Apendix

Table of Contents

Java Development Kit (): Installing, Configuring and Running

The [app]` Platform` is written in Java; therefore, before running any `server, you must have a working Java Runtime Environment () or Java Development Kit () installed on your system. In addition, the JRE or JDK you are using to run [app]` must be version 5 or higher [1: At this point in time, it is possible to run most servers, such as the JAIN SLEE Server, using a Java 6 JRE or JDK. Be aware, however, that presently the XML Document Management Server does not run on Java 6. We suggest checking the web site, forums or discussion pages if you need to inquire about the status of running the XML Document Management Server with Java 6.].

JRE versus JDK - 32-Bit versus 64-Bit

Should I Install the JRE or JDK?

Although you can run `servers using the Java Runtime Environment, we assume that most users are developers interested in developing Java-based, [app]`-driven solutions. Therefore, in this guide we take the tact of showing how to install the full Java Development Kit.

Should I Install the 32-Bit or the 64-Bit JDK, and Does It Matter?

Briefly stated: if you are running on a 64-Bit Linux or Windows platform, you should consider installing and running the 64-bit JDK over the 32-bit one. Here are some heuristics for determining whether you would rather run the 64-bit Java Virtual Machine (JVM) over its 32-bit cousin for your application:

- Wider datapath: the pipe between RAM and CPU is doubled, which improves the performance of memory-bound applications when using a 64-bit JVM.
- 64-bit memory addressing gives virtually unlimited (1 exabyte) heap allocation. However large heaps affect garbage collection.
- Applications that run with more than 1.5 GB of RAM (including free space for garbage collection optimization) should utilize the 64-bit JVM.
- Applications that run on a 32-bit JVM and do not require more than minimal heap sizes will gain nothing from a 64-bit JVM. Barring memory issues, 64-bit hardware with the same relative clock speed and architecture is not likely to run Java applications faster than their 32-bit cousin.

Note that the following instructions detail how to download and install the 32-bit JDK, although the steps are nearly identical for installing the 64-bit version.

Downloading JDK

You can download the Sun JDK 5.0 (Java 2 Development Kit) from Sun's website: http://java.sun.com/javase/downloads/index_jdk5.jsp. Click on the Download link next to "JDK 5.0 Update `<x>`" (where `[replaceable]<x>` is the latest minor version release number). On the next page, select your language and platform (both architecture—whether 32- or 64-bit—and operating system), read and agree to the `Java Development Kit 5.0 License Agreement`, and proceed to the download page.

The Sun website will present two download alternatives to you: one is an RPM inside a self-extracting file (for example, *jdk-1_5_0_16-linux-i586-rpm.bin*), and the other is merely a self-extracting file (e.g. *jdk-1_5_0_16-linux-i586.bin*). If you are installing the JDK on Red Hat Enterprise Linux, Fedora, or another RPM-based Linux system, we suggest that you download the self-extracting file containing the RPM package, which will set up and use the SysV service scripts in addition to installing the JDK. We also suggest installing the self-extracting RPM file if you will be running [app]``` in a production environment.

Installing

The following procedures detail how to install the Java Development Kit on both Linux and Windows.

Procedure: Installing the JDK on Linux

1. Regardless of which file you downloaded, you can install it on Linux by simply making sure the file is executable and then running it:

```
~]$ chmod +x "jdk-1_5_0_<minor_version>-linux-<architecture>-rpm.bin"
~]$ ./"jdk-1_5_0_<minor_version>-linux-<architecture>-rpm.bin"
```

Moving from Non-RPM Installer to SysV Service Scripts

If you download the non-RPM self-extracting file (and installed it), and you are running on an RPM-based system, you can still set up the SysV service scripts by downloading and installing one of the `-compat` packages from the JPackage project. Remember to download the `-compat` package which corresponds correctly to the minor release number of the JDK you installed. The compat packages are available from link:ftp://jpackage.hmdc.harvard.edu/JPackage/1.7/generic/RPMS.non-free/.

You do not need to install a `-compat` package in addition to the JDK if you installed the self-extracting RPM file! The `-compat` package merely performs the same SysV service script set up that the RPM version of the JDK installer does.

Installing JDK on Windows

1. Using Explorer, simply double-click the downloaded self-extracting installer and follow the instructions to install the JDK.

Configuring

Configuring your system for the JDK consists in two tasks: setting the `JAVA_HOME` environment variable, and ensuring that the system is using the proper JDK (or JRE) using the `alternatives` command. Setting `JAVA_HOME` usually overrides the values for `java`, `javac` and `java_sdk_1.5.0` in `alternatives`, but we will set them all just to be safe and consistent.

Setting Linux JAVA_HOME Environment Variables

Setting the JAVA_HOME Environment Variable on Generic Linux

After installing the JDK, you must ensure that the JAVA_HOME environment variable exists and points to the location of your JDK installation.

Setting the Correct Java Version

Setting java, javac and java_sdk_1.5.0 Using the alternatives command

As the root user, call /usr/sbin/alternatives with the --config java option to select between JDKs and JREs installed on your system:

Setting JAVA_HOME Environment Variables on Windows

Setting the JAVA_HOME Environment Variable on Windows

For information on how to set environment variables in Windows, refer to http://support.microsoft.com/kb/931715.

Uninstalling JDK on Linux and Windows

Uninstalling

There is usually no reason (other than space concerns) to remove a particular JDK from your system, given that you can switch between JDKs and JREs easily using alternatives, and/or by setting JAVA_HOME.

Uninstalling the JDK on Linux

On RPM-based systems, you can uninstall the JDK using the `yum remove <jdk_rpm_name> ` command.

Uninstalling the JDK on Windows

On Windows systems, check the JDK entry in the Start menu for an uninstall command, or use Add/Remove Programs.

Setting the JBOSS_HOME Environment Variable

The [app]` Platform` (`) is built on top of the [app]`JBoss Application Server (JBoss AS). You do not need to set the JBOSS_HOME environment variable to run any of the [app]` Platform` servers *unless* JBOSS_HOME is *already* set.

The best way to know for sure whether JBOSS_HOME was set previously or not is to perform a simple check which may save you time and frustration.

Checking to See If JBOSS_HOME is Set on Unix

At the command line, echo $JBOSS_HOME to see if it is currently defined in your environment:

```
~]$ echo $JBOSS_HOME
```

The [app]` Platform` and most &PLATFORM_NAME; servers are built on top of the JBoss Application Server (JBoss AS). When the [app]` Platform` or &PLATFORM_NAME; servers are built *from source*, then JBOSS_HOME *must* be set, because the &PLATFORM_NAME; files are installed into (or "over top of" if you prefer) a clean JBoss AS installation, and the build process assumes that the location pointed to by the JBOSS_HOME environment variable at the time of building is the JBoss AS installation into which you want it to install the &PLATFORM_NAME; files.

This guide does not detail building the [app]` Platform` or any &PLATFORM_NAME; servers from source. It is nevertheless useful to understand the role played by JBoss AS and JBOSS_HOME in the &PLATFORM_NAME; ecosystem.

The immediately-following section considers whether you need to set JBOSS_HOME at all and, if so, when. The subsequent sections detail how to set JBOSS_HOME on Unix and Windows

Even if you fall into the category below of *not needing* to set JBOSS_HOME, you may want to for various reasons anyway. Also, even if you are instructed that you do *not need* to set JBOSS_HOME, it is good practice nonetheless to check and make sure that JBOSS_HOME actually *isn't* set or defined on your system for some reason. This can save you both time and frustration.

You *DO NOT NEED* to set JBOSS_HOME if…

- …you have installed the [app]` Platform` binary distribution.
- …you have installed a &PLATFORM_NAME;server binary distribution *which bundles* JBoss AS.

You *MUST* set JBOSS_HOME if…

- …you are installing the [app]` Platform` or any of the &PLATFORM_NAME; servers *from source*.
- …you are installing the [app]` Platform` binary distribution, or one of the &PLATFORM_NAME; server binary distributions, which *do not* bundle JBoss AS.

Naturally, if you installed the [app]` Platform` or one of the &PLATFORM_NAME; server binary releases which *do not* bundle JBoss AS, yet requires it to run, then you should install JBoss AS before setting JBOSS_HOME or proceeding with anything else.

Setting the JBOSS_HOME Environment Variable on Unix

The JBOSS_HOME environment variable must point to the directory which contains all of the files for the [app]` Platform` or individual &PLATFORM_NAME; server that you installed. As another hint, this topmost directory contains a *bin* subdirectory.

Setting JBOSS_HOME in your personal *~/.bashrc* startup script carries the advantage of retaining effect over reboots. Each time you log in, the environment variable is sure to be set for you, as a user. On Unix, it is possible to set JBOSS_HOME as a system-wide environment variable, by defining it in */etc/bashrc*, but this method is neither recommended nor detailed in these instructions.

Procedure: To Set JBOSS_HOME on Unix…

1. Open the *~/.bashrc* startup script, which is a hidden file in your home directory, in a text editor, and insert the following line on its own line while substituting for the actual install location on your system:

```
export JBOSS_HOME="/home/<username>/<path>/<to>/<install_directory>"
```

2. Save and close the *.bashrc* startup script.
3. You should `source` the *.bashrc* script to force your change to take effect, so that `JBOSS_HOME` becomes set for the current session [2: Note that any other terminals which were opened prior to your having altered .bashrc will need to source ~/.bashrc as well should they require access to JBOSS_HOME.].

```
~]$ source ~/.bashrc
```

4. Finally, ensure that `JBOSS_HOME` is set in the current session, and actually points to the correct location:

The command line usage below is based upon a binary installation of the [app]` Platform`. In this sample output, `JBOSS_HOME` has been set correctly to the `topmost_directory` of the `installation. Note that if you are installing one of the standalone [app]` servers (with `JBoss AS` bundled!), then `JBOSS_HOME` would point to the `topmost_directory` of your server installation.

```
~]$ echo $JBOSS_HOME
/home/silas/
```

Setting the JBOSS_HOME Environment Variable on Windows

The `JBOSS_HOME` environment variable must point to the directory which contains all of the files for the &PLATFORM_NAME;Platform or individual &PLATFORM_NAME;server that you installed. As another hint, this topmost directory contains a *bin* subdirectory.

For information on how to set environment variables in recent versions of Windows, refer to http://support.microsoft.com/kb/931715.

Setting CATALINA_HOME on Linux and Windows

Procedure: Setting the `CATALINA_HOME` Environment Variable on Linux

1. The `CATALINA_HOME` environment variable must point to the location of your Tomcat installation. Any &PLATFORM_NAME; server which runs on top of the Tomcat servlet container has a topmost directory, i.e. the directory in which you unzipped the zip file to install the server, and underneath that directory, a *bin* directory. `CATALINA_HOME` must be set to the topmost directory of your &PLATFORM_NAME; server installation.

   Setting this variable in your personal *~/.bashrc* file has the advantage that it will always be set

(for you, as a user) each time you log in or reboot the system. To do so, open *~/.bashrc* in a text editor (or create the file if it doesn't already exist) and insert the following line anywhere in the file, taking care to substitute *<sip_server>* for the topmost directory of the &PLATFORM_NAME; server you installed:

```
export CATALINA_HOME="/home/<username>/<path>/<to>/<sip_server>"
```

Save and close *.bashrc*.

2. You can—and should--`source` your *.bashrc* file to make your change take effect (so that `CATALINA_HOME` is set) for the current session:

```
~]$ source ~/.bashrc
```

3. Finally, make sure that `CATALINA_HOME` has been set correctly (that it leads to the right directory), and has taken effect in the current session.

   The following command will show the path to the directory pointed to by `CATALINA_HOME`:

```
~]$ echo $CATALINA_HOME
```

   To be absolutely sure, change your directory to the one pointed to by `CATALINA_HOME`:

```
~]$ cd $CATALINA_HOME && pwd
```

Procedure: Setting the `CATALINA_HOME` Environment Variable on Windows

1. The `CATALINA_HOME` environment variable must point to the location of your Tomcat installation. Any &PLATFORM_NAME; server which runs on top of the Tomcat servlet container has a topmost directory, i.e. the directory in which you unzipped the zip file to install the server, and underneath that directory, a *bin* directory. `CATALINA_HOME` must be set to the topmost directory of your &PLATFORM_NAME; server installation.

   If you are planning on running the Tomcat container as the Administrator, then you should, of course, set the `CATALINA_HOME` environment variable *as the administrator*, and if you planning to run Tomcat as a normal user, then set `CATALINA_HOME` as a user environment variable.

   For information on how to set environment variables in Windows, refer to http://support.microsoft.com/kb/931715.