Integrating Apache Web Server with Tomcat Application Server

The following document describes how to build an Apache/Tomcat server from all source code. The end goal of this document is to configure the Apache web server to forward all web application requests on ports 80 and 443 to the Tomcat Application server. Both servers run on the same system. The tutorial configures Apache to access the example servlets provided by Tomcat with the option to execute the servlets.

The configuration environment is listed below:

- Solaris 9 for Intel – Release 12/03
- Apache Web Server – 2.0.59
- Apache Tomcat Server – 5.0.17
- mod_jk Tomcat Connector – 1.2.18
- Java SDK – 1.4.2-08

Preparing the Solaris 9 Intel System

The Solaris 9 base operating system was used as the platform for Apache/Tomcat configuration. The following actions were taken to install the OS:

- Distribution – Entire distribution
- IP Address – `172.16.169.128/255.255.255.0`
- Default router – `172.16.169.1`

All of the following actions must be run as the `root` superuser.

The Tomcat server is installed in the directory `/apps/BOXIR2/bobje`. This directory needs to be created:

```
# mkdir –p /apps/BOXIR2/bobje
```

The Tomcat server runs as the user bobjer2. This user must be created first:

```
# useradd –d /apps/BOXIR2 bobjer2
# passwd bobjer2
New Password:
Re-enter new Password:
passwd: password successfully changed for bobjer2
```

© 2006 McManus Consulting

Since the user bobjer2 will stop and start the Tomcat server, the following profile must be created:

```
# su – bobjer2
$ vi .profile
#       This is the default standard profile provided to a user.
#       They are expected to edit it to meet their own needs.

MAIL=/usr/mail/${LOGNAME:?}
HOME=/apps/BOXIR2/; export HOME
ORACLE_HOME=/opt/Oracle/product/9.2.0; export ORACLE_HOME
LD_LIBRARY_PATH=$ORACLE_HOME/lib; export LD_LIBRARY_PATH
PATH=$PATH:$ORACLE_HOME/bin; export PATH
LANG=en_US.UTF-8; export LANG
JAVA_HOME=/apps/BOXIR2/java/j2sdk1.4.2_08; export JAVA_HOME
cd $HOME
```

Preparing Solaris 9 to Build Software

Solaris 9 does not ship with any of the GNU complier software utilities. These must be installed in order to be able to compile the Apache web server and the mod_jk connector. The following packages were downloaded from the ibiblio.org mirror:

```
http://www.ibiblio.org/pub/solaris/freeware/intel/9
```

As the root user, create a repository for the Solaris package files:

```
# cd /var/tmp
# mkdir SOURCE FREEWARE
# cd FREEWARE
```

Using either wget or ftp, download the following packages from the ibiblio.org website:

```
# pwd
/var/tmp/FREEWARE
# ls
autoconf-2.59-sol9-intel-local
glib-2.6.2-sol9-intel-local
automake-1.9-sol9-intel-local
libiconv-1.9.2-sol9-x86-local
openssl-0.9.8b-sol9-x86-local
binutils-2.11.2-sol8-intel-local
libtool-1.5-sol9-intel-local
tar-1.15.1-sol9-intel-local
bison-1.875d-sol9-intel-local
zlib-1.2.3-sol9-x86-local
flex-2.5.4a-sol9-intel-local
m4-1.4.2-sol9-intel-local
gcc_small-3.4.2-sol9-intel-local
make-3.80-sol9-intel-local
```

The binutils package is not available for Solaris 9, but is compatible with Solaris 8. Download binutils from the following link:

```
 http://www.ibiblio.org/pub/solaris/freeware/intel/8
```

© 2006 McManus Consulting

Using the pkgadd command, add all of the software packages. The following example demonstrates how to install the autoconf package. Repeat this command for each one of the above mentioned files:

```
# pkgadd –d autoconf*
```

In order to compile software, the root profile must contain certain variables. These include:

```
# vi /.profile
PATH=/usr/local/bin:/usr/local/sbin:/usr/local/ssl/bin:$PATH:/usr/sfw/bin
export PATH
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib:/usr/local/ssl/lib
export LD_LIBRARY_PATH
EDITOR=vi
export EDITOR
MANPATH=$MANPATH:/usr/local/man:/usr/local/ssl/man
export MANPATH
CC=gcc
export CC

# . /.profile
```

Finally, link the gcc compiler to a file named cc in order for the compiler scripts to locate the appropriate compiler:

```
# cd /usr/local/bin
# ln gcc cc
```

The Solaris 9 environment is now ready to compile software.

Building Apache from Source

The Apache web server is built from source using the compiling environment just created on the Solaris 9 system. In order to compile Apache, the source code must be downloaded from the Apache website. At the writing of this document, version 2.0.59 is used.

Using the wget command, download the Apache source:

```
# cd /var/tmp/SOURCE
# wget http://apache.cs.utah.edu/httpd/httpd-2.0.59.tar.gz
```

Unpack the Apache source software:

gzcat httpd-2.0.59.tar.gz | tar xvf –

Once the source is unpacked, prepare the source code using the following configure string:

```
# cd httpd-2.0.59
# ./configure --prefix=/opt/apache2 --enable-ssl --enable-module=so
```

Once the software is prepared, run the make command to compile the software:

```
# make
```

Once the software compiles, install the software into the /opt/apache2 directory:

```
# make install
```

© 2006 McManus Consulting

Verify that Apache installed correctly by validating the build:

```
# cd /opt/apache2/
# ls
bin      cgi-bin  error    icons    lib      man      modules
build    conf     htdocs   include  logs     manual
```

Apache is now installed.

Change the ownership of the Apache directory to the `bobjer2` user:

```
# chown –R bobjer2:other /opt/apache2
```

Edit the Apache configuration file to make the `bobjer2` user the owner of the Apache process:

```
# cd /opt/apache2/conf
# vi httpd.conf
```

CHANGE:

```
User nobody
Group #-1
```

TO

```
User bobjer2
Group other
```

Installing Java SDK

The Java SDK must be installed in order for Tomcat to function properly. The Java SDK is available from Sun Microsystems download page at:

`http://java.sun.com/products/archive/j2se/1.4.2_08/index.html`

Be sure to download the self-extracting file for Solaris Intel (not SPARC), not the Java packages. The Java SDK must be installed in the Tomcat directory. Download the SDK and place it in the `/apps/BOXIR2/java` directory:

```
# cd /apps/BOXIR2
# mkdir java
# cd java
# ls
j2sdk-1_4_2_08-solaris-i586.sh
```

Make the file executable with the `chmod` command and run the installer. You will be prompted to accept the license agreement.

```
# chmod +x j2sdk*
# ./j2sdk-1_4_2_08-solaris-i586.sh
```

You have now installed the Java SDK.

© 2006 McManus Consulting

Installing Tomcat Server

The Tomcat Application server does not need to be compiled. It is a full Java implementation and will work with the Java SDK installed. At the time of the writing of this document, the Tomcat 5.0.17 server was used. This is an older version of the server and it can only be downloaded from the archive site.

Change directories to the `/apps/BOXIR2/bobje` directory and download the Tomcat server.

```
# cd /apps/BOXIR2/bobje
# wget http://archive.apache.org/dist/tomcat/tomcat-5/archive/v5.0.27/src/jakarta-tomcat-5.0.27-src.zip
```

Unzip the Tomcat server in the current location:

```
# unzip jakarta-tomcat-5.0.27-src.zip
```

Change the ownership of the Tomcat server to the `bobjer2` user:

```
# chown –R bobjer2:other jarkata-tomcat-5.0.27
```

The main configuration file `server.xml` contains control characters in it. This file must be cleaned using the `dos2unix` command:

```
# cd Jakarta-tomkat-5.0.27/conf
# dos2unix server.xml server.xml
```

Building the JK Connector

The JK Connector (`mod_jk.so`) is an Apache module that glues Apache to Tomcat. By default, all requests that come in on 80 or 443 that are mapped to Tomcat directories, are forwarded to Tomcat locally over port 8009. The JK Connector needs to be built as a loadable module into the Apache web server. This is possible due to the fact that the Apache web server was compiled with DSO support (`--enable-module=so`). At the time of the writing of this document, JK 1.2.18 was the latest module available.

Download the JK source from the Apache website:

```
# cd /var/tmp/SOURCE
# wget http://www.axint.net/apache/tomcat/tomcat-connectors/jk/source/jk-1.2.18/tomcat-connectors-1.2.18-src.tar.gz
```

Unpack the connector:

```
# gzcat tomcat-connectors-1.2.18-src.tar.gz | tar xvf –
```

Change directories into the native directory and run the `buildconf.sh` script. You can safely ignore errors:

```
# chmod +x buildconf
# ./buildconf.sh
```

Once this is complete, build the JK module using the `configure` and make commands:

```
# ./configure –with-apxs=/opt/apache2/bin/apxs
# make
# make install
```

© 2006 McManus Consulting

You are now ready to test the entire setup.

Configuring Tomcat Examples for Apache

The following section configures Apache and Tomcat to communicate with each other. You can test the link between the two by serving and executing the example servlets provided with the Tomcat distribution. By the end of this exercise, you will be able to access example servlets by having Apache forward requests to Tomcat.

Edit the Tomcat server.xml configuration file to include the sample Tomcat/Apache classes. Locate the main “Server” container (that uses port 8005) and the “Host” container and add the following line underneath the top of each:

```
# cd /apps/BOXIR2/bobje/jakarta-tomcat-5.0.27/conf
# vi server.xml

<Server port="8005" shutdown="SHUTDOWN" debug="0">
 <Listener className="org.apache.jk.config.ApacheConfig"
modJk="/opt/apache2/modules/mod_jk.so" />


<Host name="localhost" debug="0" appBase="webapps" unpackWARs="true" autoDeploy="true"
xmlValidation="false" xmlNamespaceAware="false">

<Listener className="org.apache.jk.config.ApacheConfig" append="true" forwardAll="false"
modJk="/opt/a pache2/modules/mod_jk.so" />
```

Create a `workers.properties` file for the JK module.

```
# mkdir jk
# cd jk
# vi workers.properties
# BEGIN workers.properties
worker.list=ajp13
worker.ajp13.port=8009
# change this line to match apache ServerName and Host name in server.xml
worker.ajp13.host=localhost
worker.ajp13.type=ajp13
# END workers.properties
```

Modify the Apache configuration file to include the JK configuration file called `mod_jk.conf`. The Tomcat server creates this file automatically and it does not need to be edited or created to use the Tomcat example servlets. In the event that, other web applications will be used, this file will need to be modified to reflect those changes.

Add the bold line to the bottom of the Apache `httpd.conf` file:

```
# cd /opt/apache2/conf
# vi httpd.conf

Include /apps/BOXIR2/bobje/jakarta-tomcat-5.0.27/conf/auto/mod_jk.conf
```

Finally, edit both the httpd.conf and the server.xml files to reflect the IP address of the Apache/Tomcat server:

```
# cd /opt/apache2/conf
# vi httpd.conf
```

CHANGE:

```
#ServerName www.example.com:80
```

© 2006 McManus Consulting

TO:

```
ServerName 172.16.169.128:80


# cd /apps/BOXIR2/bobje/jakarta-tomcat-5.0.27/conf/
# vi server.xml
```

CHANGE:

```
<Host name="localhost" debug="0" appBase="webapps"
```

TO:

```
<Host name="172.16.169.128" debug="0" appBase="webapps"
```

Reset the permissions on both the Apache and Tomcat directories to the `bobjer2` user permissions:

```
# chown –R bobjer2:other /apps/BOXIR2/bobje/Jakarta-tomcat-5.0.27/conf/
# chown –R bobjer2:other /opt/apache2
```

Starting the Apache and Tomcat Servers

The servers are now ready to be tested. The Apache web server must be started as root due to the fact that it runs on privileged ports 80 and 443. The Tomcat server must be run as the user bobjer2.

Start the Tomcat server:

```
# su – bobjer2
$ cd /apps/BOXIR2/bobje/jakarta-tomcat-5.0.27/bin
$ ./startup.sh
```

Wait about 20 seconds until Tomcat has finished starting then start Apache:

```
$ exit
# cd /opt/apache2/bin
# ./apachectl start
```

Open a web browser and attempt to connect to both `http://172.16.169.128:8080/servlets-examples` and `http://172.16.169.128/servlets-examples`. If you can connect to both sites and get the example servlets page, then you have configured Apache to successfully forward requests to Tomcat.

Configuring Apache to Forward Port 443 to Tomcat

It is possible to forward encrypted requests over https (port 443) to the Tomcat server. All of the configuration required happens to the Apache web server. No special configuration of the Tomcat server is required in order to make this happen.

To configure Apache for SSL, first create the certificate directories:

```
# cd /opt/apache2/conf
# mkdir ssl.crt ssl.key
```

© 2006 McManus Consulting

Create test certificates. This document demonstrates how to make a certificate for testing purposes. This certificate is not signed by any certificate authority and is not intended for any production use. You will be prompted to enter your identifying information on the certificate:

```
# openssl req -new -x509 -nodes -out /opt/apache2/conf/ssl.crt/server.crt \
> -keyout /opt/apache2/conf/ssl.key/server.key
```

In order to accommodate both ports 80 and 443, you must create Apache `VirtualHost` directives in the `mod_jk.conf` file. Since this file is overwritten every time the Tomcat server starts, you must copy it out of its default directory and over to the Apache directory. This way, the file will remain static.

```
# cd /apps/BOXIR2/bobje/jarkata*/conf/auto
# cp mod_jk.conf /opt/apache2/conf
```

The easiest way to create an SSL virtual host for the JK connector is to copy and modify the `mod_jk.conf` file:

```
# cd /opt/apache2/conf
# cp mod_jk.conf mod_jk-ssl.conf
```

© 2006 McManus Consulting

Edit the `mod_jk-ssl.conf`:

```
# vi mod_jk-ssl.conf
```

REMOVE:

```
########## Auto generated on Mon Sep 04 12:21:10 CDT 2006##########

# Load the Tomcat Connector

<IfModule !mod_jk.c>
  LoadModule jk_module "/opt/apache2/modules/mod_jk.so"
</IfModule>

JkWorkersFile "confnf/workers.properties"
JkLogFile "logs/mod_jk.log"

JkLogLevel emerg
```

CHANGE:

```
<VirtualHost 172.16.169.128>
ServerName 172.16.169.128
```

TO:

```
<VirtualHost 172.16.169.128:443>
#    ServerName 172.16.169.128
```

ADD:

```
<VirtualHost 172.16.169.128:443>
#    ServerName 172.16.169.128


SSLEngine On
SSLCipherSuite ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP:+eNULL
SSLCertificateFile /opt/apache2/conf/ssl.crt/server.crt
SSLCertificateKeyFile /opt/apache2/conf/ssl.key/server.key
```

Once completed, append the contents of the `mod_jk-ssl.conf` file to the end of the `mod_jk.conf` file:

```
# cat mod_jk-ssl.conf >> mod_jk.conf
```

© 2006 McManus Consulting

Edit the Apache configuration file to reflect the new `mod_jk.conf` file:

```
# vi httpd.conf
```

CHANGE:

```
Include /apps/BOXIR2/bobje/jakarta-tomcat-5.0.27/conf/auto/mod_jk.conf
```

TO:

```
Include conf/mod_jk.conf
```

Stop both Tomcat and Apache and restart them with SSL enabled:

```
# su – bobjer2
$ cd /apps/BOXIR2/bobje/jakarta-tomcat-5.0.27/bin
$ ./shutdown.sh
```

Wait about 20 seconds until Tomcat has finished starting then stop Apache:

```
$ exit
# cd /opt/apache2/bin
# ./apachectl stop
```

Restart Tomcat:

```
# su – bobjer2
$ cd /apps/BOXIR2/bobje/jakarta-tomcat-5.0.27/bin
$ ./startup.sh
```

Wait about 20 seconds until Tomcat has finished starting then start Apache:

```
$ exit
# cd /opt/apache2/bin
# ./apachectl start
```

Test the following links in a web browser. Each should display the example servlet page for Tomcat:

```
http://172.16.169.128:8080/servlets-examples
http://172.16.169.128/servlets-examples
https://172.16.169.128/servlets-examples
```

© 2006 McManus Consulting