NHIN Direct REST Java Installation Notes

This document details steps for installing the NHIN Direct REST Java reference implementation on a Windows server and verifying correct installation. No assumption is made that the server already has certificates suitable for use with NHIN Direct, or even that a Certificate Authority is available to issue certificates for use with NHIN Direct. This document will therefore describe the steps necessary to set up one's own Certificate Authority, issue one's own certificates, sign those certificates by the Certificate Authority, and install the certificates in the necessary keystore and truststore files for use by NHIN Direct.

Throughout the remainder of this document, the single term Direct is a reference to the NHIN Direct REST Java implementation. The Java implementation of NHIN Direct REST is specifically addressed; no other implementations of Direct are herein addressed, and no implications regarding the suitability of any other Direct implementation are intended.

Directory path specifications in this document will use \ and / interchangeably. The backslash (\) will be used when the context is most likely DOS, and the forward slash (/) will be used when the context is most likely Cygwin.

Hardware / Network Requirements

For an instance of Direct to be of much value, it is required that Direct be installed on a minimum of two machines, and may be installed on any number of machines, each capable of connecting to the other over a port to be specified during the Direct installation. The installation of Direct on one machine is similar in nature to the installation of Direct on any other machine; the specific configuration details may differ, but the steps are the same. So this document will describe only the installation steps for one machine, and identify where customizations in configuration are required.

All machines that will make use of Direct for secure transport between them must use the same port for Direct connections. The default port for Direct is 8443, but this can be changed, as described later.

Direct was tested on a Windows machine with the following specifications:

- Windows Server 2008 Service Pack 2
- Intel Pentium 4 CPU 3.80GHz 3.79 GHz
- Memory (RAM): 3.00 GB
- System type: 32-bit Operating System

Direct was also tested on a Windows machine with the following specifications:

- Windows XP 2002 Service Pack 3
- Intel Core Duo CPU
- 2.93 GHz, 3.48 GB of RAM
- System type: 32-bit Operating System

Software Requirements

The following software tools are needed on each machine where Direct is to be installed. The versions listed were used in actual tests. Other versions of these software packages might also work.

- Java 1.6.0_18 (Java 1.6.0_24 was also tested)
- Java Cryptography Extension Unlimited Strength Jurisdiction Policy Files 6
- Python 2.6.5
- Maven 2.2.1

The above software tools must be installed according to instructions provided with them. Ensure that the JAVA_HOME environment variable is set for Java.

Moreover, one machine must be identified to act as the Certificate Authority for the other machines. This need not be a machine on which Direct is installed, but may be. On this machine, these software tools are required, and the versions used in testing are shown, though other versions might also work:

- Java 1.6.0_24
- OpenSSL 0.9.8r 8 Feb 2011
- Perl v5.10.1

These software tools should be installed according to instructions provided with them. Special setup considerations for OpenSSL will be discussed in the section Setting up a Certificate Authority. On the test machines, OpenSSL was installed by installing Cygwin and including OpenSSL as a package to be included in the installation of Cygwin.

The Direct software is included in the PHIX package. For sake of easy reference, the top-level PHIX package directory will be henceforth referred to as $HUB_HOME. The Direct software may then be found at $HUB_HOME\dependencies\direct\nhin-d-rest. Copy the entire nhin-d-rest directory to the location from which you wish to run Direct.

The Direct software may instead be downloaded according to the instructions found at http://code.google.com/p/nhin-d-rest/source/checkout.

Several Java libraries will be need when building Direct, but these are automatically downloaded by Maven as necessary, with one exception, discussed later.

The top-level directory from which Direct will run is referred to throughout this document as $DIRECT_HOME; this environment variable may be set if desired, or the path to the directory may be typed in any commands that make reference to $DIRECT_HOME.

Setting up a Certificate Authority

Note first that this document is not intended as an advanced explanation of certificate authority operation; only sufficient steps needed for getting Direct working are addressed, and some of these steps may need further research, depending on your exact set-up. Security is a very touchy business, and if anything is not exactly right within a given environment, the system likely will not work, or will not work as expected or required. A security specialist should be consulted before setting up an operational instance of Direct.

A Certificate Authority (CA) is needed to sign certificates used by Direct for establishing secure communications. Moreover, the certificate for the CA itself must be included in a truststore which Direct will use to establish trust in the certificates signed by the CA.

As mentioned above, the CA need only be set up on one machine. This is the CA Server. Also as mentioned above, OpenSSL can be installed as part of a Cygwin install. This was the choice made for the test machines. Since much of the information available about OpenSSL is in the context of a Linux environment, it made the process easier to use Cygwin to provide a Linux-style environment on Windows. When running any OpenSSL commands, they are assumed to be run from inside the Cygwin command window, and the configuration will assume that OpenSSL is running in Cygwin. Note that it is entirely possible to use a Linux machine as your CA Server, and still run Direct on Windows machines.

OpenSSL Configuration

Once OpenSSL has been installed, the openssl.cnf file must be located and modified. In the default installation on the test machine, the openssl.cnf file was found in /usr/ssl. The directory where you find openssl.cnf may differ; we will refer to this directory as $OPENSSL_HOME throughout this document. Edit the $OPENSSL_HOME/openssl.cnf file and make the following changes (as a precaution, you may wish to make a backup copy of the original file before making your edits):

1. In the `[ CA_default ]` section, make these changes:
    a. Change the value of the “dir” property to the full path where the CA data files are to be stored. This needs to be a directory in which the user who will run the openssl command has permission to create files.
    b. Change the value of the “default_days” property to the number of days for which it is desired that certificates signed by this CA will remain valid. Note that at the end of this period, the certificates signed by this CA will cease to work for Direct until the certificates are renewed, or new certificates are created and signed.
2. In the `[ req_distinguished_name ]` section, make these changes:
    a. Change the value of the “countryName_default” property to “US” (without the quotes), or to whatever two-letter code is appropriate for your country.
    b. Change the value of the “stateOrProvinceName_default” property to the full name of your state or province.
    c. Change the value of the “0.organizationName_default” property to the name of your organization.

Step 1a above is the most important. Make a note of the value you use for the “dir” property. This value will be referred to hereafter as $CA_DIR. This may be set as a Cygwin environment variable if desired. The other steps are not as important, since they can all be specified in other ways and are only provided here as defaults; refer to the OpenSSL documentation for details.

Certificate Authority Creation

The following steps will create a new CA based in $CA_DIR.

1. Copy the file CA.pl from $OPENSSL_HOME/misc to $CA_DIR.
2. Edit the $CA_DIR/CA.pl file in $CA_DIR as follows:
    a. Change the value of $CATOP to $CA_DIR/data.
    b. Change the numeric portion of the $DAYS variable to the same value as you used for the "default_days" property in the $OPENSSL_HOME/openssl.cnf file.
    c. Change the value of $CADAYS to the number of days you want the CA certificate itself to be valid. This is typically set for a longer period of time than the number of days that certificates signed by the CA are valid for. At the end of $CADAYS days after you set up the CA, the CA certificate will become invalid, at which time the CA certificate must be renewed and the truststore updated on every Direct machine that uses the CA. How to do this is beyond the scope of this document.
3. In a command window, cd (change directory) to $CA_DIR and perform the following steps (these steps assume you are using a Cygwin command window):
    a. Enter this command:

    ```
    ./CA.pl -newca
    ```

    b. This results in a series of prompts. For the first, you will be asked to enter a certificate filename or to press Enter to create a new one. Press Enter at this prompt.
    c. Some informative messages are printed. The next prompt asks for a PEM pass phrase. Enter the password you wish the CA certificate to have. You must remember this password, as there is no easy way to recover it, and if you forget it, you may have to create a new CA. Henceforth, this password will be referred to as CA Password.
    d. The next prompt asks you to verify the PEM pass phrase. Enter the CA Password again.
    e. For the next several prompts, you will be asked to enter identifying information for your CA, what is called a *Distinguished Name*. If you edited the $OPENSSL_HOME/openssl.cnf file as directed earlier, you will see your default value in brackets at the end of each prompt; when you see this, you can simply press Enter to accept the default. Otherwise, type your desired value and press Enter. When prompted to enter a "Common Name," enter the name by which you want your CA to be known; this should be something notable, preferably unique to your CA. It does not have to be the domain name of the machine, as will be the case for other certificates you'll create later. For the test CA Server, the Common Name "PHLISSA Demo CA" was used. It's recommended not to use the domain name of the CA Server if Direct will also be installed on the CA Server, as this could cause problems when you create the keystore for the machine, as described later.
    f. At some point, you will be asked to enter the pass phrase for a file whose base file name is cakey.pem; enter the CA Password.
    g. When you have entered something for each prompt, your new CA will be created, along with a data directory in $CA_DIR/data, where you will find a new directory structure,

under which, among other things, reside the CA's self-signed certificate (cacert.pem) and the CA's RSA private key (cakey.pem).

h. Set the permissions on your $CA_DIR/data directory so that outsiders cannot snoop around in there.
i. Back up the $CA_DIR directory, or at least the certificate and private key, in a secure place.

At this point, the CA is created, and is able to sign certificates for clients, as will be described later. It is rather useless at this point, however, in that neither it nor any certificates it signs will be trusted by any Direct instances. For that, you need to create and install a truststore containing the CA certificate, to be installed on each machine running Direct, as described later.

Creating a Trust Store for Direct Hosts to Trust Your CA

Each Direct instance requires a truststore file that contains one or more trusted CA certificates for establishing SSL REST connections. The Direct software package, as downloaded from Google, contained several keystores and truststores for different purposes, including a truststore for a CA for SSL REST connections, but certificates signed by that CA had expired, and Direct will not work with expired certificates.

Some of the keystores and truststores included in the Direct software package used different CAs than others. This in itself is not a problem, because the different keystores and truststores are put to different purposes. However, the situation can be confusing. The CA that is created by the above instructions could theoretically be used for all of the keystores and truststores as the signing CA, but this document will only describe its use to the extent that was necessary in the tested situation, which was to replace the use of the CA that had signed the expired certificates, which were intended for establishing SSL REST connections.

Note that technically there is no difference between a keystore file and a truststore file; they are both created and managed by the same set of tools. It is the purpose of a keystore file that determines whether it is a truststore or simply a keystore. A truststore file is a keystore file which contains certificates that are known to be trusted for a particular use by an application. Different applications, even on the same machine, may use different truststore files; even the same application may use multiple truststores, each for a different purpose. Java applications generally will trust certificates that are in the standard `cacerts` file, but this is not always the case. An application that is presented a certificate to trust that is not in the correct truststore will check to see which CA signed the certificate, and if the CA is in the correct truststore (or was signed by another trusted CA), then the certificate signed by the CA will be trusted. Thus, a truststore typically only contains trusted CA certificates.

To create a truststore with the CA certificate, you can use `keytool`, a standard Java program, to create an empty keystore, then add the CA certificate to it, by following these steps in a command window (these steps assume you use a Cygwin command window):

1. Change directory to $CA_DIR/data and verify that the CA certificate, `cacert.pem`, resides there. If it is not there, locate it and change directory to where it resides.

2. Convert the CA certificate to a DER format using openssl:

   ```
   openssl x509 -in cacert.pem -outform DER -out cacert.der
   ```

3. Create the empty keystore to be used as the truststore that will eventually contain the CA certificate.
   a. Create a keystore with the following command. You will be prompted for various bits of information, the *only* important piece of which is the keystore password, which it will prompt for first, and ask you to re-enter to verify. For test purposes, Direct is already configured to use “password” (without the quotes) as the password for the CA truststore. You’ll want something more secure for operational purposes. We’ll refer to the password you use here as the CA Truststore Password.

      ```
      keytool -genkey -alias tmpca -keystore truststore
      ```

   b. Delete the automatically generated certificate in the keystore with this command (enter the CA Trusted Password when prompted for the keystore password):

      ```
      keytool -delete -alias tmpca -keystore truststore
      ```

4. Add the CA certificate to the truststore with this command (enter the CA Trusted Password when prompted for the keystore password):

   ```
   keytool -import -file cacert.der -alias directca -keystore
   truststore
   ```

You now have a truststore file containing your CA certificate. This file will be referred to as the CA Truststore. Where to install the CA Truststore will be discussed later.

Creating a Keystore for Your Direct Host Machine

Having set up the CA and created the CA Truststore, you must next create a keystore for the machine on which you will install Direct, and you must know the domain name of this machine as it will be known to other machines with which it will be communicating via Direct.

The keystore hereby created will contain a certificate for the Direct host machine, signed by the CA. There are a number of ways by which this keystore file may be created, and we will discuss here one such set of steps that will produce the desired result. Consult a security expert to know what is best for an operational system.

The keystore file will need to contain both the certificate and the private key, because the private key will be required for successful encryption and decryption of data passed between Direct hosts. The keystore file for a Direct host will only reside on the Direct host itself; the certificate will be sent to other Direct hosts, but the private key will not. However, for another Direct host to trust the certificate it receives, the certificate must be signed by a CA whose certificate resides in the CA Truststore. Thus, we must create a keystore file with a matching certificate and key, and the certificate must be signed by our trusted CA. Both keytool and openssl will be of use in this process, and it is assumed the following openssl commands will be run from Cygwin. It is okay if the keytool commands are also run from Cygwin. It does not matter in which directory you start this process, and you could even create the keystore on the Direct host if you so chose, as it must end up there eventually. It is suggested you read the following steps carefully before executing any of them.

1. First, we create a keystore using keytool, producing a keystore file that contains both the certificate and the private key. In the command below, replace <alias> with a name of your choice; it is recommended that this alias be descriptive of the Direct host on which this keystore will be installed, to make it easy in the future to examine the keystore contents and know which machine it is for. You will need to re-use this alias later, so henceforth when you see <alias>, you must enter the same value as you choose here. Enter this command:

   ```
   keytool –keystore keystore –alias <alias> –genkey –keyalg RSA
   ```

   a. You will be presented with a series of prompts. The first prompt is for a keystore password. Direct's default configuration expects this to be "password" and that is okay for testing purposes. Choose something more secure for operational purposes. We'll refer to this password as the Direct Keystore Password.
   b. When prompted, re-enter the Direct Keystore Password to verify.
   c. You are next prompted for "first and last name." Ignore the wording of this prompt and enter the fully qualified domain name of the Direct host for which this keystore is being created. If the machine has a subdomain name by which it is known on the network, include the subdomain name. For instance, a test machine on the example.com domain might use "test.example.com" here.
   d. For the next few prompts, you will be asked organizational information that you may respond to as you see fit, the more descriptive of your organization, the better. For the key password, press Enter to let it default to the Direct Keystore Password.
   e. The keystore file is now created. We will refer to this keystore file as Direct Keystore, which at this point resides in a file named keystore.
2. You must now generate a certificate signing request for the certificate contained by Direct Keystore. You can do this using keytool. Enter this command, supplying the Direct Keystore Password when prompted for a password:

   ```
   keytool –certreq –alias <alias> –keystore keystore –file
   newreq.pem
   ```

3. This creates the request in the file newreq.pem. This file must be submitted for signing to the CA. To produce a signed certificate, copy newreq.pem to $CA_DIR on the CA Server. The following steps then take place in a command window on the CA Server (we'll assume you are using a Cygwin command window). Change directory to $CA_DIR. Then enter the following command (this requires you have Perl installed on your system):

   ```
   ./CA.pl –sign
   ```

   a. You will be prompted for the pass phrase for the CA private key; enter the CA Password.
   b. The details for the certificate will be displayed, so double check that you are signing the certificate request you think you are.
   c. You will be asked whether to sign the certificate, and then whether to commit the request; answer in the affirmative to both.
   d. The signed certificate is created in the file $CA_DIR/newcert.pem.

4. Testing has shown that trying to use only this signed certificate without it being coupled with the CA certificate does not work well for Direct's purposes. So we now create a combined certificate in a PKCS#7 format, which has a standard file extension of .p7b. Still in the $CA_DIR directory, enter the following command (this assumes your CA certificate is in $CA_DIR/data; if it resides elsewhere, you must change the last argument to point to the actual location of the CA certificate):

   ```
   openssl crl2pkcs7 –nocrl –certfile newcert.pem –out certs.p7b
   –certfile data/cacert.pem
   ```

5. You must now import the certs.p7b file into the Direct Keystore. Copy certs.p7b to the directory where the Direct Keystore resides. Then enter the following command:

   ```
   keytool –import –keystore keystore –alias <alias> –file
   certs.p7b
   ```

   a. Enter the Direct Keystore Password when prompted for a password.
   b. If you are informed that the top-level certificate is not trusted and asked whether to install anyway, answer in the affirmative.

6. You can verify that the Direct Keystore contains the necessary certificates. Enter the following command:

   ```
   keytool –list –v –keystore keystore
   ```

   a. Enter the Direct Keystore Password when prompted for a password.
   b. Starting from the top of the output, you should see the following info:
      i. The keystore contains 1 entry
      ii. The alias name matches what you chose for <alias>.
      iii. The creation date is the current date.
      iv. The entry type is PrivateKeyEntry
      v. The certificate chain length is 2.
      vi. The details for two certificates constitute the remainder of the output. For the first certificate, you should see the information for your Direct host machine, and it should be stated that the certificate was issued by the CA you used to sign the certificate.
      vii. The information for the first certificate is followed by the information for the second certificate, the certificate for the CA. It will show that the CA certificate was issued by the CA itself. In an operational system, you might be given

certificates signed by a chain of CAs, and in that case you'd expect to see the information for all certificates in the CA chain. Using a chain of CAs is outside the scope of this document.

At this point, the Direct Keystore is ready to be installed on the Direct host machine, along with the CA Truststore, as will be described below.

As mentioned earlier, there are multiple keystores and truststores used by Direct. The CA Truststore we created above is for use in establishing the SSL REST connection between Direct host machines, and the Direct Keystore is for one such Direct host machine. Other Direct host machines will need to have their own keystore files, but will use the same CA Truststore.

The steps for creating these files were necessary because the certificates in certain sample keystores provided in the Direct download had expired. At the time of this writing, keystores and truststores used by Direct for other purposes contain certificates that have not yet expired, and they serve their purposes as they are, so no new keystores and truststores were created to replace them. At some point in time these certificates too may need to either be renewed or be replaced; moreover, in an operational system, all certificates, keystores, and truststores would need to be created rather than using the sample ones provided in the Direct download. While the manner in which all certificates, keystores, and truststores are created is expected to be similar to that described above, this was not tested for all cases, and is beyond the scope of this document.

Configuring Direct on a Direct Host Machine

Follow these steps to configure Direct on your Direct host machine:

1. First, you may wish to create a backup copy of the original Direct installation. If something goes wrong, it might be easier to revert back to the original state from a local backup copy.
2. In a Command Prompt window, cd to $DIRECT_HOME. In this directory you will find some subdirectories, and only two of these are of importance to the Java version of Direct REST: driver and spring-maven-poc. Cd to spring-maven-poc. In this file you will find a README.TXT file; read it.
    a. In the "Introduction" section, note the versions of JDK and Maven required.
    b. In the "Installing JCE" section, we recommend that you download the required policy files directly from the official JCE download site.
    c. In the "Building" section, follow the directions for installing the nhin-d-jagent snapshot included in the nhin-d-rest project. The alternative is to download the nhin-d-jagent source and build it with `mvn install`, but the recommendation in the README.TXT file is to install the already built snapshot, and this is what was done during our testing.
    d. Next in the "Building" section you are instructed to build the software with a `mvn jetty:run` command. It is okay to do this now, though Direct will not yet be ready to work properly, so when the output indicates that the jetty server has started, press Ctrl-C to kill the server. The first time you run this command, it will take a while, so be

patient; Maven is downloading all the required libraries. Later, you will execute this command again to actually run Direct, but there are other configuration parameters that must be set first.

e. In the "S/MIME" section, there is nothing to do, since you will want to keep S/MIME enabled.

f. In the "Endpoint Certificates/Trust Anchors" section, there is a discussion of files certs.jks, trust-in.jks, and trust-out.jks. This discussion can be ignored, because these files are generated by Direct. Just above the "SSL Notes" section, it is mentioned that a mode might be set up where the HISP (the Direct host) has the CA certificate and generates keys/certificates/anchors for new users if they do not exist. This apparently was done, because it worked during testing. It is not stated how to turn off this mode if it is desired for these files to not be generated automatically. It appears that this functionality makes use of a CA certificate stored in the file `$DIRECT_HOME\spring-maven-poc\etc\endpoint-ca.jks`, and *not* the CA certificate in the CA Truststore. The CA certificate in endpoint-ca.jks is set to expire on Jun 5, 2012, so before then it will have to be either renewed or replaced, and/or the automatic generation of certificates/truststores for users turned off. All of these options are beyond the scope of this document, and were not tested.

g. In the "SSL Notes" section, it states that the server starts an HTTPS server on port 8443, and that the port is configurable in pom.xml. Indeed, the HTTPS server is started on port 8443, but if you wish to change it, you must do more than edit pom.xml as described in README.TXT. In addition to editing pom.xml, you must also edit the file `$DIRECT_HOME\spring-maven-poc\src\main\java\org\nhindirect\platform\rest\RestClient.java` and change the occurrence of 8443 in that file (line 66) to the port you wish to use. This change to RestClient.java and to the pom.xml file must be made on every Direct host machine in your network. If you want to be able to use different port numbers on different machines, the RestClient.java file mentioned above will need to be modified more than simply changing 8443 to some other number; what to do in this regard is beyond the scope of this document. Note: The next time the server is run, it will automatically build the modified RestClient.java file.

h. The "SSL Notes" section tells how to use HTTP on port 8080 instead of using HTTPS. Do not do this.

i. The "Keystore/Truststore" section discusses sample keystore and truststore files under $DIRECT_HOME\spring-maven-poc\etc. At the bottom of the section, it is stated that this information is obsolete. Not only is the information obsolete, but the certificates in the keystores are expired. Replace the keystore and the truststore files in $DIRECT_HOME\spring-maven-poc\etc with the Direct Keystore and CA Truststore files created earlier.

j. When you created the Direct Keystore and CA Truststore files, you gave each a password. If one or both have a password other than "password", you will need to change the configuration in pom.xml as depicted in the prior "SSL Notes" section. It is assumed this will work without requiring changes to other files, such as a Java file, but it was not tested. On the test servers, the default keystore and truststore password of "password" was used.

k. In the "Authentication & Authorization Notes" section, it is stated that either Basic Authentication or Client Cert Authentication may be used. In our tests, only Basic Authentication was used for user accounts, whereas the Direct host machines used Client Cert Authentication, which was the point of setting up the Direct Keystore and CA Truststore files.
l. The "Basic Authentication" section discusses how to edit the nhin-d-rest-security.xml file to create authenticated users and their passwords and roles. Place the user names and passwords in this file for those users who will send Direct messages from or receive Direct messages at this Direct host, and assign each user the ROLE_EDGE role. Do not include in this file the domain names of Direct hosts (HISPs), because they are set up to use certificates.
m. There follows some discussion of Roles and Client Cert Authentication, but nothing for you to do here.
n. In the "Domain Configuration" section, a domain.properties file is discussed. In this file, list the Direct addresses assigned to each user defined in the nhin-d-rest-security.xml file earlier. A Direct address looks exactly like an email address: user@domain. For best results, give each Direct address defined in the domain.properties file the same domain name as the Direct host machine. Note that any given Direct address can be made accessible by more than one user, and any one user can be granted access to more than one Direct address, if desired.
o. The remainder of the README.TXT file is informational only.

3. One item not mentioned in the README.TXT file that you want to check is whether $DIRECT_HOME\spring-maven-poc\data has write permissions. When Direct runs, it will create new directories for users to whom messages are sent, and if the data directory does not have write permissions, the attempt to send messages may fail with a cryptic error message.
4. At this point, you may run the command `mvn jetty:run` again and your Direct host machine is ready to communicate with other Direct hosts. To prepare other Direct host machines, you will follow all of the steps in this document again except for creating the CA Truststore; for that, all Direct host machines will use the same CA Truststore. Also remember to configure all Direct host machines to use the same port for Direct.

Verifying Your Direct Installation

A Python script, nhin-d-rest-client.py, resides in the $DIRECT_HOME\driver directory. A README.TXT file also resides in this directory. The README.TXT file explains the use of the Python script. The main thing to note is that any message sent by the Python script must already be in the RFC822 format; it must contain the correct "From:" and "To:" headers, and these must correlate to the arguments passed to the Python script.

To verify the Direct install on a given machine, send a message from a user on that machine to the same or another user on the same machine. You can then use the Python script to check that the message was sent. The README.TXT file explains the options and arguments for this.

It is possible to run the Python script from any machine with Python installed. Just copy the script to the machine with Python. To send a message, you first construct an RFC822-formatted message to be sent via Direct. Then you construct the command line for invoking the script. In the command line for the script, you use the --host option to specify the host on which Direct is running, and the --port option to specify the port on that machine to which Direct is listening. The --user option specifies the user that is sending the message, and needs to correlate to the “From:” header in the message (it doesn’t have to be the same user name, but it must be a user who is authorized for that address, as explained above). You use the --pass option to specify the password for the specified user. You will want to use the --ssl option, having configured Direct to use SSL as described above. One-way SSL was deemed sufficient for testing, so the --cert and --key options were not used and so are not herein documented.

The arguments follow the -- options, and are positional. The arguments to use when sending a message are as follows, in this order: The “to” address, POSTMSG, and the path to the RFC822-formatted file. The “to” address argument must match what is in the “To:” header in the RFC822-formatted file.

After you send a message from a user on a Direct host to that same user on that same host, if no error messages are forthcoming, you can use the Python script to check for the message. The README.TXT file explains how to do this. You can also look in the $DIRECT_HOME\spring-maven-poc\data directory for the subdirectory corresponding to your user and check that the message resides in a file there.

Once you have verified that a Direct instance can send and receive locally, you will want to check that it can send to and receive from another Direct instance on the same network. Both of these Direct instances must be using the same port for Direct. You use the Python script to send messages as described above, but now the “To:” address will direct the message to a different Direct instance than the Direct instance that handles the “From:” address. You will need to construct a new RFC822-formatted message containing the correct “To:” address, and change your arguments for the Python script to reflect the correct “to” address and correct file path for the new RFC822-formatted message.

Hopefully all goes smoothly with these tests. If they do not, then double-check that $DIRECT_HOME\spring-maven\data is writable, double-check that both machines are using the same CA Truststore file, and double-check that the certificates in the keystores were signed by the CA whose certificate is in the CA Truststore file. If you correct any issues found and yet problems still occur, you may need to check the validity of all certificates used by Direct, including making sure that none of them have expired.

The PHIX package contains a Java jar file that can be used to test Direct installations. The jar file is in the $HUB_HOME\DirectRESTIface\lib directory. The file name of the jar is hubdirect.jar. The jar contains a HubDirectSender class that may be used in a fashion similar to the Python script, but only handles plain text messages without attachments, and will automatically format the message as RFC822, generating the correct headers from the arguments supplied. The class requires the Java Mail library to be included in the class path. The JavaMail library can be found in the Maven repository after you install Direct. Execute HubDirectSender with no arguments to see a help message that lists all valid arguments.

If you have problems getting Direct to work correctly, you may wish to research on the web how to debug applications as they run in jetty, and use Eclipse or some other debugger to better see where errors are occurring in the web application and what the state of the variables are just before the errors occur. This is an advanced topic beyond the scope of this document.