Running DSpace on Standard Ports

Running DSpace on Standard Ports (80 for http:// and 443 for https://)

Since it is not trivial to get a Java servlet container, such as Apache Tomcat to listen on the "standard" TCP ports for a web server, this page explains alternative ways to accomplish it.

What does "port" mean? A server program, like a web server, has to "listen" (i.e. tell the operating system it is offering a service; accepts incoming requests) at a *well-known port* so that clients, like your web browser, can find it. The combination of *host* and *port* uniquely identifies a service:
For example, the URL *http://dspace.mit.edu/* identifies a network service running on host *dspace.mit.edu* and port *80* (the default HTTP port).

It is desirable to implement DSpace on the *default* ports so you don't have to clutter your URLs with port numbers and try to get users to remember them.

The problem arises on Unix-based servers because the default, well-known, web server ports are in the range that require root (superuser) privileges to listen on. Since the DSpace server (a Java VM) should always be running as an unprivileged user, it cannot directly open these ports. It can only listen on higher-numbered ports. So, the solution is to run the JVM as an unprivileged user and find a way to accept HTTP requests on the standard ports and *redire ct* them to the higher-numbered ports.

When using Apache 2.4.2 (and lower) in front of a DSpace webapp deployed in Tomcat, mod_proxy_ajp and possibly mod_proxy_http breaks the connection to the back end (Tomcat) prematurely leading to response mixups. This is reported as bug CVE-2012-3502 (http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2012-3502) of Apache and fixed in Apache 2.4.3 (see http://www.apache.org/dist/httpd/CHANGES_2.4). The 2.2.x branch hasn't shown this problem only the 2.4.x branch has.

See Also

- SecuringDspace – appropriate security for a DSpace server.

Method 1 - redirecting with IP tables

This is known to work on Red Hat Enterprise Linux 3 and other similar versions of GNU/Linux. Use the *iptables* utility to redirect requests on the default ports to the higher-numbered ports where the DSpace servlet container is actually listening.

This allows you to use a pure Java servlet container such as Jetty or Apache Tomcat as the actual web server.

See your system's documentation for the *iptables* and *iptables-save* commands for more information.

Example: Route HTTP (port 80) to port 8080 and HTTPS (port 443) to port 8443:

```
/sbin/iptables -t nat -I PREROUTING -p tcp --destination-port 80 -j REDIRECT --to-ports 8080
/sbin/iptables -t nat -A OUTPUT -p tcp -d _[server_ip_address|server_ip_address]_ --dport 80 -j  REDIRECT --to-port 8080
/sbin/iptables -t nat -I PREROUTING -p tcp --destination-port 443 -j REDIRECT --to-ports 8443
```

Then write the current state of the tables to a configuration file so they are automatically restored to this state after the system is booted:

```
/sbin/iptables-save > /etc/sysconfig/iptables
```

Configure your Java Servlet container to listen to ports 8080 and 8443 for HTTP and HTTPS connections, respectively.

Method 1b - Write your own iptables config file

Here is an expanded example:
edit the */etc/sysconfig/iptables* file (make a backup of this file first!)

Open the standard ports 80 and 443 and the redirect ports 7780 and 7781 in this example inside the *filter block of statements followed by the redirect statements in the *nat block of statements (nat stands for network address translation) ... here is an example of that file (redhat WS3, a 2.4 linux kernel is requried, consult the excellent HOWTOS Documentation at http://www.netfilter.org)

```
# Firewall configuration written by redhat-config-securitylevel
# Manual customization of this file is not recommended.
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
:RH-Firewall-1-INPUT - [0:0]
-A INPUT -j RH-Firewall-1-INPUT
-A FORWARD -j RH-Firewall-1-INPUT
-A RH-Firewall-1-INPUT -i lo -j ACCEPT
# trust eth1 for heartbeat
-A RH-Firewall-1-INPUT -i eth1 -j ACCEPT
-A RH-Firewall-1-INPUT -p icmp --icmp-type any -j ACCEPT
-A RH-Firewall-1-INPUT -p 50 -j ACCEPT
-A RH-Firewall-1-INPUT -p 51 -j ACCEPT
-A RH-Firewall-1-INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A RH-Firewall-1-INPUT -m state --state NEW -m tcp -p tcp --dport 22 -j ACCEPT
# dspace ports
-A RH-Firewall-1-INPUT -m state --state NEW -d xxx.xxx.xxx.xxx -m tcp -p tcp --dport 80 -j ACCEPT
-A RH-Firewall-1-INPUT -m state --state NEW -d xxx.xxx.xxx.xxx -m tcp -p tcp --dport 443 -j ACCEPT
-A RH-Firewall-1-INPUT -m state --state NEW -d xxx.xxx.xxx.xxx -m tcp -p tcp --dport 7780 -j ACCEPT
-A RH-Firewall-1-INPUT -m state --state NEW -d xxx.xxx.xxx.xxx -m tcp -p tcp --dport 7781 -j ACCEPT
-A RH-Firewall-1-INPUT -j REJECT --reject-with icmp-host-prohibited
COMMIT
*nat
-A PREROUTING -d xxx.xxx.xxx.xxx -p tcp --dport 80 -j DNAT --to xxx.xxx.xxx.xxx:7780
-A PREROUTING -d xxx.xxx.xxx.xxx -p tcp --dport 443 -j DNAT --to xxx.xxx.xxx.xxx:7781
COMMIT
```

Restart the network

```
/etc/init.d/iptables restart
```

Restart Tomcat as non-root user on port 7780/7781 (edit your conf/server.xml file for this)

```
su - dspace -c "/usr/local/jakarta-tomcat-5.0.27/bin/startup.sh"
```

Load the URL for port 80 which will get redirect to your tomcat on port 7780

And I'll just add this here on how to get tomcat running in ssl mode:

- create a directory like /usr/local/jakarta-tomcat-5.0.27/conf/ssl.new
- copy or link your insitution server.key and server.crt files in this directory
- execute this command to create the keystore file, this will prompt you for a password you must supply "changeit" without the quotes

```
/usr/bin/openssl pkcs12 -export -out keystore.pkcs12 -in ./server.crt -inkey ./server.key
```

* Then change the ownership on

```
keystore.pkcs12
```

to the user which runs the tomcat server

```
chown dspace:dspace  keystore.pkcs12
```

* Next edit your conf/server.xml file and edit the section for the 8443 connector and add these 3 lines (be sure to add them inside the block of statements that starts with <Connector port="7781" ... and ends with ... />, you may have to uncomment that entire block of statements too since by default these connector statements are wrapped inside tags)

```
keystoreFile="/usr/local/jakarta-tomcat-5.0.27/conf/ssl.new/keystore.pkcs12"
keystoreType="PKCS12"
keystorePass="changeit"
```

* Restart tomcat

Method 2a - Use Apache HTTPD (mod_proxy_ajp) + Tomcat (port 8009)

This assumes you have the following working:

1. httpd is running and listening on port 80
2. tomcat is running and listening on port 8009,8080

By convention, web-servers listen on port 80 to deliver content such as static html files. So that web browsers can

```
## use the more familiar url
 http://www.dspace-instance.org

 ## instead of....
 http://www.dspace-instance.org:8080/jspui
 http://www.dspace-instance.org:8080/xmlui
```

- Set Tomcat to serve up DSpace by default

```
cd /usr/local/tomcat/webapps
mv ROOT ROOT_hold
ln -s /dspace/webapps/jspui ROOT
## for the Manakin interface replace jspui with xmlui
```

* Configure /etc/httpd/conf.d/ssl.conf or proxy_ajp.conf

```
#
# Put in VirtualHost element
#
ProxyPass  /do_not_touch  !
ProxyPass  /  ajp://localhost:8009/
ProxyPassReverse  /  ajp://localhost:8009/
```

Note:

1. You'll need to reload or restart the httpd service
2. The "!" <bang> sets Apache web-server to NOT REDIRECT everything under /do_not_touch

Method 2b - use Apache HTTPD / Tomcat connector (mod_jk)

Run Apache HTTPD as a front-end for Tomcat, see
the mod_jk wiki page
This is tricky to set up, but secure.

Method 3 - use Apache to redirect requests to Tomcat on port 8080

For RedHat LINUX server:

In /etc/httpd/conf/httpd.conf:

1. Ensure the following modules are listed under "# Dynamic Shared Object (DSO) Support", this is a list showing all modules loaded by Apache:

```
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_http_module modules/mod_proxy_http.so
```

2. Add the following lines after the <tt><Location></tt> context:
(Note: our dspace url is: http://rose.bris.ac.uk)

```
ProxyPass / [http://rose.bris.ac.uk:8080/]
ProxyPassReverse / [http://rose.bris.ac.uk:8080/]
```

3. Ensure Server Name is also set:

```
ServerName rose.bris.ac.uk
```

4. Restart Apache:

```
/usr/sbin/apachectl stop
/usr/sbin/apachectl start
```

In order to display the dspace home page, not the tomcat home page when accessing http://rose.bris.ac.uk

1. Create a page called HelloWorld.jsp in the directory below:

```
$CATALINA_HOME/webapps/ROOT/
```

2. Add the following lines to the HelloWorld.jsp:

```
<% response.sendRedirect("/dspace/"); %>
```

Next. In the same directory, add the following to index.jsp just after the <body> tag:-

```
<body>
<jsp:forward page="HelloWorld.jsp"/>
```

3. Edit: ../ROOT/WEB-INF/web.xml, disable the following text as follows:

```
<!-- JSPC servlet mappings start
<servlet>
<servlet-name>org.apache.jsp.index_jsp</servlet-name>
<servlet-class>org.apache.jsp.index_jsp</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>org.apache.jsp.index_jsp</servlet-name>
<url-pattern>/index.jsp</url-pattern>
</servlet-mapping>
-->
```

4. Re-start tomcat

Method 4 - Use Tomcat's jsvc daemon

Note: Only works for Unix like platforms.
This documentation is based on Solaris 10 using the Solaris Service Management Facility.
See http://jakarta.apache.org/commons/daemon/jsvc.html
for more details

1. Download and install Tomcat from jakarta.apache.org
2. Compile jsvc
    - export JAVA_HOME=/usr/java
    - cd $CATALINA_HOME/bin
    - gunzip jsvc.tar.gz && tar -xvf jsvc.tar
    - ./configure
    - gmake
    - cp jsvc ..
    - cd ..

- chown tomcat:tomcat jsvc
- rm -rf jsvc-src

3. Setup your Service Management Facility process
    a. create the following file: /lib/svc/method/dspace

```
#!/bin/sh
. /lib/svc/share/smf_include.sh
JAVA_HOME=/usr/java
JAVA_OPTS="-Xmx512M -Xms64M -Dfile.encoding=UTF-8"
CATALINA_HOME=<where tomcat is installed>
DAEMON_HOME=<where tomcat is installed>
TOMCAT_USER=tomcat
TMP_DIR=/var/tmp
PID_FILE=/var/run/txspace.pid
CATALINA_BASE=<where tomcat is installed>
CLASSPATH=$JAVA_HOME/lib/tools.jar:$CATALINA_HOME/bin/commons-daemon.jar:$CATALINA_HOME/bin
/bootstrap.jar
start()
{
$DAEMON_HOME/bin/jsvc -user $TOMCAT_USER -home $JAVA_HOME -Dcatalina.home=$CATALINA_HOME \
-Dcatalina.base=$CATALINA_BASE -Djava.io.tmpdir=$TMP_DIR -wait 10 -pidfile $PID_FILE \
-outfile $CATALINA_BASE/logs/catalina.out -errfile '&1' -Xms256m -Xms1024m \
-cp $CLASSPATH org.apache.catalina.startup.Bootstrap
}
stop()
{
$DAEMON_HOME/bin/jsvc -stop -pidfile $PID_FILE org.apache.catalina.startup.Bootstrap
}
case "$1" in
'start')
echo "Starting Tomcat"
start
;;
'stop')
echo "Stopping Tomcat"
stop
;;
'refresh')
echo "Restarting Tomcat"
stop
start
;;
*)
echo "Usage tomcat.sh start/stop/refresh"
exit 1;;
esac
exit $SMF_EXIT_OK
```

    b. Create the profile: /var/svc/manifest/application/dspace.xml

```
<?xml version="1.0"?>
<!DOCTYPE service_bundle SYSTEM "/usr/share/lib/xml/dtd/service_bundle.dtd.1">
<service_bundle type='manifest' name='dspace'>
<service name='application/dspace' type='service' version='1'>

<create_default_instance enabled='true'/>
<single_instance/>

<dependency name='loopback' grouping='require_all'
restart_on='error' type='service'>
<service_fmri value='svc:/network/loopback:default'/>
</dependency>
<dependency name='physical' grouping='require_all'
restart_on='error' type='service'>
<service_fmri value='svc:/network/physical:default'/>
</dependency>
<exec_method type='method' name='start'
exec='/lib/svc/method/txspace start'
timeout_seconds='60' />
<exec_method type='method' name='stop'
exec='/lib/svc/method/txspace stop'
timeout_seconds='5' />
<exec_method type='method' name='refresh'
exec='/lib/svc/method/txspace refresh'
timeout_seconds='5' />
<stability value='Unstable' />
<template>
<common_name>
<loctext xml:lang='C'>DSpace</loctext>
</common_name>
<documentation>
<manpage title='dspace' manpath='/opt/apps/man' section='8' />
<doc_link name='tomcat.apache.org'
uri='http://tomcat.apache.org' />
</documentation>
</template>
</service>
</service_bundle>
</code>
```

3. Now import the profile

```
/usr/sbin/svccfg import /var/svc/manifest/application/dspace.xml
```

4. Modify Tomcat's Connector in server.xml

```
<Connector className="org.apache.catalina.connector.http.HttpConnector"
port="80" minProcessors="5" maxProcessors="75"
enableLookups="true" redirectPort="8443"
acceptCount="10" debug="0" connectionTimeout="60000"
address="192.168.0.1"/>
```

5. Enable Tomcat

```
/usr/sbin/svcadm enable dspace
```

Method 5 - Tomcat on low ports natively with authbind

See:

- DSpace on Ubuntu: Enable authbind for Tomcat
- Tomcat 6: Binding to a Privileged Port on Debian/Ubuntu