

ModJk

Running DSpace on Apache HTTPD and Tomcat using the mod_jk connector

For some background on why you'd want to do this, and the principles behind the configuration, see pages on [Securing DSpace](#) and [Running DSpace on Standard Ports](#).

Note: These instructions are for Linux, and may be somewhat specific to Red Hat Enterprise Linux 3.2.3-52 and the following software versions (but hopefully they are still helpful for other distros)

- DSpace 1.3.x and above
- Apache HTTPD 2.0.46
- Tomcat 5.5.9 and above

Anyone who has successfully set up mod_jk connector under different conditions should feel free to add their notes!

- Instructions for Gentoo Linux can be found at http://gentoo-wiki.com/index.php?title=HOWTO_Apache2_and_Tomcat5&redirect=no

Step 1 - Check if mod_jk is installed

Check to see if the mod_jk connector is installed. Most likely (at least on Red Hat), it should be in /etc/httpd/modules/. However, you can try to locate it using the following command:

```
locate mod_jk
```

If there is no response, then mod_jk is not installed. Otherwise, if it is installed, you can obviously skip the next step!

Step 2 - Install mod_jk from source

(1) Login as the root user. (or someone with "root"-like privileges)

(2) Before trying to build mod_jk, you must make sure you have the following pre-requisite programs installed (use the which command to check for each):

- libtool (<ftp://ftp.gnu.org/gnu/libtool>)
 - autoconf (<http://ftp.gnu.org/gnu/autoconf>)
 - ant (<http://jakarta.apache.org/ant/>)
- ant should already be installed (since it's necessary for DSpace). If libtool or autoconf are missing (both should be in /usr/bin), download the source and compile using the following commands:

```
./configure
make
make install
```

(3) In addition, you must have the Apache Web Server development tools installed. A quick way to check for this is to check for the APache eXtenSion tool (apxs). It should probably be in /usr/sbin, if installed:

```
which apxs
```

If apxs is missing, you can use the following command in Red Hat to install the httpd-devel RPM as root (*Other distros may need to find and install this RPM through other means*):

```
up2date -i httpd-devel
```

(4) Download the latest mod_jk source from the Tomcat Download site http://jakarta.apache.org/site/downloads/downloads_tomcat.html.

(5) Unzip the contents into your home directory:

```
gunzip -c jakarta-tomcat-connectors-1.2.14.1-src.tar.gz | tar -xvf -
```

(6) Configure the connectors with the path to the apxs file on your system:

```
cd jakarta-tomcat-connectors-1.2.14.1-src
cd jk/native
./configure --with-apxs=/usr/sbin/apxs
```

(7) Build mod_jk with the following command:

```
make
```

(8) Assuming all went well, the mod_jk.so file will be created in the apache-2.0 subdirectory. You need to copy this file to Apache's shared object files directory (e.g. /etc/httpd/modules/). From the same jk/native directory run the following:

```
cp apache-2.0/mod_jk.so /etc/httpd/modules
```

(9) In addition, copy the sample workers.properties file to the Apache configuration directory (e.g. /etc/httpd/conf/). Assuming you are still in the jk/native directory, run the following commands:

```
cd ../conf
cp workers.properties /etc/httpd/conf
```

Step 3 - Configure workers.properties

Once the mod_jk connector has been installed, you will have to configure Apache to use this connector to communicate with Tomcat. First, modify the existing workers.properties.minimal file (should be in /etc/httpd/conf/, or wherever you copied it to in Step 2 above):
You will need to modify the following Tomcat and Java home directories:

```
workers.tomcat_home=tomcat
workers.java_home=java
```

Also add ajp13 to the worker list:

```
worker.list=ajp13,lb,jk-status
```

In addition, you may need to uncomment (and possibly change) the JVM for Unix:

For later versions of mod_jk (I installed 1.2.40) worker.iprocess.jvm_lib is deprecated and would throw an warning. I omitted this line with no problems.

```
# Unix - Sun VM or blackdown
worker.inprocess.jvm_lib=$(workers.java_home)$(ps)jre$(ps)lib$(ps)i386$(ps)server$(ps)libjvm.so
```

Note: initially the path above was java/jre/lib/i386/classic/libjvm.so

However, the correct path of the libjvm.so (at least for Red Hat) is java/jre/lib/i386/**server**/libjvm.so (i.e. "server/libjvm.so", not "classic/libjvm.so")

Step 4 - Configure mod_jk connector

Next, you need to create a configuration file for the mod_jk module (alternatively, you could just add the following configuration directly into your Apache httpd.conf. I just like to separate things out a bit). In the /etc/httpd/conf.d/ directory (or whatever directory holds your external configuration files, which httpd.conf loads), create a file called jk.conf which has the following content (make sure to edit any paths so they are valid on your server!):

```
#
# Use the JK Module to connect to Tomcat Instance
#
# Load mod_jk module
LoadModule      jk_module  modules/mod_jk.so

# Where to find workers.properties
JkWorkersFile /etc/httpd/conf/workers.properties

# Where to put jk logs
JkLogFile      /var/log/httpd/mod_jk.log

# Set the jk log level debug/error/info
JkLogLevel      info

# Select the log format
JkLogStampFormat "%a %b %d %H:%M:%S %Y "

# JkOptions indicate to send SSL KEY SIZE,
#JkOptions      +ForwardKeySize +ForwardURICompat -ForwardDirectories
# Found that these options were necessary with Apache 2.2:
JkOptions      +ForwardKeySize +ForwardURIEscaped +ForwardDirectories

# JkRequestLogFormat set the request format
JkRequestLogFormat "%w %V %T"

# Send all requests for /dspace context to worker ajp13
# Note: ajp13 is defined in workers.properties and
# uses the AJP 1.3 Protocol
JkMount  /dspace/* ajp13

# ... and ditto if you want OAI
JkMount  /dspace-oai/* ajp13

#For extra security, deny direct access to any WEB-INF and META-INF directories
<LocationMatch "/WEB-INF/">
AllowOverride None
Deny from all
</LocationMatch>

<LocationMatch "/META-INF/">
AllowOverride None
Deny from all
</LocationMatch>
```

The big thing to pay attention to is the **context** which you specify in JkMount! If you specify /dspace/*, then only requests to http://my-host-name/dspace/* will be forwarded to Tomcat! However, if you specify /*, then **all** requests to http://my-host-name/* will be forwarded to Tomcat.

You can get a little tricky by doing something like:

```
# Send all requests for root context / to worker ajp13
# Note: ajp13 is defined in workers.properties and
# uses the AJP 1.3 Protocol
JkMount  /* ajp13

# Use SetEnvIf to set "no-jk" when /cgi-bin/ is encountered.
# This is necessary so that /cgi-bin/ scripts
# are run in Apache (and not forwarded to Tomcat).
SetEnvIf Request_URI "/cgi-bin/" no-jk

# Set "no-jk" for /anotherApp/ as well (so it is run from Apache)
SetEnvIf Request_URI "/anotherApp/" no-jk
```

Notice, first you specify that **all** requests should be forwarded to Tomcat. But, then for specific UI's you can specify to ignore mod_jk (using the no-jk environment variable). So, the above specifies that everything except paths matching http://my-host-name/cgi-bin/* or http://my-host-name/anotherApp/* are forwarded to Tomcat.

Step 5 - Configure Tomcat

Next, you need to take a look at the Tomcat `server.xml` configuration file (in the `/conf` subdirectory, wherever Tomcat is installed). Ensure that the following AJP 1.3 Connector is uncommented:

```
<!-- Define an AJP 1.3 Connector on port 8009 -->
<Connector port="8009" UIEncoding="UTF-8" tomcatAuthentication="false"
enableLookups="false" redirectPort="8443" protocol="AJP/1.3" />
```

Make sure that the port specified corresponds to the port you defined for the `ajp13` worker (this port number is defined in the `workers.properties` file, as shown above). In addition, make sure the `UIEncoding` is set to `UTF-8`, and `tomcatAuthentication` is set to `false` (assuming you want authentication taken care of in Apache rather than Tomcat). Finally, make sure you set the `redirectPort` to be the port that Tomcat is running on (usually either 8443, for HTTPS, or 8080, for HTTP).

Step 6 - Restart everything and Test!

Restart Tomcat and Apache!

Now, test the connection between Apache and Tomcat. You should now be able to get to DSpace whether you specify port 8080 (for Tomcat) or not. For example, the following URLs should bring you to the same DSpace (you may need to replace `localhost` with your server path):

- <http://localhost:8080/dspace>
- <http://localhost/dspace>

Hopefully everything works for you! If it doesn't, ask questions to the [Mailing Lists](#). If you find any problems with the above instructions, feel free to **edit and enhance** them!