Introduction

The aim of this document is to provide information on the Remedy Midtier 6.3, from installing and upgrading, to basic trouble shooting. Some of the information may need tailoring for your specific environment, given every Unix installation is generally bespoke in some way. The document also covers basic Apache/Tomcat installation and configuration steps.

Installing or upgrading a Remedy Midtier

This guide should provide someone with Unix skills enough detail to install or upgrade the Remedy Midtier. We assume you are using Apache Tomcat, and that the version installed is supported on the Remedy website. The location of the Tomcat installation is specified as [tomcat]. This procedure does not use the Remedy installation files because they do not work with Tomcat and are not really suitable for a bespoke Apache/Tomcat installation.

1. Download latest Midtier 'patch' file as a war.
For example, the URL on the Remedy support website will look similar to this:

http://supportweb.remedy.com/patches/ars/6.3/patch10/14/midtier/war

The support page will most probably link to a level above this, so browse to the midtier directory and look for a set of midtier_arch.war files (where arch is Windows, Solaris, Linux, etc.).

2. Copy the war file onto the Midtier machine and place it at any location, such as /tmp.

3. Stop Tomcat. i.e.

```
[tomcat]/bin/shutdown.sh
```

4. If you are upgrading, copy the file [tomcat]/webapps/arsys/WEB-INF/classes/config.properties
into a safe location, such as [tomcat]/. i.e.

```
cp [tomcat]/webapps/arsys/WEB-INF/classes/config.properties  [tomcat]/
```

5. Move the directory [tomcat]/webapps/arsys to [tomcat]/arsysold. i.e.

```
mv [tomcat]/webapps/arsys  [tomcat]/arsysold
```

6. Copy the new war file into the directory [tomcat]/webapps/
and **ensure that you call it arsys.war**. i.e. :

```
cp /tmp/midtier_solaris.war  [tomcat]/webapps/arsys.war
```

If this is a new installation, go to step 9.
If this is an upgrade then follow steps 7 and 8 too.

7. Start Tomcat. Wait a few minutes while Tomcat unpacks the new war file. You should now see a new directory [tomcat]/webapps/arsys. After a minute, stop Tomcat. i.e.

```
[tomcat]/bin/startup.sh
```

 # Wait a minute

```
[tomcat]/bin/shutdown.sh
```

8. Copy the config.properties file you backed up earlier into the new Midtier installation's WEB-INF/classses directory, i.e.

```
cp [tomcat]/config.properties  [tomcat]/webapps/arsys/WEB-INF/classes
```

9. Edit the Tomcat catalina.sh file (found in [tomcat]/bin) with your favourite text editor, and add the following line to the top:

```
JAVA_OPTS="-Djava.awt.headless=true"
```

This instructs the Java Virtual Machine that there is no display attached to the machine (and as such, the application is a 'server side' application). To ensure you've set this correctly, view a Flashboard once the installation is complete.

10. Start Tomcat, i.e.

```
[tomcat]/bin/startup.sh
```

11. Login to the configuration page on the browser and check it works! i.e.

http://hostname:8080/arsys/shared/config/config.jsp

or

http://hostname/arsys/shared/config/config.jsp

etc.

*(The default password is '**arsystem**'.)*

12. Check the configuration is still as desired and then take a backup of the config.properties file for future reference. i.e.:

```
cp [tomcat]/webapps/arsys/WEB-INF/classes  [tomcat]/config.properties
```

Modifying a Tomcat installation so it can run the Remedy Midtier

After installing a new version of Tomcat, a small change is required to the start up script.

The technical detail of this change involves informing the Java Virtual Machine where to find the Remedy AR API.

Unix users

To do this, open the file **[tomcat]/bin/catalina.sh** and insert the lines:

```
LD_LIBRARY_PATH=[tomcat]/webapps/arsys/WEB-INF/lib
export LD_LIBRARY_PATH
```

into the second line of the file.

Windows users

Find the Remedy AR API DLLs (located in the arsys/WEB-INF/lib directory) and copy them all into your c:\windows\system32 directory. This is a little messy, but the most reliable way I've ever found of ensuring it will work on Windows. Sadly, Remedy Engineering refuse to accept the need for a Java API call to actually set the location of the libraries before they are invoked, and hence, you can not 'point' the JVM at them in a sensible fashion.

Logging

The Tomcat logs are not written to the same place as the Midtier logs, however both sets are useful for debugging. The Tomcat logs are written into [tomcat]/logs, while the location of the Midtier logs are configured through the Midtier configuration tool.

Common problems

LD_LIBRARY_PATH not set correctly

If you try to login and receive an error page (of some kind), and the current log file in [tomcat]/logs contains the following:

```
----- Root Cause -----
java.lang.UnsatisfiedLinkError: no arjni63 in java.library.path
        at java.lang.ClassLoader.loadLibrary(ClassLoader.java:1491)
        at java.lang.Runtime.loadLibrary0(Runtime.java:788)
        at java.lang.System.loadLibrary(System.java:834)
        at com.remedy.arsys.api.Proxy.<clinit>(Proxy.java:51)
        at com.remedy.arsys.api.ProxyPool.get(ProxyPool.java:62)
```

Then the Midtier has been unable to find the AR native libraries. These are a set of shared libraries stored in [tomcat]/webapps/arsys/WEB-INF/lib. To correct this fault, ensure that you have followed the steps detailed in **'Modifying a Tomcat installation so it can run the Remedy Midtier'**.

Midtier configuration 'disappeared'

You have forgotten to put the copy of [tomcat]/webapps/arsys/WEB-INF/classes/config.properties file back into the Midtier installation after Tomcat has unpacked the new war file. To do this, stop Tomcat, put your copy of config.properties into the correct location, and start Tomcat. i.e.

```
[tomcat]/bin/shutdown.sh

cp [tomcat]/config.properties    [tomcat]/webapps/arsys/WEB-INF/classes

[tomcat]/bin/startup.sh
```

JAVA_HOME can not be found

A common problem that occurs when operating Tomcat is the JAVA_HOME environment variable not being set. You will see the following:

```
The JAVA_HOME environment variable is not defined
This environment variable is needed to run this program
```

To rectify this, you will need to set the JAVA_HOME environment variable. Assuming your Java installation is in the directory /usr/local/java, then you will need to type the following:

```
export JAVA_HOME=/usr/local/java
```

Here is an example of this problem occurring and being rectified:

```
root@myserver # bin/catalina.sh  start
The JAVA_HOME environment variable is not defined
This environment variable is needed to run this program

root@myserver # export JAVA_HOME=/usr/local/java
root@myserver # bin/catalina.sh  start
Using CATALINA_BASE:  /usr/local/jakarta-tomcat-4.1.30
Using CATALINA_HOME:  /usr/local/jakarta-tomcat-4.1.30
Using CATALINA_TMPDIR: /usr/local/jakarta-tomcat-4.1.30/temp
Using JAVA_HOME:      /usr/local/java
```

Testing Tomcat without Apache

You may come across a situation where the webserver returns nothing but an error page. When using Apache in front of Tomcat, the problem could lie in Apache, Tomcat or the Midtier configuration.

Tomcat comes with a 'standalone HTTP connector' configured to be operational by default. This means you can connect to Tomcat, without going through Apache, and test the system.

The standalone connector runs on port 8080 and you can connect to it by using a URL in this format:

http://host**:8080**/arsys/home

Where the important addition is the port number (:8080).

By connecting in this fashion you can take Apache out of the equation and see if Tomcat and Midtier are running correctly. If the URL returns the desired page, the problem most probably lies in the Apache configuration. If you still see the error then you need to check the Tomcat and Midtier configuration, however assuming an out of the box Tomcat installation, the problem most likely lies in the Midtier configuration.

Installing Tomcat

Obtaining the packages

Apache Tomcat can be found on the Apache Jakarta website: http://jakarta.apache.org, but more specifically, you probably want to look here: http://tomcat.apache.org

I would advise you to always use the latest and greatest stable Tomcat build, and have seen Midtier 6.3 functioning with Tomcat 5.5.x. Tomcat 5.5 is designed for the Java 1.5 VM and requires a 'compatibility package' if you are using a Java 1.4 VM (no-one should be using a pre-1.4.2 VM). When you download Tomcat for your platform, do not forget to download the compatibility package if required!

Installing Tomcat is very straight forward. If you're using Windows, you will find a Windows installer. If you're using Unix, simply untar the package into a directory. Also, Unix users will (probably) have to set the CATALINA_HOME and JAVA_HOME environment variables, and the most sensible location to do this will (probably) be in the /etc/profile file.

The only bespoke step is telling Tomcat where to find the AR API shared libraries (given the Remedy Java API is actually nothing more than a C library with a Java wrapper). To do this, read the section titled *"Modifying a Tomcat installation so it can run the Remedy Midtier"*.

Installing Apache2 for Solaris

Obtaining the packages

Apache2 packages can be found at http://www.sunfreeware.com, and you require software for your Solaris platform. You are currently running Sun OS 5.9 which is also known as Solaris 9. Be sure to download not only the Apache2 package, but all required packages. After transferring them to the Solaris machine, you can install them with the pkgadd command. However you must un(g)zip them first. For example:

```
gunzip packagename.gz
pkgadd –d ./packagename
```

Currently, Apache installs into the /usr/local/apache2 directory.

Configuring Apache, part 1

Change to the [apache]/conf directory and copy the create an httpd.conf file from the sample provided.

```
cd /usr/local/apache2/conf
cp httpd-std.conf httpd.conf
```

Now check Apache runs correctly. Use the following command to start it:

```
/usr/local/apache2/bin/apachectl start
```

Open up a web browser and point your browser at the machine in question. If you see a welcome page, the installation was successful. If you do not, read the logs in [apache]/logs. You may stop apache by using the stop (instead of start) parameter to apachectl.

Once Apache has been successfully installed and can be seen to run, you need to install a Tomcat connector called **mod_jk**. The purpose of mod_jk is to provide the bridge between Apache and Tomcat.

Building the mod_jk module

Before you start this step, check to see if you have a compiler. Simply type gcc at a Unix command prompt and see if the program exists. If it does not, go to Sunfreeware and download a copy. Install it using the pkgadd command, detailed in an example above. If you do not have a compiler (gcc), you'll also need to ensure you have the make package.

Mod_jk is available in binary form, however modules will often not work on your version of Solaris and it's therefore best to build your own. You may obtain the latest module from this URL: http://www.apache.org/dist/jakarta/tomcat-connectors/jk/source. Be sure to read the mod_jk homepage (http://jakarta.apache.org/tomcat/connectors-doc/) for up to date news and information on what module version is the most recent.

Once you've downloaded the source, copy it to your server, unpack and compile the module. The steps are currently as follows:

```
gunzip source.gz
tar xf source.gz
cd source/jk/native
./configure –with-apxs=/usr/local/apache2/bin/apxs
make
make install
```

You will note that the configuration process requires us to pass the location of the apxs file within the Apache bin directory.

While building the module, I made the following observations:

- The [apache]/build/libtool script was looking in /usr/local/bin for the sed utility, yet sed was installed in /usr/bin. Therefore, if you type 'which sed' and find sed is not installed in /usr/local/bin, you will have to edit the libtool script and replace all references to /usr/local/bin/sed with the correct location.

- The ar command was not in the path, which broke the make process. If you type ar and find the command is not found, ensure it is added to the path. It will probably be found under /usr/ccs/bin, therefore type: **export PATH=$PATH:/usr/ccs/bin**

- The apache apxs script was looking for perl in /usr/local/bin yet it's installed in /usr/bin. Therefore, open up the script in your favourite text editor and change the first line to: **#!/usr/bin/perl -w**

Once the build process has completed, the module should be installed at the following location: **[apache]/modules/mod_jk.so**.

Configuring Apache, part 2

Apache requires configuration directives to make use of mod_jk, however Tomcat will create this for us if we enable a plugin. For Tomcat 4.1.x. (the version we are using), open the [tomcat]/conf/server.xml file in your favourite text editor and follow these steps:

1. Insert the following directive after the <server> tag (ensure you type modJk as the attribute is case sensitive):

<Listener className="org.apache.ajp.tomcat4.config.ApacheConfig"
modJk="/usr/local/apache2/modules/mod_jk.so"
/>

2. Insert the following directive after each <host> tag:

<Listener className="org.apache.ajp.tomcat4.config.ApacheConfig"
append="true"
/>

If your server will use name based virtual hosts (i.e. you use multiple hostnames to connect to your webserver), further work is required. In order for the Apache virtual host configuration to be correctly generated, you must edit the server.xml file again. Follow these steps:

1. Find the <Engine> tag and look for the defaultHost attribute. Change this value to the IP address of the machine. I.e.:

<Engine name="Standalone" defaultHost="192.168.0.10" debug="0">

2. Find the <Host> tag and look for the name attribute. Change the value of this to the IP adderss of the machine. I.e.:

<Host name="192.168.0.10" debug="0" appBase="webapps" unpackWARs="true" autoDeploy="true">

3. Now add each hostname in an alias after the Host tag, i.e.:

<Host name="192.168.0.10" debug="0" appBase="webapps" unpackWARs="true" autoDeploy="true">
<Alias>midtier</Alias>
<Alias>remedy</Alias>

More detailed information on this process can be found at this URL: http://jakarta.apache.org/tomcat/tomcat-4.1-doc/config/host.html#Host%20Name%20Aliases

You may wonder why we are setting the IP addresses of the engine and host in the server.xml file, and this is so the configuration file generated by Tomcat has the correct Apache Virtualhost directive, i.e.: <Virtualhost IP> as opposed to <Virtualhost localhost>.

Now restart Tomcat and you should find the following file is created:

[tomcat]/conf/auto/mod_jk.conf

The final configurational step is to tell Apache about the automatically generated mod_jk.conf file. To do this, append the following to the httpd.conf file:

Include /usr/local/jakarta-tomcat-4.1.30/conf/auto/mod_jk.conf

You can quickly do this using the following command:

```
echo “Include /usr/local/jakarta-tomcat-4.1.30/conf/auto/mod_jk.conf” >> /usr/local/apache2/conf/httpd.conf
```

Testing the installation

In summary, you should have now:

- Installed Apache2 and created the httpd.conf file from the template.
- Built the mod_jk module.
- Configured Tomcat to create the mod_jk.include file.
- Appended the Include directive to the Apache httpd.conf file.

Now you should be able to restart both Apache and Tomcat, and point your browser at the Midtier installation running behind Apache. To do this, open a browser and type: http://server/arsys.

If you encounter problems with Apache failing to start, look at the logs. These are inside the [apache]/logs directory, and ensure you read **error_log**.

Further reading

Tomcat Connectors (aka mod_jk): http://jakarta.apache.org/tomcat/connectors-doc/
Apache2 Documentation: http://httpd.apache.org/docs/2.0/