Installation Guide to Restcomm jSS7 7.0.0-SNAPSHOT

Table of Contents

Preface . . . 1
Document Conventions . . . 2
Typographic Conventions . . . 2
Pull-quote Conventions . . . 4
Notes and Warnings . . . 5
Provide feedback to the authors! . . . 6
1. Introduction . . . 7
2. Pre-Requisites . . . 8
3. Hardware Setup . . . 10
4. Downloading . . . 11
4.1. Binary Download . . . 11
4.2. Setup from Source . . . 12
4.2.1. Release Source Code Building . . . 13
4.2.2. Development Trunk Source Building . . . 14
5. Installing Restcomm jSS7 . . . 15
5.1. Installation Options . . . 15
5.1.1. Restcomm jSS7 as a JBoss AS Service . . . 15
5.1.2. Restcomm jSS7 as a Signaling Gateway . . . 16
5.1.3. Shell - Command Line Interface (CLI) . . . 16
5.1.4. Installing the Restcomm jSS7 Simulator . . . 17
5.1.5. Restcomm jSS7 as a Standalone Component . . . 18
5.2. Installing of hardware native libraries. . . . 18
5.2.1. Installing of hardware native libraries for dahdi cards. . . . 18
5.2.2. Installing of hardware native libraries for dialogic cards. . . . 18
5.3. Post Installation Configuration . . . 19
5.3.1. Memory Settings . . . 19
6. Uninstalling . . . 20
Appendix A: Java Development Kit (): Installing, Configuring and Running . . . 21
Appendix B: Setting the JBOSS_HOME Environment Variable . . . 24

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 jSS7` `, or contact the authors.

When submitting a bug report, be sure to mention the manual's identifier: Restcomm jSS7

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 jSS7 is the only Open Source Java based implementation of the SS7 protocol stack. It provides implementation for MTP2, MTP3, ISUP, SCCP, TCAP, CAMEL (Phase I, Phase II, Phase III and Phase IV) and MAP protocols for a dedicated equipment. It also has in-built support for SIGTRAN (M3UA) over IP and strictly adheres to the standards and specifications defined by the International Telecommunications Union (ITU) and American National Standards Institute (ANSI). The platform offers developers with a flexible API set that hides the lower layer details (legacy SS7 links or SIGTRAN) and therefore makes it simple and easy to develop SS7 applications as well as to migrate your applications from TDM equipments to M3UA. Restcomm jSS7 is based on an easily scalable and configurable load-balancing architecture.

The platform offers developers with a flexible API set that hides the lower layer details (legacy SS7 links or SIGTRAN) and therefore makes it simple and easy to develop SS7 applications as well as to migrate your applications from TDM equipments to M3UA. Restcomm jSS7 is based on an easily scalable and configurable load-balancing architecture.

Restcomm jSS7 supports TDM hardware offered by major vendors in the market, namely Intel family boards (Dialogic) and Zaptel/Dahdi (Digium, Sangoma). Though for production we recommend Dialogic boards with MTP2 and MTP3 on-board only.

If you intend to use only M3UA you can install the jSS7 on any Operating System that supports Java and SCTP protocol. However if you wish to use SS7 cards, the native libaries for these are only compiled for Linux at the moment.

Restcomm jSS7 comes with JSLEE TCAP, MAP, CAP and ISUP Resource Adaptors (RA) that enable developers to build SS7 applications with ease. Developers only require an understanding of Resource Adaptors and can focus on building applications quickly and efficiently rather than worry about the SS7 stack. If you wish to use JSLEE Resource Adapters, the Command Line Interface (CLI - Shell Management tool) or the GUI for run-time configuration, then you must have JBoss Application Server installed and running. However if you do not wish to use the Resource Adaptors or CLI then Restcomm jSS7 can work as a standalone library.

The Open Source Software gives you the flexibility to understand the readily available source code and customise the product for your Enterprise needs.

This guide will assist you in installing Restcomm jSS7 . For more details on configuring and using the platform or for information regarding the supported protocols and compliant standards, please refer to the Restcomm jSS7 User Guide.

Chapter 2. Pre-Requisites

Restcomm jSS7 's core requirement is Java. The following table details the Hardware, Operating System and Software requirements for a clean installation of Restcomm jSS7 .

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 20MB. RAM of at least 1.5 GB	Higher the RAM and processing power, better is the throughput
TDM Hardware	For legacy SS7 links, you must have dahdi or dialogic cards installed along with their native libraries.	You do not require any specific hardware for SIGTRAN (M3UA)
Operating System	The platform can be installed on any OS that supports Java and SCTP. But native libraries for SS7 cards are compiled only for Linux at the moment and therefore supported only on Linux Operating System.	The libraries will be compiled for Windows in future releases.
Java	You must have a working Java Runtime Environment (JRE) or Java Development Kit (JDK) installed on your system and it must be version 5 or higher. If you intend to use SIGTRAN (M3UA) then you must have JDK 7 installed. M3UA uses Java SCTP which is available only from JDK 7 onwards. JAVA 8 is not supported now.	If you are using M3UA, then you must use JDK 7 to run the stack as well as to compile the source code. For TDM cards JDK 7 is also recommended.
JBoss Application Server	Restcomm jSS7 Stack can work as a standalone library if you do not require JSLEE Resource Adaptors and the Shell Management Tool (Command Line Interface) for run-time configuration of the platform. But if you wish to avail of these, then you must have JBoss Application Server (version 5.1.0.GA) installed on your machine.	Refer to the appendix section for more details.

Component	Requirement	Notes
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` and `JBOSS_HOME` Environment variables are set properly for the user account(s) that will run the server. For more details on setting these variables, please refer to the sections Java Development Kit (): Installing, Configuring and Running and Setting the JBOSS_HOME Environment Variable.

Chapter 3. Hardware Setup

You do not require any specific hardware if you intend to use only SIGTRAN (M3UA). But if you wish to use legacy SS7 links, then you must have relevant SS7 cards installed along with their native libraries.

Restcomm jSS7 supports `dahdi` based SS7 cards like `diguim` and `sangoma` as well as `dialogic` cards. Since `dahdi` based SS7 cards 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 and recommended.

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/

Before JSS7 stack can interoperate with an installed card you need to provide extra native libs for both Dialogic and dahdi cards. The way where you can obtain native libs is explained in Installing of hardware native libraries..

Chapter 4. Downloading

Installing Restcomm jSS7 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.

4.1. Binary Download

The binary release is available for download at TeleStax Customer Support Portal https://github.com/RestComm/jss7/releases/latest

Procedure: Binary Download

1. Download the zip file -*.zip* to any folder of your choice.
2. Extract the contents of the zip file.

```
Downloads]$ unzip -.zip
```

3. Verify the contents of the newly created directory.

When you download the binary release, you will notice that the top level directory is named - and immediately below this are five sub-directories named *asn*, *_docs*, *oam*, *sctp* and *ss7*, encompassing the major services and libraries that make up Restcomm jSS7 . For details on what is included in the sub-directories, please refer to the Restcomm jSS7 User Guide.

The major functional modules of the jSS7 are:

1. SS7 Service *restcomm-ss7-service*]
2. Signaling Gateway *restcomm-ss7-sgw*]
3. Shell *shell*]
4. SS7 Simulator *restcomm-ss7-simulator*]

```
|- restcomm-jss7-<version>
        |- asn

        |- _docs

        |- oam

        |- sctp

        |- ss7
                |+ protocols
                |+ shell
                |+ restcomm-ss7-service
                |+ restcomm-ss7-sgw
                |+ restcomm-ss7-simulator
                |+ restcomm-ss7-trace-parser
                |+ template
```

4.2. Setup from Source

Restcomm jSS7 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 jSS7 are quite complicated, you may find ways to become contributors.

Restcomm jSS7 works with JDK1.7 or above. In addition you must have the following tools installed.

Pre-Requisites for Building from Source

- `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
- `jmxtools:jar` : This library is required to build the Simulator source code. The library com.sun.jdmk:jmxtools:jar:1.2.1 must be downloaded manually and placed in your maven repository. Instructions are provided below.

Instructions to Download `jmxtools:jar` for building Simulator source code are as follows:

- The file can be downloaded from Oracle's website. The link is: http://www.oracle.com/technetwork/java/javasebusiness/downloads/java-archive-downloads-java-plat-419418.html#7657-jmx-1.2.1-oth-JPR&usg=AFQjCNGOqUaCB1ULVG7cMULio9u70MKocA.
- The webpage will display a list of Java Platform Technology Downloads. From the list, select "Java Management Extension (JMX) 1.2.1"
- Read and understand the License Agreement and then press "Accept License Agreement".
- Download the *jmx-1_2_1-ri.zip* file.

- Extract *lib/jmxtools:jar*
- Once extracted, place this file in your maven repository as *.m2/repository/com/sun/jdmk/jmxtools/1.2.1/jmxtools-1.2.1.jar* file if *.m2/repository* is a root of this repository. Alternatively you can register it.

4.2.1. Release Source Code Building

1. Downloading the source code

   Checkout a specific release source using GIT. The base URL to clone from is https://github.com/Restcomm/jss7. Then add the specific release version, in the below example we are downloading the version "3.0.0".

   ```
   [usr]$ git clone
   [usr]$ cd jss7
   [usr]$ git checkout 3.0.0
   ```

2. Building the source code

   Now that we have the source code, the next step is to build and install the source. Restcomm jSS7 uses Maven 3 to build the system. There are two build profiles namely default and dahdilinux. The Default profile builds only the java source code. The profile "dahdilinux" compile relevant native modules in addition to building the java source.

   At the moment, native modules are supported only for Linux OS.

   - Building using "default" Build profile

     To build Restcomm jSS7 without building any native libraries use the default profile as shown below.

     ```
     [usr]$ cd jss7
     [usr]$ mvn install
     ```

     If you are using Restcomm jSS7 without any native dependencies, Restcomm jSS7 can run on any OS.

   - Building using "dahdilinux" profile

     Use this profile only if the linux server on which this code is built has `dahdi` module installed. Make sure you pass the "include.zap" system property with a value pointing to the directory where `dahdi` is installed

     ```
     [usr]$ cd jss7
     [usr]$ mvn install -Pdahdilinux -Dinclude.zap=/usr/include/dahdi
     ```

3. Use Ant to build the binary.

```
[usr]$ cd jss7/release
[usr]$ ant
```

4.2.2. Development Trunk Source Building

To build from development trunk source, follow the same procedure as above but at the time of checkout do not switch to the specific release tag.

```
[usr]$ git clone
[usr]$ cd jss7
[usr]$ git checkout
```

The rest of the steps are as outlined in the above section Release Source Code Building

Chapter 5. Installing Restcomm jSS7

5.1. Installation Options

The Restcomm jSS7 is logically divided into two sections - lower and upper. The lower section provides implementation for SS7 Level 2 and Level 3 and is therefore influenced by the type of SS7 hardware (Level 1) used. The upper section provides implementation for SS7 Level 4 and above. Restcomm jSS7 gives you the flexibility to install and use the Levels 2,3 and 4 in the same JVM and machine where SS7 Hardware (Level 1) is installed. Alternately, you can also install Level 1,2 and 3 in one machine and install Level 4 on a separate machine. In the second scenario, `M3UA` over `SCTP` is used for communication between Level 3 and Level 4 (on different machines) and this is referred to as Restcomm Signaling Gateway.

Restcomm jSS7 can be installed to function as a standalone component if you do not wish to use JBoss Application Server. However if you intend to use JSLEE Resource Adaptors or Shell (CLI), then you must deploy it as a JBoss AS Service. The sections below provide instructions for installing the Stack for use with JBoss AS or as a standalone component.

5.1.1. Restcomm jSS7 as a JBoss AS Service

Restcomm SS7 Service can be deployed in any container that supports `JMX` and exposes `JNDI`. By using the Restcomm SS7 Service you will be able to configure the SS7 stack using CLI (Command Line Interface) commands. The SS7 Service wraps SS7 Level 4 (i.e., MAP, CAP and ISUP) and the lower layers and exposes via JNDI, such that the layer above can perform the look-up and use it in any application.

Procedure: Installing Restcomm SS7 Service

1. Pre-Requisites:
    - The Restcomm SS7 Service binary requires that you have JBoss Application Server installed and `JBOSS_HOME` Environment variable set properly. The latest binaries of JBoss Application Server can by obtained from https://github.com/RestComm/jain-slee/releases/latest. For more details on setting the `JBOSS_HOME` variable, please refer to the section Setting the JBOSS_HOME Environment Variable.
    - Ant 1.6 (or higher) must be used to install the binary. Instructions for using Ant, including install, can be found at http://ant.apache.org/.
    - If you are using the SS7 board on server, you must ensure that the `java.library.path` variable is set to point to the directory containing the native component. Alternatively you can copy it to the JBoss native library path manually.
2. Deploying the SS7 Service:
    - You can now deploy the service using the `ant deploy` command as shown below:

```
[usr1]$ cd -/ss7
[usr1]$ ant deploy
```

3. Result:
    - If the service has been deployed successfully, you will find the below message appear on screen:

```
Buildfile: /home/vinu/Downloads/restcomm-jss7-3.0.0/ss7/build.xml

deploy:
     [copy] Copying 38 files to /home/vinu/Downloads/restcomm-jss7-
3.0.0/ss7/${system.JBOSS_HOME}/server/default/deploy/restcomm-ss7-service
     [copy] Copying 2 files to /home/vinu/Downloads/restcomm-jss7-
3.0.0/ss7/${system.JBOSS_HOME}/bin
     [copy] Copying 9 files to /home/vinu/Downloads/restcomm-jss7-
3.0.0/ss7/${system.JBOSS_HOME}/lib

BUILD SUCCESSFUL
```

    - You have now deployed Restcomm SS7 Service successfully. Note that this procedure will also install the Shell Components (shell scripts and libraries) on this machine.

5.1.2. Restcomm jSS7 as a Signaling Gateway

Restcomm jSS7 can function as a standalone Signaling Gateway installed on any machine. It acts as a signaling agent that receives/sends Switched Circuit Network (SCN) native signaling at the edge of an IP network. Installing the Signaling Gateway is straightforward.

Procedure: Installing Signaling Gateway

1. Pre-Requisites:
    - You must have JDK 1.7 and lksctp library installed on your machine.
2. Copy *Restcomm-jSS7-7.0.0-SNAPSHOT/ss7/restcomm-ss7-sgw* to any folder of your choice in any machine that supports the requirements specified. Your Signaling Gateway is now ready to be used.

5.1.3. Shell - Command Line Interface (CLI)

Once you have installed Restcomm SS7 Service and the Signaling Gateway, you can configure and manage them both using Shell commands. Restcomm jSS7 comes with a Shell Management Interface that enables easy run-time configuration. You can install the Shell (CLI) Component on any machine (usually remote) and easily connect to and manage the Stack on a remote machine with simple commands. For more details on using the Shell and the commands available, please refer to the Restcomm jSS7 User Guide.

Procedure: Installing CLI

1. Pre-Requisites
    - You must have JBoss Application Server installed and `JBOSS_HOME` Environment variable set properly. For more details on setting the `JBOSS_HOME` variable, please refer to the section

Setting the JBOSS_HOME Environment Variable.

- Ant 1.6 (or higher) must be used to install the binary. Instructions for using Ant, including install, can be found at http://ant.apache.org/.

2. Copy *Restcomm-jSS7-7.0.0-SNAPSHOT/ss7* to any folder of your choice in any machine that supports the requirements specified.
3. You can now deploy using the `ant deploy` command as shown below:

```
[usr1]$ cd -/ss7
[usr1]$ ant deploy
```

4. Result:
   - If the Shell has been deployed successfully, you will find the below message appear on screen:

```
Buildfile: /home/vinu/Downloads/restcomm-jss7-3.0.0/ss7/build.xml

deploy:
     [copy] Copying 38 files to /home/vinu/Downloads/restcomm-jss7-
3.0.0/ss7/${system.JBOSS_HOME}/server/default/deploy/restcomm-ss7-service
     [copy] Copying 2 files to /home/vinu/Downloads/restcomm-jss7-
3.0.0/ss7/${system.JBOSS_HOME}/bin
     [copy] Copying 9 files to /home/vinu/Downloads/restcomm-jss7-
3.0.0/ss7/${system.JBOSS_HOME}/lib

BUILD SUCCESSFUL
```

   - You have now deployed the Shell Components (shell scripts and libraries) successfully on this machine. You can now Start the Shell and connect it to any SS7 service instance and manage the Stack from the Command Line.

5.1.4. Installing the Restcomm jSS7 Simulator

Restcomm jSS7 Simulator can function as a standalone component installed on any machine. The Simulator module will enable you to test and understand the functionality of the Stack.

Procedure: Installing Simulator

1. Pre-Requisites:
   - You must have JDK 1.7 installed on your machine.
2. Copy *Restcomm-jSS7-7.0.0-SNAPSHOT/ss7/restcomm-ss7-simulator* to any folder of your choice in any machine that supports the requirements specified. Your Simulator is now ready to be used.

5.1.5. Restcomm jSS7 as a Standalone Component

Restcomm jSS7 can be installed as a standalone Java library without any dependency on JBoss, Restcomm JSLEE or any other container. The Restcomm jSS7 User Guide will assist you in implemeting this and also give some details of how jSS7 layers can be configured. If you do not intend to use it with JBoss AS, then you must follow the regular way of initializing jSS7 Stack, which is to build each of the protocols, configure individually and bind them together.

5.2. Installing of hardware native libraries.

If you use hardware SS7 cards you need to make extra actions for installing of hardware native libraries. A hardware SS7 card must be already installed (refer to the section Hardware Setup).

5.2.1. Installing of hardware native libraries for dahdi cards.

Follow steps for installing of native libraries dahdi cards:

- We have the driver only for 32-bit JAVA for linux. Other OS and 64-bit is not supported.
- We need to have `JAVA 7` and `GCC` compiler installed
- Download latest dahdi complete from their site http://downloads.asterisk.org/pub/telephony/dahdi-linux-complete/ and unzip it
- Copy "dahdi_config.h" and "user.h" from the unzipped folder like: `dahdi-linux-complete-<version>/linux/include/dahdi/` to the linux "include" folder: `/usr/include/dahdi` (in case when linux "include" files are located in `/usr/include` folder)
- Compile dahdi native lib - run from the folder of downloaded jss7 sources (how to download source code - check Setup from Source) `jss7/hardware/dahdi` by the command: `mvn install -Pdahdilinux -Dinclude.zap=/usr/include/dahdi`
- Copy the compiled native lib "libmobicents-dialogic-linux.so" from `jss7/hardware/dahdi/native/linux/target` folder to
  - `jboss-5.1.0.GA/bin/META-INF/lib/linux2/x86` folder if you run JSS7 under jboss
  - other folder where JSS7 will search a native lib in case of a stand-alone usage
- in JBOSS case by default "jboss-beans.xml" config file in the jboss deploy folder (the folder like `jboss-5.1.0.GA/server/default/deploy/Restcomm-ss7-service/META-INF`) is configured to use m3ua stack by default. If you want to use SS7 hardware cards you need to update it - update the definition of bean "Mtp3UserPart" to the corresponded hardware depended bean.

5.2.2. Installing of hardware native libraries for dialogic cards.

Follow steps for installing of native libraries dialogic cards:

- Download latest DSI Software from https://www.dialogic.com/en/products/signaling-and-ss7-components/download/dsi-interface-protocol-stacks.aspx and unzip it (it may be already done during the hardware installation procedure)
- copy the native library "libgctjni.so"

    - for 32-bit JAVA: from subfolder `32` to jboss folder `jboss-5.1.0.GA/bin/META-INF/lib/linux2/x86` or other folder where JSS7 will search a native lib in case of a stand-alone usage
    - for 64-bit JAVA: from subfolder `64` to jboss folder `jboss-5.1.0.GA/bin/META-INF/lib/linux2/x64` or other folder where JSS7 will search a native lib in case of a stand-alone usage

- copy the gctApi library "gctApi.jar" from subfolder `JAVA` to the folder where JSS7 stack jar files are deployed.
    - In JBOSS case the target folder may be folder `jboss-5.1.0.GA/server/default/deploy/Restcomm-ss7-service/lib` for the default profile and `jboss-5.1.0.GA/server/simulator/deploy/Restcomm-ss7-service/lib` for the simulator profile.
    - For products like SMSC GW, USSD GW, CAMEL GW, GMLC the target folder will be the folders like `jboss-5.1.0.GA/server/default/deploy/Restcomm-smsc-server/lib` for the default profile and `jboss-5.1.0.GA/server/simulator/deploy/Restcomm-smsc-server/lib` for the simulator profile.
    - For stand-alone installation the target folder is where SS7 stack jars are deployed

- in JBOSS case by default "jboss-beans.xml" config file in the jboss deploy folder (the folder like `jboss-5.1.0.GA/server/default/deploy/Restcomm-ss7-service/META-INF`) is configured to use m3ua stack by default. If you want to use SS7 hardware cards you need to update it - update the definition of bean "Mtp3UserPart" to the corresponded hardware depended bean. Examples of jboss-beans.xml you can find in the following folders (you can just these files to `META-INF` folder):
    - for usage only a Dialogic board without m3ua (sigtran) - `ss7/template/META-INF-dialogic`
    - for usage a Dialogic board together with m3ua (sigtran) - `ss7/template/META-INF-m3ua-dialogic`

5.3. Post Installation Configuration

Now that you have installed Restcomm jSS7 to suit your needs, you can go ahead and configure the Stack to meet your requirements. The User Guide (available along with this Installation Guide) in the *Restcomm-jSS7-7.0.0-SNAPSHOT/docs* folder will assist you in configuring and managing the Stack. The Shell Management module will enable you to easily configure the Stack using the Command Line Interface (CLI) tool.

5.3.1. 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

Chapter 6. Uninstalling

If you wish to remove Restcomm jSS7 you can do so by deleting the installation directory. If you installed it as a JBoss Service then you must remember to clean up the SS7 Service files within the JBoss directory by undeploying the service as shown below. The procedure below can be ignored if you installed the Stack as a standalone component.

Procedure: Uninstalling Restcomm SS7 Service

Undeploy:

```
[usr1]$ cd -/ss7
[usr1]$ ant undeploy
```

Result:

```
Buildfile: /home/vinu/Downloads/restcomm-jss7-6.1.3.GA/ss7/build.xml

undeploy:
   [delete] Deleting directory /home/vinu/Downloads/restcomm-jss7-
6.1.3.GA/ss7/${system.JBOSS_HOME}/server/default/deploy/mobicents-ss7-service

BUILD SUCCESSFUL
Total time: 0 seconds
```

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 (version 7 is recommended, version 8 is not supported now) [1: 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 Oracle JDK 7 (Java Development Kit) from Oracle's website: http://www.oracle.com/technetwork/java/javase/downloads/java-archive-downloads-javase7-521261.html. Click on the Download link next to "Java SE Development Kit `<version>`" (where `[replaceable]`<version>` is the latest 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 `Oracle Binary Code License Agreement for Java` , and proceed to the download page.

The Oracle website will present two download alternatives to you: one is an RPM inside a self-extracting file (for example, *jdk-<version>-linux-i586.rpm*), and the other is merely a self-extracting

file (e.g. *jdk-<version>-linux-i586.tar.gz*). 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 jSS7 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-<version>-linux-<architecture>-rpm.bin"
~]$ ./"jdk-<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_1.7.0` 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_1.7.0` 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 "1.7.0_16"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.7.0_16-b03)
Java HotSpot(TM) Client VM (build 1.7.0_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 Restcomm Platform (Restcomm) is built on top of the JBoss Application Server. You do not need to set the JBOSS_HOME environment variable to run any of the Restcomm 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 Restcomm Platform or most of Restcomm servers are built on top of the JBoss Application Server (JBoss Application Server). When the Restcomm Platform or Restcomm servers are built on top of the JBoss Application Server (JBoss Application Server), then JBOSS_HOME *must* be set, because the Restcomm files are installed into (or "over top of" if you prefer) a clean JBoss Application Server 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 Application Server installation into which you want it to install the Restcomm files.

This guide does not detail building the Restcomm or any Restcomm servers from source. It is nevertheless useful to understand the role played by JBoss Application Server and JBOSS_HOME in the jSS7 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 Restcomm binary distribution.
- ...you have installed a Restcomm server binary distribution *which bundles* JBoss Application Server.

You *MUST* set JBOSS_HOME if...

- ...you are installing the Restcomm or any of the Restcomm servers *from source*.
- ...you are installing the Restcomm binary distribution, or one of the Restcomm server binary

distributions, which *do not* bundle JBoss Application Server.

Naturally, if you installed the Restcomm or one of the Restcomm server binary releases which *do not* bundle JBoss Application Server, yet requires it to run, then you should install 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 Restcomm or individual Restcomm; 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 Restcomm. In this sample output, JBOSS_HOME has been set correctly to the topmost_directory of the Restcomm installation. Note that if you are installing one of the standalone Restcomm servers (with JBoss Application Server 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 jSS7 Platform or individual jSS7 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.

Unresolved directive in SS7_Stack_Installation_Guide.adoc - include::Revision_History.adoc[]