
     <Set name="MaxThreads">1024</Set>
     <Set name="MaxIdleTimeMs">30000</Set>
     <Set name="LowResourcePersistTimeMs">2000</Set>
     <Set name="Keystore"><SystemProperty name="jboss.server.home.dir"/>/conf/selfcert</Set>
     <Set name="Password">1234567890</Set>
     <Set name="KeyPassword">abcdefghijklmnop</Set>
    </New>
  </Arg>
</Call>
```

Excerpt from jboss-service.xml configured to use the keystore generated above.
7. Copy this section of code for use in the next step and save and close this file.

8. Edit dynix/hzapp/jboss/server/default/conf/jboss-bindings.xml and configure it to use SSL by pasting the code from step 7 into the same location in this file (between the "Add and configure a HTTP listener to port 222" section and the "Add a AJP13 listener on port 8009" section).
9. Save and close this file.

IMPORTANT: The settings MUST be identical in each file in order for SSL to function. Otherwise, errors will occur when restarting HIP and SSL will not be enabled.

Configure SSL support in the HIP admin tool

HIP 3 can be configured in two modes, SSL required or SSL optional. Test SSL with HIP set for SSL optional before setting HIP to require SSL. The SSL setting uses HIP variables and can be set on a per profile basis. See the *HIP Administrator's Guide* for more information.

The SSL variable is found in the HIP admin tools under Customize → Interface → Variables → Information Portal Preferences (your copy thereof) under general. The variable is "enable switching from http to https". The default setting is false.

- To force all visitors to your online catalog to use SSL, set this value to “true”. *When set to true, HTTPS is enabled and any HTTP requests will be switched over to HTTPS.
- To make SSL optional, set this to “false”. *When set to false, both HTTP and HTTPS are enabled and either can be used.

Restart HIP and test
You must restart HIP to apply the changes made to the jboss-service.xml and jboss-bindings.xml files and enable the new digital certificate.

See the *HIP Administrator’s Guide* for information on stopping and starting HIP.

Stop HIP and start it in a console window by using the shortcut in the START → Programs, you should see the following during startup:
(HIP will pause for 5-10 seconds in the middle of this to generate keys.)

```
[JettyService] jetty.ssl.keystore=../../jetty/etc/selfcert
[JettyService] jetty.ssl.password=********
[JettyService] jetty.ssl.keypassword=********
[JettyService]
SSLServerSocketFactory=com.sun.net.ssl.internal.ssl.SSLServ
erSocketFactoryImpl@77460885
[JettyService] JsseListener.needClientAuth=false
[JettyService] Started SocketListener on 0.0.0.0:443
```

If HIP prompts for a password or generates an error here, go back and correct the problem.

Test your SSL implementation with your web browser by connecting to the HIP server using http**s** instead of http. For example, if I normally connect to my HIP server using http://HIP.library.org/ipac20/ipac.jsp?profile=main, I would connect using **https**://HIP.library.org/ipac20/ipac.jsp?profile=main
If you are using a self-signed digital certificate, your browser will ask you if you wish to trust the web server. Once you are connected to HIP, you can view the certificate and its trust tree using your web browser. See your web browser documentation for more information.

Obtaining a CA signed certificate for SSL with Jetty/Jboss

Follow the steps above to create a self-signed certificate and test it to ensure you have HIP properly configured.

Once you have HIP working with a self-signed certificate, you can use this self-signed certificate to generate a Certificate Signature Request [CSR], submit it to a Certificate Authority to get a signed certificate, and import the signed certificate into your keystore.

If you are testing using a CA test certificate, you will need to import the Test Root certificate first. If the Certificate Authority you are using is new, you may also need to install the CA root certificate. Both certificates will be available from your chosen CA. Import them following the same steps but use a different alias. An example of importing the Verisign Test Root certificate is shown in Appendix A.

Generate a CSR

Use keytool to generate a Certificate Request. Specify the following options:

Option	Value	Meaning
-certreq		Generate a certificate request
-keystore	Selfcert	Enter full path to dynix/hzapp/jboss/server/default/conf/selfcert
-alias	Jetty	The alias for this certificate is “jetty”

Create a Certificate Request with the following command:

keytool -certreq -keystore dynix/hzapp/jboss/server/default/conf/selfcert -alias jetty

Use the same keystore and alias used when you generated the keystore and self-signed certificate.

You will be prompted for the password to the keystore. This is the first password you entered when you generated the keystore and self-signed certificate.

For example, on a Solaris system with my current directory as /opt/dynix/hzapp/jboss/server/default/conf, I would enter:

/opt/java/j2sdk1.4.2/bin/keytool -certreq -keystore selfcert -alias jetty
Enter keystore password: *1234567890*

After a few seconds it will respond with a Certificate Request that looks similar to the following:

```
-----BEGIN NEW CERTIFICATE REQUEST-----
MIIBrTCCARYCAQAwbTELMAkGA1UEBhMCVVMxDTALBgNVBAgTBFV0YWgxDjAMBgNVBAcTBVB
yb3ZvMREwDwYDVQQKEwhlcGl4dGVjaDEMMAoGA1UECxMDZGV2MR4wHAYDVQQDExVpcGFjdG
VzdC5lcGl4dGVjaC5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBALYx/mUmP/LGD
w85NOfC0TGK8yb0kiSdPLMA0ViquEnO2Q0n/5SvdG6zByUJuBMSbG8QqKETBK3i7VxeELJT
xErAkA5D/hrk//yFKvBHhfW9WIluH6zvnfdu40Bae8AXPt6Cjyp6QyGQ7lzayBd9MIUsVGe
fbivlc/WqbI33ffjLAgMBAAGgADANBgkqhkiG9w0BAQQFAAOBgQCshL9UADfroaV2vs0AtI
ouyqttZjvPjVp0LuD1ZYHhcJZyDVsgVwjWisoMbOj12h4PGwHgyihMxClcxO9OcY9pet3Vd
```

```
wENKi0wp/53aNXTwRQUeby83uBwfC/dC5Hvj9mnEYGyojWElXstQiDj+0aWWLpys1oDmd3/
cLH0Q6N0Tw==
-----END NEW CERTIFICATE REQUEST-----
```

This is the CSR that you will need to copy and paste into the browser window as part of your Certificate Request to your chosen CA. Copy this text to a file and save it so that you can submit it to your Certificate Authority.

Installing a signed digital certificate

After you submit the right documentation and payment to your chosen CA, they will send you a signed certificate similar to the one below. Save this request to a text file. Make sure there is a blank line at the end of the file. (It must end with two carriage returns.)

```
-----BEGIN CERTIFICATE-----
MIIDLTCCAtegAwIBAgIQIbJbVsycbAt+0W8tk/+wDzANBgkqhkiG9w0BAQUFADCB
qTEWMBQGA1UEChMNVmVyaVNpZ24sIEluYzFHMEUGA1UECxM+d3d3LnZlcmlzaWdu
LmNvbS9yZXBvc2l0b3J5L1Rlc3RDUFMgSW5jb3JwLiBCeSBSZWYuIExpYWIuIExU
RC4xRjBEBgNVBAsTPUZvciBWZXJpU2lnbiBhdXRob3JpemVkIHRlc3Rpbmcgb25s
eS4gTm8gYXNzdXJhbmNlcyAoQylWUzE5OTcwHhcNMDIwNjE3MDAwMDAwWhcNMDIw
NzAxMjM1OTU5WjBtMQswCQYDVQQGEwJVUzENMAsGA1UECBMEVXRhaDEOMAwGA1UE
BxQFUHJvdm8xETAPBgNVBAoUCGVwaXh0ZWNoMQwwCgYDVQQLFANkZXYxHjAcBgNV
BAMUFWlwYWN0ZXN0LmVwaXh0ZWNoLmNvbTCBnzANBgkqhkiG9w0BAQEFAAOBjQAw
gYkCgYEAtjH+ZSY/8sYPDzk058LRMYrzJvSSJJ08swDRWKq4Sc7ZDSf/lK90brMH
JQm4ExJsbxCooRMEreLtXF4QslPESsCQDkP+GuT//IUq8EeF9b1YiW4frO+d927j
QFp7wBc+3oKPKnpDIZDuXNrIF30whSxUZ59uK+Vz9apsjfd9+MsCAwEAAaOB0TCB
zjAJBgNVHRMEAjAAMAsGA1UdDwQEAwIFoDBCBgNVHR8EOzA5MDegNaAzhjFodHRw
Oi8vY3JsLnZlcmlzaWduLmNvbS9TZWN1cmVTZXJ2ZXJUZXN0aW5nQ0EuY3JsMFEG
A1UdIARKMEgwRgYKYIZIAYb4RQEHFTA4MDYGCCsGAQUFBwIBFipodHRwOi8vd3d3
LnZlcmlzaWduLmNvbS9yZXBvc2l0b3J5L1Rlc3RDUFMwHQYDVR0lBBYwFAYIKwYB
BQUHAwEGCCsGAQUFBwMCMA0GCSqGSIb3DQEBBQUAA0EAWjLZAZ5VfFtisYfSGRvd
oQAVCdNE5z6kUBsDCjMeilyJ8KdeQ0nwaq7Ug7KAT4lpsLb29JUxSapx+SIL9rov
UQ==

-----END CERTIFICATE-----
```

Use the keytool command to install this certificate in your keystore.

Specify the following options:

Option	Value	Meaning
-import		Add a certificate to this keystore
-trustcacerts		Trust the recognized certificate authorities
-keystore	selfcert	Enter full path to HIP/JBoss/jetty/etc/selfcert
-alias	Jetty	The alias for this certificate is “jetty”
-file	Signed.cer	Full path to the CA certificate file you just saved

Import the signed digital certificate with the following command:
keytool -import -trustcacerts -keystore selfcert -alias jetty -file signed.cer
Use the same alias used to create the keystore and generate the CSR.

You will be prompted for the password to the keystore. This is the first password you entered when you generated the keystore and self-signed certificate.

For example, on a Windows system with my current directory as C:\dynix\hzapp\jboss\server\default\conf, and the first entry in my path set to C :\ j2sdk1.4.2\bin, and the signed certificate saved in the current directory as signed.cer, I would enter:

C:\> keytool –import -trustcacerts -keystore selfcert -alias jetty –file signed.cer
Enter keystore password: *1234567890*
Certificate reply was installed in keystore

If you have problems, check the instructions from your CA and contact them for further information if needed.

You can check the contents of your keystore with the command:

keystore –list

You can also view the certificate in readable format using your web browser connected to HIP via https.

Once you have imported the signed digital certificate, you must restart HIP to reload the keystore. First make sure that jboss-bindins.xml contains the right keystore path and passwords and recheck the settings for Min and Max connections to match your load. You should also watch memory usage and tune if necessary.

Appendix A: Example of importing the Verisign Test Root Certificate

You may see the following error with a trial certificate:
C:\jdk1.3.1_04\bin>keytool.exe -import -v -trustcacerts -alias jetty - file cacert –keystore selfcert

Enter keystore password: password
keytool error: java.lang.Exception: Failed to establish chain from reply

This can be resolved by importing the right root certificate from your CA.

Download and import the test CA root certificate from your CA. (Verisign's is currently available at:http://www.verisign.com/server/trial/faq/index.html)

C:\jdk1.3.1_04\bin>*keytool.exe -import -alias verisigntrial -file getcacert.cer -keystore selfcert*
Enter keystore password: *password*
Owner: OU=For VeriSign authorized testing only. No assurances (C)VS1997, OU=www.verisign.com/repository/TestCPS Incorp. By Ref. Liab. LTD., O="VeriSig n, Inc"
Issuer: OU=For VeriSign authorized testing only. No assurances (C)VS1997, OU=www.verisign.com/repository/TestCPS Incorp. By Ref. Liab. LTD., O="VeriSign, Inc"
Serial number: 52a9f424da674c9daf4f537852abef6e
Valid from: Sat Jun 06 18:00:00 MDT 1998 until: Tue Jun 06 17:59:59 MDT 2006
Certificate fingerprints:
MD5: 40:06:53:11:FD:B3:3E:88:0A:6F:7D:D1:4E:22:91:87
SHA1: 93:71:C9:EE:57:09:92:5D:0A:8E:FA:02:0B:E2:F5:E6:98:6C:60:DE
Trust this certificate? [no]: *yes*
Certificate was added to keystore
You should then be able to import your SSL certificate.