JLicensure Installation Guide

Note: This is a preliminary version of the JLicensure Installation Guide. The contents and layout of this document are subject to change in the near future.

JLicensure Server

The JLicensure Server allows for a high level of customization to fit your specific needs in respect to how licenses are granted and stored. All software parts can be replaced with user-defined classes that implement specific functionality of the server, even though you probably don't want to change most parts of the default-implementation.

The default-implementation uses JDBC and SQL to store license data, so we proceed describing how to setup a database for the JLicensure Server.

Database Installation

You are free to use any type of relational SQL database as long as it provides a JDBC driver. The default-implementions of the LicenseStorage and LicenseGrant interfaces also assume that you provide the following:

- A **database table** to hold the license data.
- A string type table column of size 255 to store a base64-encoded data token (**token column**)
- A string type table column of size >= 1024 to store a base64-encoded signature (**signature column**)
- A numeric type table column that stores a counter value (**count column**)

The following is a script example that creates a table with the required columns. Please note that you may also use an existing table and add the required columns if you already have some sort of licensees data table. Table and column names can be chosen freely, however, using the default names as shown in the example script below doesn't require you to change the values in the JLicensure Server configuration file.

```
CREATE TABLE licenses
(
  (...)
  token varchar(255),
  signature varchar(1024),
  count int4,
  (...)
)
```

The **token column** is used to store a generic data token (usually a computed hash value) and the **signature column** is used to store the associated signature of that data token. Both fields are filled in (or overwritten) whenever a new license (a "license" is basically just a signed data token) is created and handed out by the JLicensure server.

The default-implementation of the LicenseGrant interface uses a simple counter to decide whether a license for a given table row should be granted or not. This is what the

count column is for. The value in the count column is decreased by 1 each time a new license is granted. If the counter is 0, a request for a new license is rejected. Thus, newly inserted rows should initialize the count column with a number value that is greater than 0.

Your database table may (and should) contain additional columns that hold the data related to a license. To identify a specific table row (license entry) when a license request comes in, the JLicensure default-implementation maps the generic set of properties (key=value string properties) that is associated with the license request to database table columns. The number, names and values of those properties are up to you. Thus, you are completely free in defining what kind of data the issuing of a license is based on.

Let's create a simple real-life example: Probably the most common practice to identify a license is some sort of serial number or a product key that comes with each copy of a given software product. So we add a table column to store that product key and call it "serial". Let's say we also want to store a product name and the licensee's name with the license, so we add columns "name" and "product". Here's the complete script for our license table:

```
CREATE TABLE licenses
(
  serial varchar(255),
  name varchar(255),
  product varchar(128),
  token varchar(255),
  signature varchar(1024),
  count int4 NOT NULL DEFAULT 2
)
```

Please note that only the token, signature and count columns are required by the default implementation. All other columns are user-defined. The string properties of an incoming license request are mapped to column names. Those columns must be of type varchar.

We'll explain later how new entries in the table can be inserted from "the outside" by using signed commands to the JLicensure server, but for now, let's just manually add a sample row to the table:

```
INSERT INTO licenses (serial, name, product) VALUES ('1234-5678', 'John Foobar',
'myproduct');
```

Please note that in our example the count column is automatically initialized with the default value of 2. This means, John Foobar may use the serial key '1234-5678' two times to receive a new license for the product 'myproduct'.

JLicensure Server Installation

In the jlicensure server directory, open and edit the jlicensure.properties file. In the database section, set the following properties according to your database setup. Please see the vendor-specifc documentation for your database's JDBC driver.

```
# The JDBC driver class name
db.driver =

# The vendor-specific JDBC connection URL
db.url =

# Database user name
db.property.user =

# Database password
db.property.password =
```

Also adjust the following settings according to your table and column names if you are not using the default names:

```
# The license table name
db.table = licenses

# The token column name
db.tokencolumn = token

# The signature column name
db.signaturecolumn = signature

# The count column name
db.liccountcolumn = count
```

Install the JDBC driver JAR

There are two ways to make your database's JDBC driver available to the JLicensure server:

- Copy the JDBC driver JAR to your [JRE_HOME]/lib/ext directory

OR

- Copy the JDBC driver JAR into your JLicensure server/lib folder, open the jlicensure-server[.bat] start script and set the "jdbc-driver-jar" property to the name of the JAR file.

Set the JAVA_HOME environment variable

If you have not already done so, set the JAVA_HOME environment variable to point to the installation directory of your Java SDK or Java Runtime Environment (JRE).

Run the server

Run the `jlicensure-server[.bat]` script to start the server.

A normal startup should look like this:

```
31.05.2006 17:12:33 com.componio.jlicensure.server.impl.Main main
INFO: Start com.componio.jlicensure.server.impl.Main
Creating 2048 bits rsa key pair.
31.05.2006 17:12:44 com.componio.jlicensure.server.impl.Main main
INFO: Startup complete.
31.05.2006 17:12:44 com.componio.jlicensure.server.impl.SocketAccepter start
INFO: Accepting connections on TCP port 32893 or trigger on UDP port 32891
```

Please note that during the first startup, the server creates a new key pair and stores the private and the public key in files key and key.pub, respectively.You should make sure that the private key file "key" is accessible and readable only by authorized users. It should never be made publicly available in any form. The public key file "key.pub" will be used on the client side to verify licenses.

Test the server

To test the server, you can use the command line tool `test-request[.bat]` located in your JLicensure server installation folder.

Please run the following command:

```
test-request localhost 32893 sample-request.properties
```

If everything works, the output should look like this:

```
--- START LICENSE REQUEST ------------------------------------------------------
POST /dummy/path HTTP/1.0
Content-Type: text/xml
Content-Length: 632

<?xml version="1.0" encoding="UTF-8"?>
<java version="1.5.0_06" class="java.beans.XMLDecoder">
 <object class="com.componio.jlicensure.shared.impl.LicenseRequestImpl">
  <void property="licenseeData">
   <object class="com.componio.jlicensure.shared.impl.LicenseeDataImpl">
    <void property="properties">
     <object class="java.util.Properties">
      <void method="put">
       <string>serial</string>
       <string>1234-5678</string>
      </void>
     </object>
    </void>
   </object>
  </void>
  <void property="tokenEncoded">
   <string>I68vU8na2H2lkCi0MbBrow==</string>
  </void>
 </object>
</java>
--- END LICENSE REQUEST --------------------------------------------------------

--- START SERVER RESULT --------------------------------------------------------
HTTP/1.0 200 OK
Content-Type: text/xml
Content-Length: 1114

<?xml version="1.0" encoding="UTF-8"?>
<java version="1.5.0_06" class="java.beans.XMLDecoder">
 <object class="com.componio.jlicensure.shared.impl.LicenseImpl">
  <void property="authoritative">
   <boolean>true</boolean>
  </void>
  <void property="licenseeData">
   <object class="com.componio.jlicensure.shared.impl.LicenseeDataImpl">
    <void property="properties">
     <object class="java.util.Properties">
      <void method="put">
       <string>serial</string>
       <string>1234-5678</string>
      </void>
     </object>
    </void>
   </object>
  </void>
  <void property="signatureEncoded">
   <string>aShysai9BIVFMoU9qvbLRbPuuSeTKtUh4GmZ9KECiRe8w4fi5+lEpo8Ic/d9q3NbV7J1HiLbW4dUho
zXOS4+HoKca2lS/TYvqHev7YhHCGJX34/Gfje2EKgN53x3Q2l8d9pO/mirCRMqLOM23junsRAkJ+P44OAF80gba/J
mhnGqJlRWveX3EdGzbZ+MuB+rRYWWRs6Xgf8x3EcUWpP6t6FQTnbNEd6GYRNAswtKdxmrpR0SPLGAh+MMzdEpSZ/t
0RZZfHkA2G2UF3qaTHZHEZpqAwLmXEig/Ruy0O3l523YTdSK3y3yDASPXv+Ua1FeprUFX6FCMLN76+FgXCERtg==<
/string>
  </void>
  <void property="tokenEncoded">
   <string>I68vU8na2H2lkCi0MbBrow==</string>
  </void>
 </object>
</java>
--- END SERVER RESULT ---------------------------------------------------------
```

If an exception is thrown, please check the console output of the jlicensure-server process to find out what the problem is. Fix the problem or edit the file sample-request.properties if you need to change the license request properties used by test-request. Re-run the test-request command until a license is returned as shown above.

JLicensure Client

The JLicensure Client is a set of libraries that you can bundle with your own software to enable license validation in your code. In this documentation, we confine ourselves to describing the prerequisites for integrating JLicensure with your software and show sample source code excerpts that can be used to test the functionality of JLicensure and the communication with the server. For further considerations regarding the variety of different licensing approaches please refer to the JLicensure User Guide.

The following files are needed and have to be bundled with your software to enable license validation:

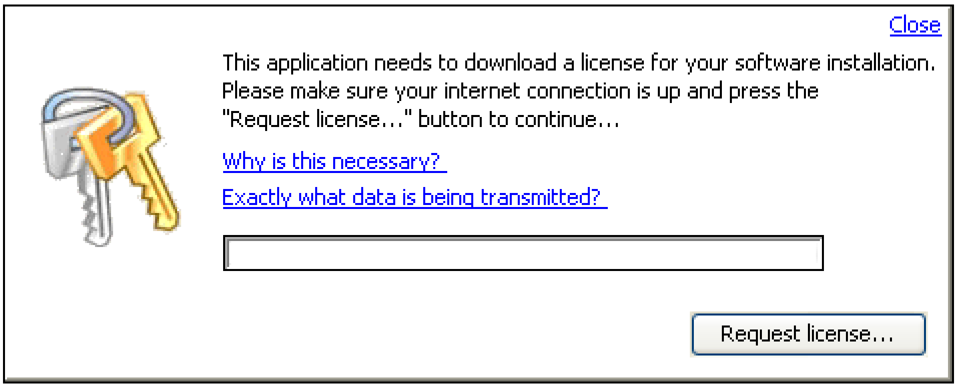
1. The server's public key file **key.pub** created during first startup of the JLicensure Server
2. The JAR libraries **jlicensure-client.jar** and **jlicensure-shared.jar** that contain the JLicensure interfaces for development
3. The SJAR libraries **jlicensure-client-impl.sjar** and **jlicensure-shared-impl.sjar** that contain the secured licensing code used during execution
4. The SJAR support libraries **sjarcl.jar** and the native **sjarcl.dll** (win32) (or **libsjarcl.so** for Linux/Unix, respectively).

We'll explain later when and where those files are needed during development and execution and in which form they have to be included.

Licensing code

To make sure that your software can only be run with a valid license, you have to include some lines of license validation code into your software. Since only you can know at what times during program execution a license validation must take place, the location(s) of the license validation code in your software is completely up to you. For instance, you may call the code only once during program startup or multiple times on invocation of a specific user action. Keep in mind, however, that there are two modes of operation of the JLicensure Client:

- **Silent mode**: This is the default mode. All validating operations work silently without a user interface and without user interaction. You may use this mode for non-interactive programs or for programs that cannot display a graphical user interface.
- **UI-mode**: This mode is enabled by calling setUIEnabled(true) in the LicenseCheck class. You should use this mode for interactive programs that display a graphical user interface. The JLicensure Client user interface pops up if (and only if) a new license must be obtained from the JLicensure Server to inform the user about the transmission of data over the network.



This application needs to download a license for your software installation. Please make sure your internet connection is up and press the "Request license..." button to continue...

Please note that once a license has been downloaded and backed up on the local machine, future license validations will run silently in the background as long as the locally backed up license stays valid.

If you're using the UI-mode of JLicensure Client and your application has its own user interface frame, you might want to call the license validating code while your application window is visible so that the JLicensure Client UI as shown above can act as a modal dialog child of your window. Use the setUIParent(...) method of the LicenseCheck class to set the UI parent for the JLicensure Client window.

Configure the JLicensure Client

Before you can use the JLicensure Client library, you must configure it for your specific setup. At this point, the only configuration parameters that you must change are host name and port of your JLicensure Server.

There are two ways to configure the JLicensure Client library:

1. **Persistently**: Update the properties file `com/componio/jlicensure/client/JLicensureFactory.properties` contained in `jlicensure-client.jar` by extracting it from the JAR and adding/updating it back into the JAR.

   These are the properties that you need to change to specify the server's host name and port:

```
# remote license server host
RemoteLicenseSource.host = localhost
```

```
# remote license server port
RemoteLicenseSource.port = 32893
```

2. **On-demand**: Set the properties that you need to change in your code before you call the create methods of the JLicensureFactory class.

   This is how you setup server host and port in your code:

```
JLicensureFactory.setProperty("RemoteLicenseSource.host", "localhost");
JLicensureFactory.setProperty("RemoteLicenseSource.port", "32893");
```

Write the licensing code

Add the two java libraries **jlicensure-client.jar** and **jlicensure-shared.jar** to your compiling classpath. The following packages must be included in the source files that use the JLicensure Client library:

```
import com.componio.jlicensure.client.*;
import com.componio.jlicensure.shared.*;
```

To obtain and validate a license, you must create a so-called license request (class LicenseRequest). To help you construct a new LicenseRequest object, you must use a LicenseRequestBuilder that can be obtained from the JLicensureFactory:

```
LicenseRequestBuilder lrb = JLicensureFactory.createLicenseRequestBuilder();
```

To constrain the validity of a license, we may add arbitrary data into the LicenseRequestBuilder by calling one of its add methods. The data is used later to construct a single hash value (the license token). Please refer to the JLicensure User Guide for further information on license validity and the license token. In our example, we bind the license only to the uniqueness of the local machine:

```
lrb.addHostUnique();
```

To identify a license or a license request, each License and LicenseRequest has a LicenseeData object that contains arbitrary information about the license or the license request. This information is transmitted in clear-text along with the license request. In our example, we only use an anonymous serial number to identify a license:

```
LicenseeData licDat = lrb.newLicenseeData();
licDat.setProperty("serial", "1234-5678");
```

Now you can obtain the constructed LicenseRequest from the LicenseRequestBuilder using the above LicenseeData:

```
LicenseRequest req = lrb.toLicenseRequest(licDat);
```

To check the availability and the validity of a license for the given license request, you need a LicenseCheck object:

```
LicenseCheck check = JLicensureFactory.createLicenseCheck();
check.setUIEnabled(true);  // optional to enable user-interface
check.setUIParent(myUI);  // optional to make JLicensure UI child of your UI
boolean result = check.checkLicense(req);
```

Here's a complete code example of a license checking method:

```
private boolean checkLicense() throws Exception
{
      LicenseRequestBuilder lrb = JLicensureFactory.createLicenseRequestBuilder();

      /* Constrain license validity */
      lrb.addHostUnique();

      /* Construct license identity */
      LicenseeData licDat = lrb.newLicenseeData();
      licDat.setProperty("serial", "1234-5678");

      /* Get resulting license request */
      LicenseRequest req = lrb.toLicenseRequest(licDat);

      /* Check license */
      LicenseCheck check = JLicensureFactory.createLicenseCheck();
      check.setUIEnabled(true);
      check.setUIParent(ui);
      return check.checkLicense(req);
}
```

Add the public key file to your classpath

Copy the server's public key file key.pub created during the server's first startup to your execution classpath. Either copy the file to one of your program's classpath folders or include it in one of your program's jar files (for instance the jlicensure-client.jar file).

If you must change the key file's name or if you want to place it in a subfolder, configure the key file name by setting the property `LicenseCheck.keyfile`. (See "Configure the JLicensure Client" above).

```
# License check public key file
LicenseCheck.keyfile = myfolder/mykey.pub
```

Run your program

In addition to the java libraries jlicensure-client.jar and jlicensure-shared.jar and the public key file, you also have to add the secure libraries **jlicensure-client-impl.sjar** and **jlicensure-shared-impl.sjar** to your execution classpath.

To support the usage of secure SJAR files in Java, do one of the following:

- Add the **sjarcl.jar** to your execution classpath and make sure the corresponding native library **sjarcl.dll** (or **libsjarcl.so** on Linux/Unix) is in your system's library path (Note: you can set java's library path by using the VM option `-Djava.library.path=<path>`)

OR

- Copy **sjarcl.jar** and the corresponding native library (**sjarcl.dll** for windows / **libsjarcl.so** for Unix/Linux) to the `[JRE_HOME]/lib/ext` directory of your java interpreter used for executing your program.

Note: The native libraries sjarcl.dll/libsjarcl.so are contained in the lib folder.

Working through firewalls with the JLicensure Proxies

Setup 1: Local Setup

In our example, we assumed that the JLicensure Client (as part of your application) connects to the JLicensure Server via a direct TCP/IP connection as shown in figure 1.

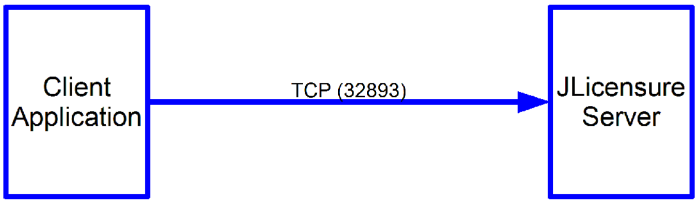


Figure 1: Local Setup

While this setup may work in a separate personal environment, things get more complicated when data transmissions must take place on the internet, possibly through existing firewalls.

Setup 2: Accessible Server Port

In a typical scenario where the place of a client installation is previously unknown to the software provider, you have to make sure that a license request can pass through any existing firewall that sets a limit to the range of ports and protocols accessible by the client application. While JLicensure uses the HTTP protocol to communicate over the network, it is up to you to provide a remote host that listens on TCP port 80 (HTTP), so that the client application can connect to it. The simplest solution is to run the JLicensure Server on a publicly available server host and to configure it to listen on TCP port 80 as shown in figure 2.

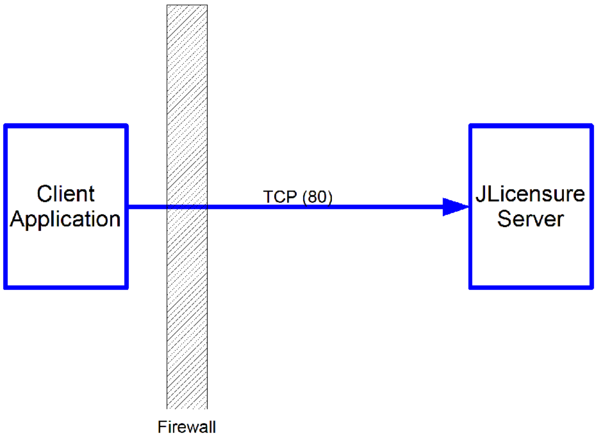


Figure 2: Accessible Server Port (Direct)

To do so, you can configure `accepter.port = 80` in your jlicensure.properties file.
In most cases, however, you might want to hide the JLicensure server location behind some sort of proxy instance as shown in figure 3.

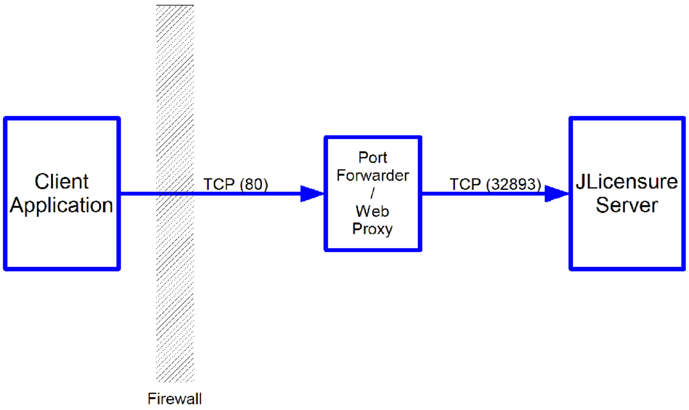


Figure 3: Accessible Server Port (Indirect)

The proxy instance could be a simple port forwarding host (possibly a masquerading firewall) or a web server (such as Apache) configured to redirect requests for a specific URL to another host and port (See "Setup 4: Web Server Proxying").

Setup 3: Secure Server

The previous setups required the JLicensure Server to expose, either directly or indirectly, its request handling TCP port to the public. Requests actively opened a TCP connection by initiating it from the outside.

To achieve a higher level of security for the JLicensure Server, you can use the JLicensure Proxies that allow both the JLicensure Server and the JLicensure Client to reside behind firewalls. Instead of opening or forwarding the request handling TCP port on the server-side firewall, only a UDP port has to be allowed to pass through the firewall, triggering the TCP connection to be opened by the JLicensure Server back to the triggering proxy.

Start the TriggeringProxy using the following command:

```
java -cp jlicensure-proxy.jar com.componio.jlicensure.proxy.TriggeringProxy
[options]
```

The options include:

```
-p<port>:  Front port number - client connection port, 80 in our example in
figure 4
-bp<port>: Back port number - server connection port, 32892 in our example in
figure 4 (You must allow the JLicensure Server to connect on this port through
the server-side firewall).
-rh<host>: Remote server host name to trigger, usually the server-side firewall
host
-rp<port>: Remote server UDP port to trigger, 32891 in our example (the default)
(You must allow and forward UDP packets on this port to pass through to the
JLicensure Server)
```

Configure the location of the TriggeringProxy at the server side by changing the property `accepter.remotehost` in your server's `jlicensure.properties` file.

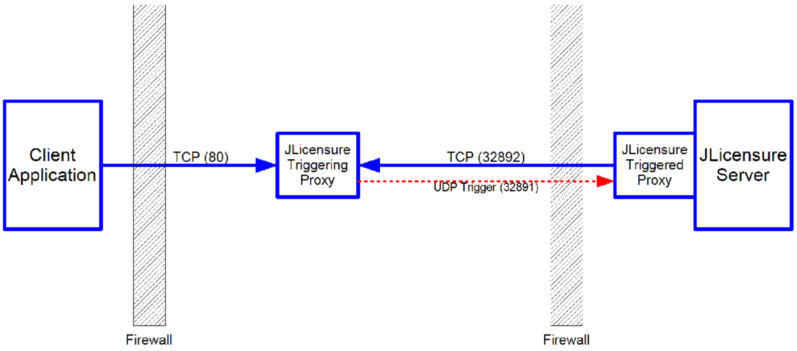


Figure 4: Secure Server

Setup 4: Web Server Proxying

If you're already running a web server that you can configure to fit your needs, then it might be a good idea to let the web server handle the incoming licensing requests on its standard port. That way, you don't have to worry about running your own server or proxy instance listening on port 80. If you combine the web server setup with a TriggeringProxy as described in "Setup 3" and if the TriggeringProxy is running in the secured environment of the web server (and therefore only accessible by the web server), then no JLicensure specific request handling port will ever be accessible from the outside at any time, giving you the highest degree of security (see figure 5).

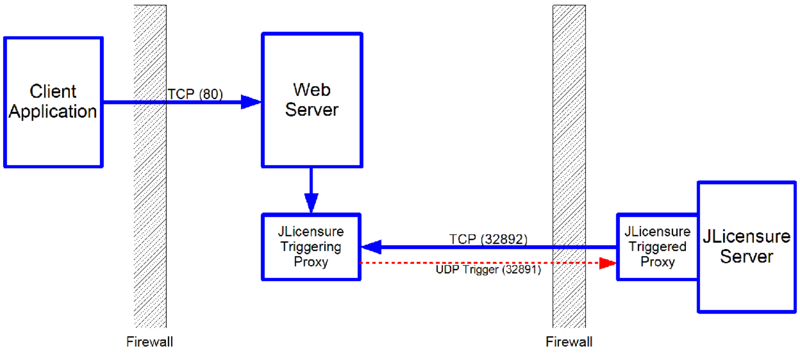


Figure 5: Web Server Proxying

To enable web server proxying, you must configure your web server to redirect requests for a specific URL to a given host and port. (For Apache, see "Reverse Proxies" in the Apache Documentation for mod_proxy).

To specify the URL that is requested by the JLicensure Client library, configure the property `HttpEncoder.path` in your JLicensure Client setup (see "Configure the JLicensure Client").

JLicensure WebModule

Besides the JLicensure Client and the JLicensure Server software components, a third component provides additional tools to communicate with the JLicensure Server. The JLicensure WebModule contains the software needed to send signed commands to the server, in particular the AddLicenseeDataCommand to add new licensees to the server's license table.

Adding new licensees using signed commands

To add new data to the license table, you may use the server's built-in capability of signed commands. The digital signature assures that only authorized sites can send those commands.

Create a key pair

To use signed commands, you must create a new keypair to be used by the JLicensure WebModule. This time, you leave the public (verifying) key at the server's side and bundle the private key with the WebModule.

To create a new key pair, please run the following command in your JLicensure Server installation folder:

```
genkey -owebkey
```

To alter key type and size options, run `genkey --help` for usage information.

The public key file webkey.pub is used by the server to validate incoming signed commands. If you have chosen a different name, edit the `verification-keyfile` property in `jlicensure.properties`.

Configure the JLicensure WebModule Library

There are two ways to configure the JLicensure WebModule library:

1. **Persistently**: Update the properties file `com/componio/jlicensure/webmodule/util/LicenseeDataAdder.properties` contained in `jlicensure-web.jar` by extracting it from the JAR and adding/updating it back into the JAR.

   These are the properties that you need to change to specify the server's host name and port:

   ```
   # server or proxy host
   server-host = jlicensure.foo.bar

   # server or proxy port
   server-port = 80
   ```

3. **On-demand**: Set the properties that you need to change in your code before you call the methods of the LicenseeDataAdder class.

   This is how you setup server host and port in your code:

   ```
   LicenseeDataAdder.setProperty("server-host", "jlicensure.foo.bar");
   LicenseeDataAdder.setProperty("server-port", "80");
   ```

Add the key file to your classpath

Copy the `webkey` file to your JLicensure Webmodule folder or simply add it to the jlicensure-web.jar.

Run the LicenseeDataAdder class

1) From the command line:

Create a properties file containing the licensee data properties. Then run the following command:

```
java -cp .;jlicensure-web.jar;jlicensure-shared.jar;jlicensure-shared-impl.sjar
-Djava.library.path=lib/win32;lib/linux
com.componio.jlicensure.webmodule.util.LicenseeDataAdder <properties-file-name>
```

Please use the colon ':' instead of the semi-colon ';' on Unix/Linux machines.

2) From Java code:

Use the LicenseeDataAdder class from your code:

```
import com.componio.jlicensure.webmodule.util.LicenseeDataAdder;
import com.componio.jlicensure.shared.*;

(...)

LicenseeData dat = LicenseeDataAdder.newLicenseeData();

dat.setProperty("serial","1234-5678");
dat.setProperty("name","John Foobar");
dat.setProperty("product","myproduct");

LicenseeDataAdder.sendLicenseeData(dat);
```