Installation Guide to Restcomm SMSC

Table of Contents

Preface

Document Conventions

This manual uses several conventions to highlight certain words and phrases and draw attention to specific pieces of information.

In PDF and paper editions, this manual uses typefaces drawn from the Liberation Fonts set. The Liberation Fonts set is also used in HTML editions if the set is installed on your system. If not, alternative but equivalent typefaces are displayed. Note: Red Hat Enterprise Linux 5 and later includes the Liberation Fonts set by default.

Typographic Conventions

Four typographic conventions are used to call attention to specific words and phrases. These conventions, and the circumstances they apply to, are as follows.

`Mono-spaced Bold`

Used to highlight system input, including shell commands, file names and paths. Also used to highlight key caps and key-combinations. For example:

> To see the contents of the file *my_next_bestselling_novel* in your current working directory, enter the `cat my_next_bestselling_novel` command at the shell prompt and press `Enter` to execute the command.

The above includes a file name, a shell command and a key cap, all presented in Mono-spaced Bold and all distinguishable thanks to context.

Key-combinations can be distinguished from key caps by the hyphen connecting each part of a key-combination. For example:

> Press `Enter` to execute the command.
>
> Press to switch to the first virtual terminal. Press to return to your X-Windows session.

The first sentence highlights the particular key cap to press. The second highlights two sets of three key caps, each set pressed simultaneously.

If source code is discussed, class names, methods, functions, variable names and returned values mentioned within a paragraph will be presented as above, in `Mono-spaced Bold`. For example:

> File-related classes include `filesystem` for file systems, `file` for files, and `dir` for directories. Each class has its own associated set of permissions.

`Proportional Bold`

This denotes words or phrases encountered on a system, including application names; dialogue box text; labelled buttons; check-box and radio button labels; menu titles and sub-menu titles. For example:

> Choose **System** > **Preferences** > **Mouse** from the main menu bar to launch `Mouse Preferences`. In the Buttons tab, click the Left-handed mouse check box and click **[Close]** to switch the primary mouse button from the left to the right (making the mouse suitable for use in the left hand).
>
> To insert a special character into a `gedit` file, choose **Applications** > **Accessories** > **Character Map** from the main menu bar. Next, choose **Search** > **Find** | **] from the `Character Map` menu bar** | **type the name of the character in the Search field and click [Next**. The character you sought will be highlighted in the Character Table. Double-click this highlighted character to place it in the Text to copy field and then click the **[Copy]** button. Now switch back to your document and choose **Edit** > **Paste** from the `gedit` menu bar.

The above text includes application names; system-wide menu names and items; application-specific menu names; and buttons and text found within a GUI interface, all presented in Proportional Bold and all distinguishable by context.

Note the menu:>[] shorthand used to indicate traversal through a menu and its sub-menus. This is to avoid the difficult-to-follow 'Select from the **Preferences** | **] sub-menu in the menu:System[** menu of the main menu bar' approach.

`Mono-spaced Bold Italic` or `Proportional Bold Italic`

Whether Mono-spaced Bold or Proportional Bold, the addition of Italics indicates replaceable or variable text. Italics denotes text you do not input literally or displayed text that changes depending on circumstance. For example:

> To connect to a remote machine using ssh, type `ssh username@domain.name` at a shell prompt. If the remote machine is *example.com* and your username on that machine is john, type `ssh john@example.com`.
>
> The `mount -o remount file-system` command remounts the named file system. For example, to remount the */home* file system, the command is `mount -o remount /home`.
>
> To see the version of a currently installed package, use the `rpm -q package` command. It will return a result as follows: `package-version-release`.

Note the words in bold italics above —username, domain.name, file-system, package,

version and release. Each word is a placeholder, either for text you enter when issuing a command or for text displayed by the system.

Aside from standard usage for presenting the title of a work, italics denotes the first use of a new and important term. For example:

> When the Apache HTTP Server accepts requests, it dispatches child processes or threads to handle them. This group of child processes or threads is known as a *server-pool*. Under Apache HTTP Server 2.0, the responsibility for creating and maintaining these server-pools has been abstracted to a group of modules called *Multi-Processing Modules* (*MPMs*). Unlike other modules, only one module from the MPM group can be loaded by the Apache HTTP Server.

Pull-quote Conventions

Two, commonly multi-line, data types are set off visually from the surrounding text.

Output sent to a terminal is set in `Mono-spaced Roman` and presented thus:

```
books        Desktop   documentation  drafts  mss    photos   stuff  svn
books_tests  Desktop1  downloads      images  notes  scripts  svgs
```

Source-code listings are also set in `Mono-spaced Roman` but are presented and highlighted as follows:

```
package org.jboss.book.jca.ex1;

import javax.naming.InitialContext;

public class ExClient
{
   public static void main(String args[])
       throws Exception
   {
      InitialContext iniCtx = new InitialContext();
      Object         ref    = iniCtx.lookup("EchoBean");
      EchoHome       home   = (EchoHome) ref;
      Echo           echo   = home.create();

      System.out.println("Created Echo");

      System.out.println("Echo.echo('Hello') = " + echo.echo("Hello"));
   }

}
```

Notes and Warnings

Finally, we use three visual styles to draw attention to information that might otherwise be overlooked.

Note

A note is a tip or shortcut or alternative approach to the task at hand. Ignoring a note should have no negative consequences, but you might miss out on a trick that makes your life easier.

Important

Important boxes detail things that are easily missed: configuration changes that only apply to the current session, or services that need restarting before an update will apply. Ignoring Important boxes won't cause data loss but may cause irritation and frustration.

Warning

A Warning should not be ignored. Ignoring warnings will most likely cause data loss.

Provide feedback to the authors!

If you find a typographical error in this manual, or if you have thought of a way to make this manual better, we would love to hear from you! Please submit a report in the the {this-issue.tracker.ur}, against the product Restcomm SMSC` `, or contact the authors.

When submitting a bug report, be sure to mention the manual's identifier: Restcomm SMSC

If you have a suggestion for improving the documentation, try to be as specific as possible when describing it. If you have found an error, please include the section number and some of the surrounding text so we can find it easily.

Chapter 1. Introduction

Restcomm SMSC is an Open Source Java based SMSC Gateway built on a modern extensible middleware platform. Restcomm SMSC enables operators to provide core SMSC features like mobile subscriber to subscriber SMS messaging, broadcasting campaigns and group messaging [1: Group Messaging can be achieved using RestComm. For more details please contact info@telestax.com.] between subscribers. In addition it also enables value added services through integration with OTT social networks and microblogs such as Facebook and Twitter.

Restcomm SMSC is an easy-to-install and easy-to-deploy platform that can be set up and configured very quickly. Since it is built on a modern and extensible JSLEE platform it automatically provides out-of-the-box integration with various communication protocols such as SIP, Diameter, HTTP, XCAP, XMPP, MGCP and others in addition to the base SMPP protocol for connectivity to ESMEs.

Restcomm SMSC stores messages using Cassandra database. The database is used for storing unsent messages, messages successfully sent and messages that failed to be sent.

Restcomm SMSC is cloud-ready. It can be deployed on dedicated hardware, private cloud infrastructure or public laaS such as AWS. Restcomm SMSC supports TDM hardware offered by major vendors in the market, namely Intel family boards (Dialogic) and Zaptel/Dahdi (Digium, Sangoma).

A single Restcomm SMSC node can process up to 1000 SMS/sec. Multiple SMSC nodes can be arranged in a cluster across one or more geographically distributed data centers to scale up throughput and provide various levels of redundancy, high availability and fault tolerance.

Restcomm SMSC is based on the robust and proven Restcomm JAIN SLEE 1.1 Server and Restcomm jSS7 Stack. Restcomm JAIN SLEE Server is a highly scalable event-driven application server with a robust component model and fault tolerant execution environment. It provides a set of connectors to a variety of networks elements: SS7 MAP, CAP, INAP, TCAP, ISUP, SMPP, XMPP, SIP, MGCP, HTTP, XDM, XCAP, Diameter and many others. It is fully compliant with JSR 240 (JSLEE 1.1). Restcomm jSS7 is a software based implementation of the SS7 protocol. It provides implementation for Level 2 and above in the SS7 protocol Stack. Restcomm jSS7 Stack User Guide is bundled within and you can refer to the guide for more details on the Stack.

This guide will assist you in installing Restcomm SMSC . For more details on configuring and using the platform, please refer to the Restcomm SMSC Admin Guide.

Chapter 2. Pre-Requisites

Restcomm SMSC 's core requirement is Java. The following table details the Hardware, Operating System and Software requirements for a clean installation of Restcomm SMSC .

Table 1. Installation Pre-Requisites

Component	Requirement	Notes
System Requirements	Intel Pentium 1 GHz or faster for simple applications. Hard disk space of at least 10GB. RAM of at least 1.5 GB	Higher the RAM and processing power, better is the throughput
TDM Hardware (Optional)	Restcomm SMSC Gateway supports legacy E1/T1 links via SS7 cards. You must have dahdi or dialogic cards installed along with their native libraries. Restcomm SMSC Gateway also supports SIGTRAN (M3UA).	In case if you are connecting to core network via SIGTRAN, TDM hardware is not needed.
Operating System	The platform can be installed on any OS that supports Java. But native libraries for SS7 cards are compiled only for Linux at the moment and therefore supported only on Linux Operating System.	The libraries for SS7 cards will be compiled for Windows in future releases. For SIGTRAN the OS should support SCTP protocol.
Java	You must have a working Java Runtime Environment (JRE) or Java Development Kit (JDK) installed on your system and it must be version 7 or higher. M3UA uses Java SCTP which is available only from JDK 7 onwards. Connectivity to MSC/HLR is via SIGTRAN links (M3UA).	

Component	Requirement	Notes
Database	You must have Cassandra database (version 1.2 or 2.0) installed (Version 2.1 or later is not supported now). We strongly recommend that you use the latest version of 2.0 release because the support for version 1.2 will not be soon available, the most probably in the next release. Restcomm SMSC Gateway uses Cassandra Native Driver and therefore you should enable support for native drivers by setting the value of start_native_transport to true in the file apache-cassandra-<version>/conf/cassandra.yaml. Failure to do this may result in the Gateway not starting correctly.	
Firewall Access	You must ensure that you have appropriate firewall permissions to allow access to IP:8080. This is required to access the management consoles.	
SCTP libraries	If you intend to use SIGTRAN (M3UA), you must have the lksctp library installed. The Linux Kernel Stream Control Transmission Protocol (lksctp) library provides SCTP implementation.	For more details on downloading and installing lksctp, please refer to http://lksctp.sourceforge.net/

You must ensure that the `JAVA_HOME` Environment variable is set properly for the user account(s) that will run the server. For more details on setting this variable, please refer to the appendix section.

Chapter 3. Hardware Setup

For legacy SS7 links, you must have relevant SS7 cards installed along with their native libraries.

Restcomm SMSC supports `dahdi` based SS7 cards like `diguim` and `sangoma` as well as `dialogic` cards. Since `dahdi` based SS7 crads do not have MTP2/MTP3 support on board they rely on external software to provide these services. But `dialogic` based SS7 cards have on board support for MTP2/MTP3.

This guide does not provide installation instructions for SS7 hardware. You must refer to respective vendor documentation for installing and configuring the hardware cards. The following external links point to information that will assist you in setting up the hardware.

- Sangoma: http://wiki.sangoma.com/
- Diguim: http://www.digium.com/en/products/digital/
- Dialogic: http://www.dialogic.com/

Chapter 4. Database Setup

Restcomm SMSC stores messages using Cassandra database. The database is used for storing unsent messages, messages successfully sent and messages that failed to be sent. You must have Cassandra database (version 1.2 or 2.0) installed. We stronly recommend that you use the version 2.0 (v 2.0.16 is the most recent version yet). The support for version 1.2 will not be soon available, the most probably already in the next release.

Restcomm SMSC uses Cassandra Native Driver and therefore you should enable support for native drivers by setting the value of `start_native_transport` to `true` in the file *apache-cassandra-1.2.8/conf/cassandra.yaml*. Failure to do this may result in the Gateway not starting correctly.

To install the Cassandra database, look for distributives at the offical Apache website here http://cassandra.apache.org/download/. We recommend that you download the latest binaries and install the database following the instructions specified in the Apache documentation in their website http://wiki.apache.org/cassandra/GettingStarted.

If you already use cassandra version 1.2, you must think to update it to version 2.0.16 when it is possible before next version has been released. If you do not have database routing rules and it is not important to you to lose of pending messages in SMSC GW, you can just remove RestCommSMSC Keyspace, uninstall 1.2 version, install a new one and create an empty RestCommSMSC Keyspace. Else you can update your database as DataStax recommends: http://docs.datastax.com/en/upgrade/doc/upgrade/cassandra/upgradeCassandraDetails.html.

Chapter 5. Downloading and Installing

Installing Restcomm SMSC is easy and quick with the binary download. You can either download the binary release or download the source code and set up from source.

5.1. Binary Download and Installation

The binary release is available for download at TeleStax Customer Support Portal &THIS.RELEASE_BINARY_URL;link:

Procedure: Binary Download and Installation

1. Download the zip file `<filename>` to any folder of your choice.
2. Extract the contents of the zip file.

```
Downloads]$ unzip <filename>
```

3. Verify the contents of the newly created directory.

5.2. Directory Structure

When you download the binary release, you will notice that the top level directory is named `mobicents-smscgateway-<version>` and immediately below this are five sub-directories as explained below:

- cassandra: Contains description of the Cassandra database tables' structure.
- docs: Contains all relevant documentation in respective subfolders.
- jboss-5.1.0.GA: The core server with two profiles "default" and "simulator". The "default" profile is a clean profile where you will have to start from scratch and configure the entire SS7 Stack and SMSC Gateway. The "simulator" profile is a pre-configured profile to work with jss7-simulator. Refer to the Admin Guide for instructions on how to start the server in either of the profiles.
- resources: Contains SLEE MAP RA jars.
- tools: Contains SLEE tools and SMSC test tools as explained below.
  - restcomm-hlr-simulator: Command line HLR simulator used for load testing of SMSC. restcomm-hlr-simulator is pre-configured to integrate with SMSC run in simulator profile.
  - restcomm-jss7-simulator: jSS7 Simulator that can be run in GUI or command line mode. This tool is useful to test `MoForwardSM` and other such functionality of SMSC like "when subscriber absent", etc. For more details on using this tool, please refer to Restcomm SS7Stack User Guide. restcomm-jss7-simulator is pre-configured to integrate with SMSC run in simulator profile.
  - restcomm-smpp-load: smpp-load tool is a Command line simulator to generate SMPP load. You must have `ant` installed to be able to run this tool. The smpp-load tool can be started to run as a SMPP Server accepting in-coming connection (BIND) from Restcomm SMSC or run

as a SMPP Client to send BIND to Restcomm SMSC . You can confiugre this tool by editing the *build.xml* to define how many SMPP connections should be initiated, what kind of load should be generated, etc.

- restcomm-smpp-simulator: smpp-simulator is a GUI tool to generate SMPP load. It can only initiate BIND and act as a SMPP Client. You can also use it to test other functionalities like `UCS2`, breaking `SUBMIT_SM` into multiple SMS etc.

```
|- restcomm-smsc-<-version->
        |- cassandra
        |- docs
                |+ container
                |+ jss7
                |+ management-hq
                |+ resources
                |+ slee
                |+ smsc
                |+ tools
        |- jboss-5.1.0.GA
                |+ bin    //contains start up and shutdown scripts for the Server.
                |+ client
                |+ common
                |+ docs
                |+ lib
                |- server
                    |+ default  //clean profile to set up from scratch
                    |+ simulator    //pre-configured profile to work with the jss7-
simulator
        |- resources
                |+ diameter-base
                |+ diameter-ro
                |+ map
                |+ sip11
        |- tools
                |+ eclipslee
                |+ jopr-plugin
                |+ remote-slee-connection
                |+ snmp
                |+ restcomm-hlr-simulator
                |+ restcomm-jss7-simulator
                |+ restcomm-smpp-load
                |+ restcomm-smpp-simulator
                |+ twiddle
```

5.3. Setup from Source

Restcomm SMSC is an open source project and you have the freedom to build from source. Building from source means you can stay on top with the latest features. Whilst aspects of Restcomm SMSC are quite complicated, you may find ways to become contributors.

Restcomm SMSC works with JDK1.7 or above. In addition you must have the following tools installed.

- `Git Client` : Instructions for using GIT, including install, can be found at http://git-scm.com/book
- `Maven 3.2.X` : Instructions for using Maven, including install, can be found at http://maven.apache.org/
- `Ant 1.9.X` : Instructions for using Ant, including install, can be found at http://ant.apache.org

5.3.1. Release Source Code Building

1. Downloading the source code

   Use GIT to checkout a specific release source, the base URL is https://github.com/Restcomm/smscgateway, then add the specific release version.

   ```
   [usr]$ git clone https://userid@bitbucket.org/telestax/{this-folder}-
   smscgateway.git
   [usr]$ cd {this-folder}-smscgateway
   [usr]$ git checkout <version>
   ```

2. Building the source code

   Now that we have the source the next step is to build and install the source. Restcomm SMSC uses Maven 2 to build the system. You must ensure that `JAVA_HOME` environment variable is set properly prior to building the source.

   ```
   [usr]$ mvn clean install
   ```

5.3.2. Development Trunk Source Building

Similar process as for Release Source Code Building, the only change is don't switch to specific tag.

Chapter 6. Post Installation Configuration

6.1. Memory Settings

You should fine tune the JVM memory settings based on your needs but we recommend you allocate a minimum of 3 GB for initial and maximum heap size.

-Xms3072m

Initial heap size, set in megabytes

-Xmx3072m

Maximum heap size, set in megabytes

6.2. Database Settings

You must ensure that you configure the database correctly before using the Restcomm SMSC in production. Cassandra has the concept of a keyspace, which is similar to a database in a RDBMS. You must create a keyspace by following the instructions in the procedure below:

Procedure: Create a Cassandra database (keyspace)

1. You must have the database running prior to creating the keyspace. For instructions on how to run Cassandra, please refer to the vendor documentation available in their website http://wiki.apache.org/cassandra/GettingStarted.

```
[vinu@vinu-neha]$ cd $CASSANDRA_HOME
[vinu@vinu-neha cassandra]$ ./bin/cassandra -f
```

2. When you have the database running successfully, you can proceed to creating the keyspace by making use of the cql command as below:

```
[vinu@vinu-neha]$ CREATE KEYSPACE "RestCommSMSC" WITH REPLICATION = {'class' :
'SimpleStrategy', 'replication_factor': 1};
```

   You should setup the keyspace strategy and the replication factor as required. If the above command is executed successfully, it must not return any error message.

3. It is not required for you to create any tables in the keyspace; SMSC will create the tables when started. If the keyspace contains old tables you should remove them prior to starting the SMSC.
4. A new database (keyspace) will be created locally (a single working Cassandra node). It is a Cassandra cluster which has only one node. By adding more nodes, you can make it a multi node cluster.

Procedure: Destroy a Cassandra database (keyspace)

1. To destroy a Cassandra database (keyspace) you can make use of the `cql` command:

```
DROP KEYSPACE "RestCommSMSC";
```

2. Executing the above `cql` command will result in the immediate, irreversible removal of the keyspace 'RestCommSMSC', including all column families and data contained in the keyspace.

6.3. Configuring the Gateway

Now that you have installed Restcomm SMSC to suit your needs, you can go ahead and configure the SS7 Stack and the SMSC Gateway to meet your requirements. The Restcomm jSS7 Stack Admin Guide in the *restcomm-smscgateway-<version>/docs/ss7/* folder will assist you in configuring and managing the SS7 Stack. Only when you have completely configured the SS7 Stack to meet your requirements, you must go ahead with configuring the SMSC Gateway.

The Restcomm SMSC Admin Guide in the *restcomm-smscgateway-<version>/docs/smsc/* folder will assist you in configuring and managing the SMSC Gateway. To configure and manage both the Stack and the Gateway you can use the Command Line Interface (CLI) tool or the GUI Management Console that comes with the platform.

Chapter 7. Uninstalling

If you wish to remove Restcomm SMSC you can do so by deleting the installation directory.

Appendix A: 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 [2: At this point in time, it is possible to run most servers, such as the JAIN SLEE, 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.].

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

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"
```

You Installed Using the Non-RPM Installer, but Want the 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.

Procedure: Installing the 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 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 `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 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.

Testing

Finally, to make sure that you are using the correct JDK or Java version (5 or higher), and that the java executable is in your `PATH`, run the `java -version ` command in the terminal from your home directory:

```
~]$ java -version
java version "1.5.0_16"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.5.0_16-b03)
Java HotSpot(TM) Client VM (build 1.5.0_16-b03, mixed mode, sharing)
```

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`.

Appendix B: Setting the JBOSS_HOME Environment Variable

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

This guide does not detail building the [app]` Platform` or any &THIS.PLATFORM; servers from source. It is nevertheless useful to understand the role played by JBoss AS and JBOSS_HOME in the &THIS.PLATFORM; 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 &THIS.PLATFORM;server binary distribution *which bundles [app]``*.

You *MUST* set JBOSS_HOME if...

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

Naturally, if you installed the [app]` Platform` or one of the &THIS.PLATFORM; server binary

releases which *do not* bundle `, yet requires it to run, then you should install before setting [var]`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 &THIS.PLATFORM; 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 [3: 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/<path>/<to>/<install_directory>
```

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 &THIS.PLATFORM;Platform or individual &THIS.PLATFORM;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.

Appendix C: Revision History