SIP Servlets Server User Guide

The Guide to the SIP Servlets v1.1-Certified Server

by Douglas Silas, Jean Deruelle, Vladimir Ralev, Ivelin Ivanov, and Jared Morgan

1. 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* [https://fedorahosted.org/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.

1.1. 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 **Ctrl**+**Alt**+**F1** to switch to the first virtual terminal. Press **Ctrl**+**Alt**+**F7** 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 **>** shorthand used to indicate traversal through a menu and its sub-menus. This is to avoid the difficult-to-follow 'Select **Mouse** from the **Preferences** sub-menu in the **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.

1.2. 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"));
   }

}
```

1.3. 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.

Introduction to the SIP Servlets Server

JBoss Communications SIP (Session Initiation Protocol) Servlets deliver a consistent, open platform on which to develop and deploy portable and distributed SIP and Java Enterprise Edition services. The JBoss Communications SIP Servlets Server is a *certified* implementation of the SIP Servlet v1.1 (JSR 289) specification that can run on top of either the JBoss Application Server or the Tomcat Servlet Container.

JBoss Communications SIP Servlets for JBoss (JBCP SIP Servlets for JBoss) strives to develop interoperability standards between SIP Servlets and the Java Service Logic Execution Environment (JSLEE) so that applications can exploit the strengths of both. The JAIN-SIP Reference Implementation is leveraged as the SIP stack, and the JBoss Communications JAIN SLEE Server is used as the SLEE implementation.

Features of the JBoss Communications SIP Servlets Server

- The first *certified* SIP Servlet v1.1 (JSR 289) implementation.
- Carrier grade performance.
- Load balancing, cluster and failover support.
- Converged SIP and HTTP session management.
- Media and Diameter support.
- A browser-based Management Console.
- A bundled JSLEE/SIP interoperability demonstration application for JBCP SIP Servlets for JBoss.
- JBoss Communications Media Server.
- Extensions, including SUBSCRIBE/NOTIFY

1.1. High-Availability: SIP Servlets Server Load Balancing, Clustering and Failover

Telecommunications applications demand High-Availability (HA), fault tolerance, scalability and performance. Providing highly-available end-user applications that are tolerant of faults is commonly achieved through the use of clustering technologies.

Clustering is a complex subject that is often used to collectively address a variety of techniques aimed at improving the high-availability and scalability of services and applications. Such

techniques include distributed state replication, load balancing, and failover capabilities. The usage of any one of these techniques improves either reliability or performance, but at the expense of the other. It requires careful analysis of real-world scenarios to arrive at an architecture that represents the optimal balance of performance and reliability.

Based on experience with production deployments and extensive feedback from the Open Source community, JBoss Communications HA has undergone several iterations of refinement. In its current incarnation, the architecture can be described as a "star topology" with symmetric application servers and a smart, lightweight load-balancing element with built-in failover logic. The amount of state replication is kept to a minimum for maximum scalability with sufficiently-high reliability.

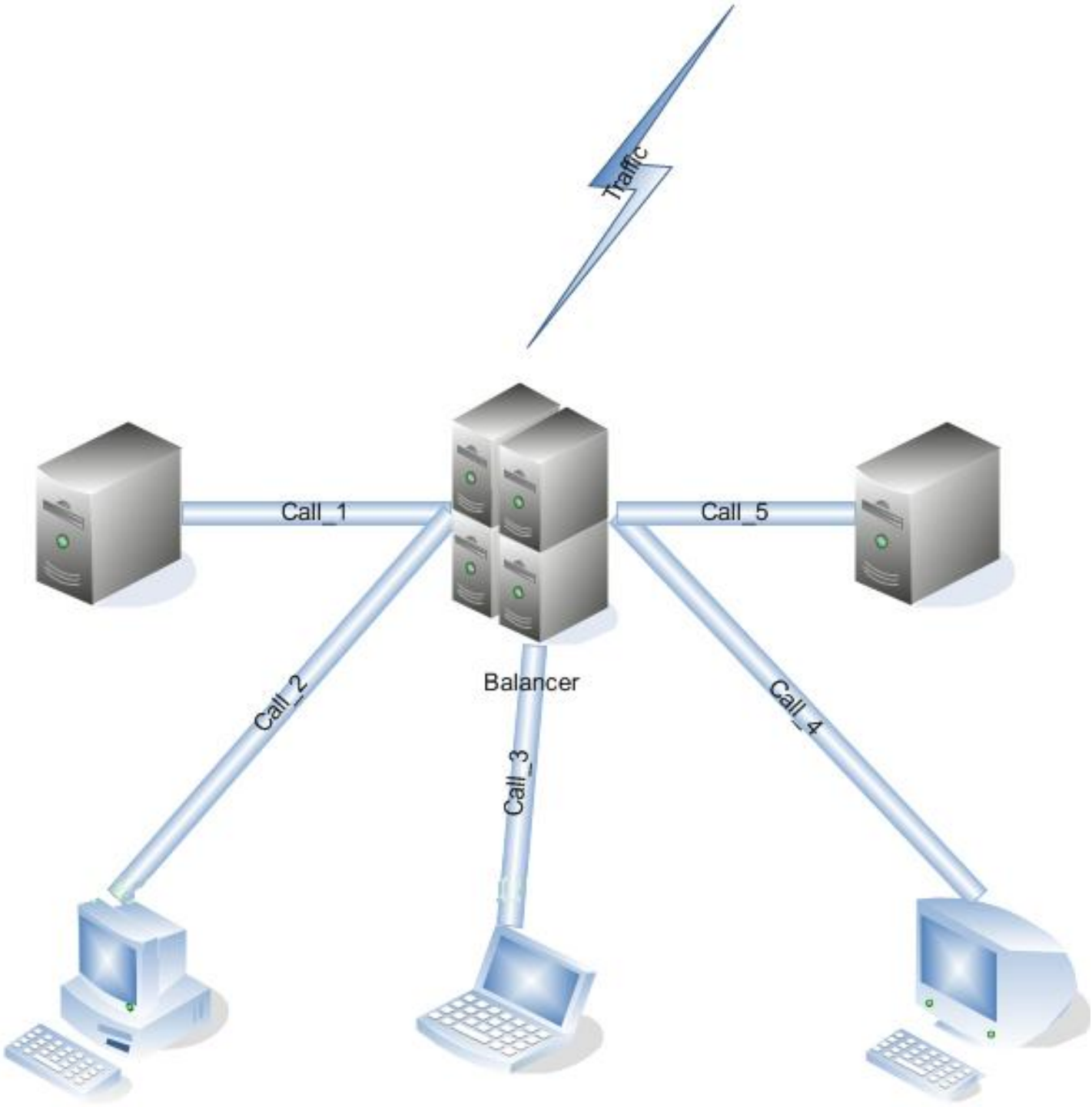


Figure 1.1. A cluster of JBoss Communications SIP Servlets Servers, showing the star network topology.

Clustering Terms and Definitions for JBoss Communications SIP Servlets. For purposes of clarity, the SIP Servlets High-Availability sections use terms like *cluster* with meanings specific to the context of JBoss Communications SIP Servlets. Therefore, the following definitions are provided to clarify more precisely what is meant by the terms *cluster*, *node*, *SIP Servlets Server*

and so on, in the subsequent sections, and in the context of JBoss Communications High-Availability.

Distinguishing Between a Cluster and Clustering Capabilities

The crux of possible confusion is this: any heterogeneous group of SIP Servlets Servers behind a SIP Load Balancer is, by definition, a *cluster*. Those SIP Servlets Servers can be either JBCP SIP Servlets for JBoss servers or JBCP SIP Servlets for Tomcat servers. However, a homogeneous group of JBCP SIP Servlets for JBoss servers served by a SIP Load Balancer, in addition to being a cluster, also possesses JBoss-specific *clustering capabilities*. Those clustering capabilities include, principally, state replication and the ability to fail over. Therefore, when specific *clustering capabilities* are spoken of, they are always referring to the context of a homogeneous cluster of JBCP SIP Servlets for JBoss server nodes served by a load balancer.

Glossary of Cluster-Related Terms

SIP Servlets Server
: A JBoss Communications *SIP Servlets Server* refers to either a SIP Servlets-enabled JBoss Application Server (JBCP SIP Servlets for JBoss) or a SIP Servlets-enabled Tomcat Servlet Container (JBCP SIP Servlets for Tomcat). Anywhere the term SIP Servlets Server is used, you are free to substitute the JBoss or the Tomcat variety depending on the one you are interested in.

node
: A *node* is simply a SIP Servlets Server in a *cluster*. In this document, a node can be either an JBCP SIP Servlets for JBoss server or an JBCP SIP Servlets for Tomcat server.

cluster
: A *cluster*, as used in this document, refers simply to a group of one or more *nodes*, i.e. *SIP Servlets Servers*, behind a SIP Load Balancer. The minimum number of nodes in a cluster is one. The case of a *cluster* with one node almost always occurs in a *degraded cluster*: one in which other nodes, for some reason, have become unavailable.

SIP Load Balancer
: The **JBoss Communications SIP Load Balancer** is not a full-fledged SIP Servlets Server itself. Rather, it is a simple *proxy server* whose primary purpose is to intelligently route SIP requests and replies between healthy and available SIP Servlets Servers residing in a *cluster* on a Local Area Network (LAN), and User Agents (UAs) accessing a SIP service or application from a Wide Area Network (WAN). The SIP Load Balancer therefore acts as a kind of gateway between a Wide Area Network with User Agents, and a Local Area Network wherein the SIP Servlets Server *cluster nodes* reside.

1.2. Working with the SIP Servlets Management Console

Once installed, the JBCP SIP Servlets for JBoss or JBCP SIP Servlets for Tomcat instance can be accessed and configured using the SIP Servlets Management Console. The management console is available at *http://localhost:8080/sip-servlets-management/*.

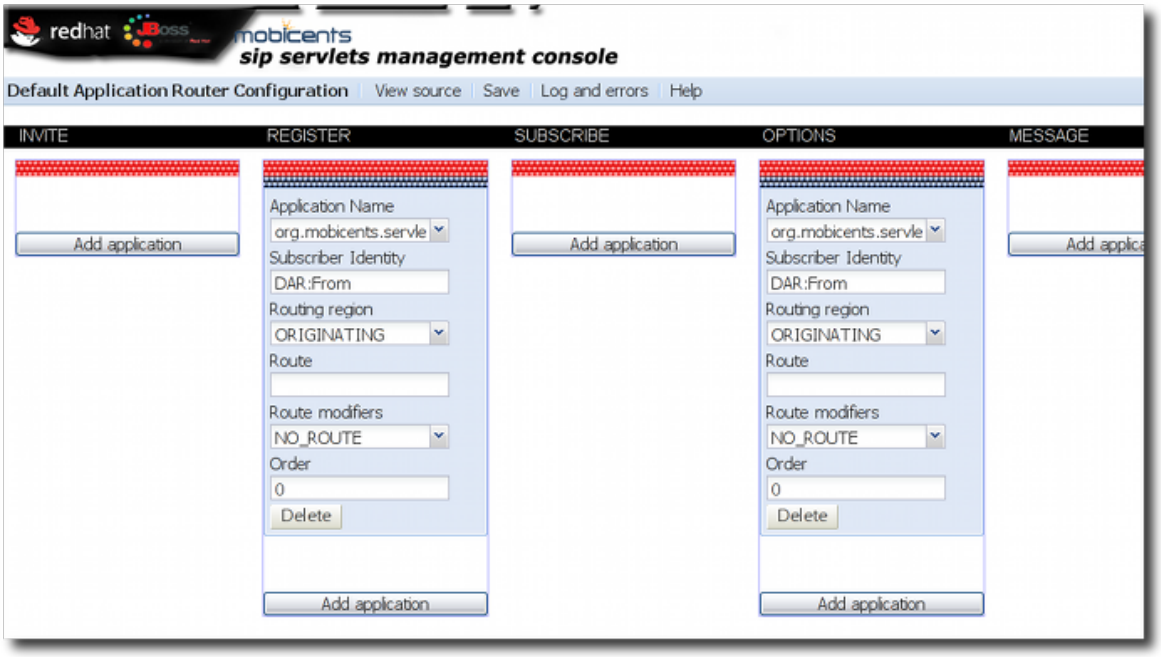


Figure 1.2. The SIP Servlets Management Console

Information on how to use the SIP Servlets Management Console is available from the **Help** link on the top main menu bar. Clicking **Help** displays a **Default Application Router Help** pop-up that can be repositioned and resized by dragging.

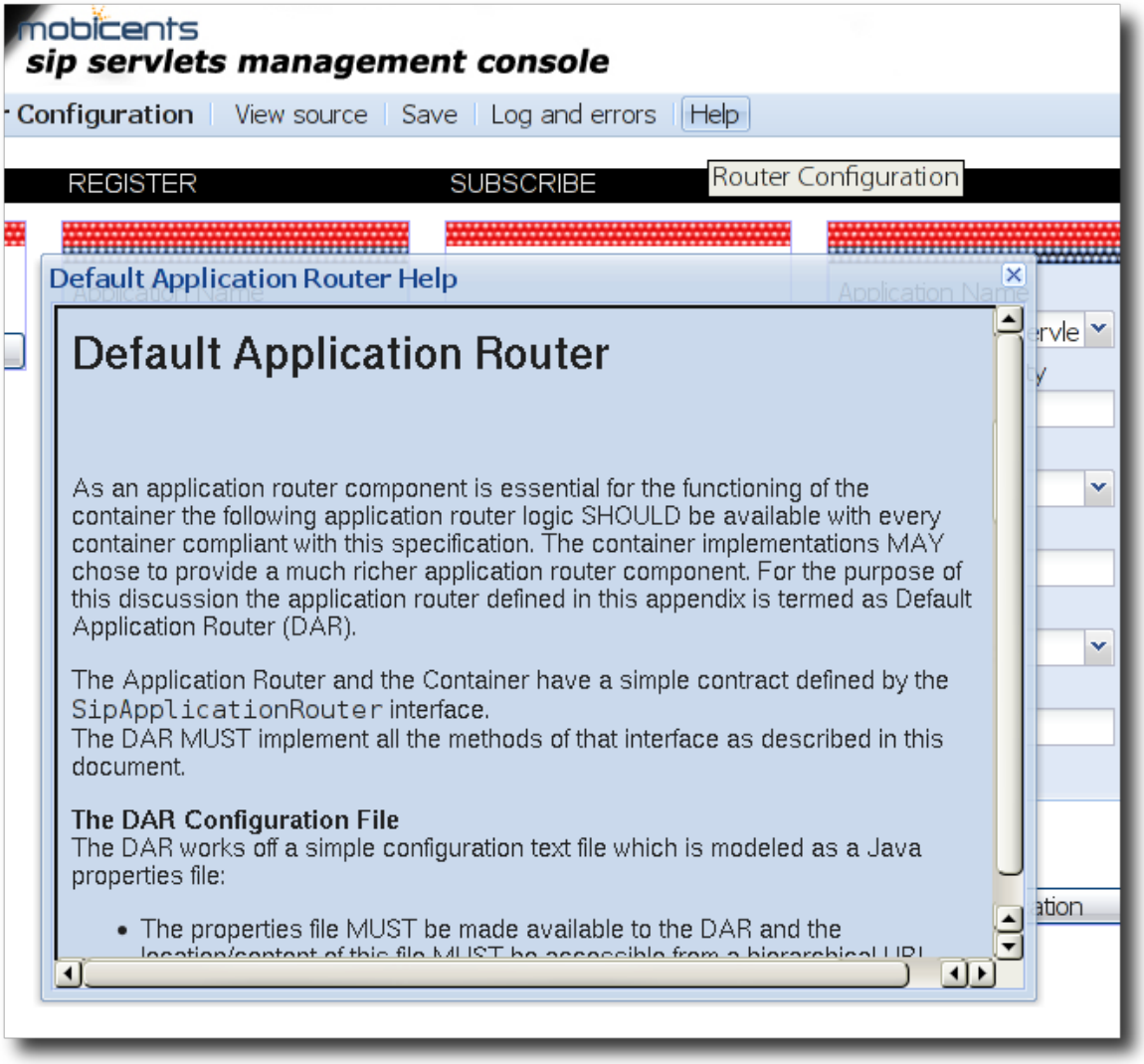


Figure 1.3. SIP Servlets Management Console: Default Application Router Help

Recent versions of the SIP Servlets Management Console feature a **Server Settings** tab, in which concurrency and congestion control settings can be tuned.

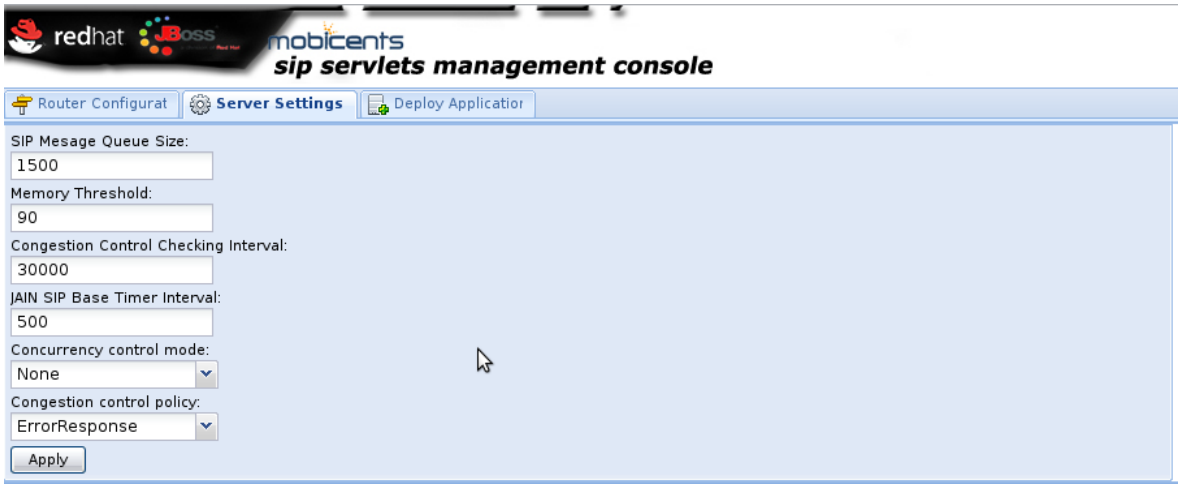


Figure 1.4. Tunable SIP Servlets Server Settings

For more information on concurrency and congestion control tuning, refer to *Configuring the Concurrency and Congestion Control Settings*.

SIP Servlets Server-Installing, Configuring and Running

2.1. SIP Servlet-Enabled JBoss Application Server: Installing, Configuring and Running

The JBoss Communications SIP Servlets Server can run on either the JBoss Application Server or the Tomcat Servlet Container. This section details how to install the SIP Servlets Server on top of the JBoss Application Server. For installation instructions for the Tomcat Servlet Container, refer to *Section 2.2, "SIP Servlet-Enabled Tomcat Servlet Container: Installing, Configuring and Running"*

Note

It is recommended that the SIP Servlets Server is run on the JBoss platform. Some functionality, including the ability to execute some SIP Extension examples, is not available in the Tomcat version.

Differences Between a Standard JBoss Installation and the JBoss Communications SIP Servlets Version. Provided here is a list of differences between a standard JBoss Application Server installation one customized for SIP Servlets. The differences include:

- The `server/default/deploy` directory contains both HTTP and SIP Servlet applications.
- The `server/default/deploy/jbossweb.sar` units have been modified to provide extended classes to the standard JBoss container classes, in order to allow SIP applications to be loaded and the SIP stack to be started.
- The `server/default/deploy/jbossweb.sar` `context.xml` files have been modified to allow the extended manager to manage SIP sessions and SIP application sessions in addition to HTTP sessions.
- The `server/default/deploy/jbossweb.sar/` `server.xml` file has been modified to provide extended classes to common JBoss Web containers. The classes allow SIP applications to be loaded, and the SIP stack to be started.
- The `server/default/deploy/jbossweb.sar/` `jboss-beans.xml` file has been modified to allow the JBoss container to process SIP messages.
- The `server/default/deployers/` `metadata-deployer-jboss-beans.xml` file has been modified to allow JBoss to parse sip.xml deployment descriptors and SIP metadata annotations.

- The `server/default/deployers/jbossweb.deployer/META-INF/war-deployers-jboss-beans.xml` file have been modified so that it is now possible for JBoss containers to correctly deploy SIP servlets and converged applications.
- A `dars` directory containing all of the Default Application Router (DAR) properties files for using the various SIP Servlets applications (which come bundled with the release) has been added to the `server/default/conf` directory.
- Additional JAR files have been added to enable SIP Servlet functionality; these are located in the `server/default/deploy/jbossweb.sar/`, `server/default/deployers/jbossweb.deployer/` and `common/lib`directories.

2.1.1. Java Development Kit (JDK): Installing, Configuring and Running

The **JBoss Communications Platform** is written in Java; therefore, before running any **JBoss Communications** server, you must have a working Java Runtime Environment (JRE) or Java Development Kit (JDK) installed on your system. In addition, the JRE or JDK you are using to run **JBoss Communications** must be version 5 or higher[1].

Should I Install the JRE or JDK? Although you can run **JBoss Communications** servers using the Java Runtime Environment, we assume that most users are developers interested in developing Java-based, **JBoss Communications**-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.

[1] At this point in time, it is possible to run most **JBoss Communications** 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 JBoss Communications web site, forums or discussion pages if you need to inquire about the status of running the XML Document Management Server with Java 6.

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 <*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 **JBoss Communications** in a production environment.

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

Procedure 2.1. Installing the JDK on Linux

- 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 *ftp://jpackage.hmdc.harvard.edu/JPackage/1.7/generic/RPMS.non-free/*.

Important

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 2.2. Installing the JDK on Windows

- 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 the `JAVA_HOME` Environment Variable on Linux. You can determine whether `JAVA_HOME` is set on your system by `echo`ing it on the command line:

```
~]$ echo $JAVA_HOME
```

If `JAVA_HOME` is not set already, then you must set its value to the location of the JDK installation on your system. You can do this by adding two lines to your personal `~/.bashrc` configuration file. Open `~/.bashrc` (or create it if it doesn't exist) and add a line similar to the following one anywhere inside the file:

```
export JAVA_HOME="/usr/lib/jvm/jdk1.5.0_<version>"
```

You should also set this environment variable for any other users who will be running **JBoss Communications** (any environment variables `exported` from `~/.bashrc` files are local to that user).

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

Selecting the Correct System JVM on Linux using `alternatives`. On systems with the `alternatives` command, including Red Hat Enterprise Linux and Fedora, you can easily choose which JDK (or JRE) installation you wish to use, as well as which `java` and `javac` executables should be run when called.

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

```
root@localhost ~]$ /usr/sbin/alternatives --config java

There are 3 programs which provide 'java'.

  Selection    Command
-----------------------------------------------
   1           /usr/lib/jvm/jre-1.5.0-gcj/bin/java
   2           /usr/lib/jvm/jre-1.6.0-sun/bin/java
*+ 3           /usr/lib/jvm/jre-1.5.0-sun/bin/java


Enter to keep the current selection[+], or type selection number:
```

In our case, we want to use the Sun JDK, version 5, that we downloaded and installed, to run the `java` executable. In the `alternatives` information printout above, a plus (+) next to a number indicates the one currently being used. As per `alternatives`' instructions, pressing **Enter** will simply keep the current JVM, or you can enter the number corresponding to the JVM you would prefer to use.

Repeat the procedure above for the `javac` command and the `java_sdk_1.5.0` environment variable, *as the root user*:

```
~]$ /usr/sbin/alternatives --config javac
```

```
~]$ /usr/sbin/alternatives --config java_sdk_1.5.0
```

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`.

2.1.2. Pre-install Requirements and Prerequisites

Hardware Requirements

Sufficient Disk Space
: Once unzipped, version 1.6.0.FINAL of the JBCP SIP Servlets for JBoss binary release requires a minimum of 20MB free disk space.

Anything Java Itself Will Run On
: JBCP SIP Servlets for JBoss is 100% Java and will run on the same hardware that the JBoss Application Server runs on.

Software Prerequisites

JDK 5 or Higher
: A working installation of the Java Development Kit (<acronym>JDK</acronym>) version 5 or higher is currently required in order to run JBCP SIP Servlets for JBoss binary distribution. For instructions on how to install the JDK, refer to *Section 2.1.4, "Installing"*

2.1.3. Downloading

The latest version of JBoss Communications SIP Servlets for JBoss is available from *http://www.mobicents.org/mss-downloads.html*. The top row of the table contains the latest version.

Each version of the SIP Servlets Server is comprised of two separate binary distribution files: one which is JBCP SIP Servlets for JBoss, and the other which is JBCP SIP Servlets for Tomcat. Download SIP Servlets Server for JBoss and continue with the following instructions.

2.1.4. Installing

Once the requirements and prerequisites have been met and you have downloaded the binary distribution zip file, you are ready to install the JBCP SIP Servlets for JBoss binary distribution. Follow the instructions below for the selected platform, whether Linux or Windows.

Version Numbers

For clarity, the command line instructions presented in this chapter use specific version numbers and directory names. Ensure this information is substituted with the binary distribution's version numbers and file names.

Procedure 2.3. Installing the JBCP SIP Servlets for JBoss Binary Distribution on Linux

It is assumed that the downloaded archive is saved in the home directory, and that a terminal window is open displaying the home directory

1. Create a subdirectory to extract the JBCP SIP Servlets for JBoss files into. For ease of identification, it is recommended that the version number of the binary is included in this directory name.

```
~]$ mkdir "jbcp-ss-jboss-<version>"
```

2. Move the downloaded zip file into the directory.

```
~]$ mv "jbcp-ss-1.0.GA-apache-tomcat-6.0.14-0904221257.zip" "jbcp-ss-
jboss-<version>"
```

3. Move into the directory.

```
~]$ cd "jbcp-ss-jboss-<version>"
```

4. Extract the files into the current directory by executing one of the following commands.

   - Java:

```
jbcp-ss-jboss-<version>]$ jar -xvf "jbcp-ss-1.0.GA-apache-
tomcat-6.0.14-0904221257.zip"
```

   - Linux:

```
jbcp-ss-jboss-<version>]$ unzip "jbcp-ss-1.0.GA-apache-
tomcat-6.0.14-0904221257.zip"
```

Note

You can also use `unzip`'-d <unzip_to_location> to extract the zip file's contents to a location other than the current directory.

5. To free disk space, you may want to delete the zip file once you've extracted its contents:

```
jbcp-ss-jboss-<version>]$ rm "jbcp-ss-1.0.GA-apache-
tomcat-6.0.14-0904221257.zip"
```

Procedure 2.4. Installing the JBCP SIP Servlets for JBoss Binary Distribution on Windows

For this procedure, it is assumed that the downloaded archive is saved in the `My Downloads` folder.

1. Create a directory in `My Downloads` to extract the zip file's contents into. For ease of identification, it is recommended that the version number of the binary is included in the folder name. For example, `JBCP SIP Servlets-jboss-<version>`.
2. Extract the contents of the archive, specifying the destination folder as the one created in the previous step.
3. Alternatively, execute the `jar -xvf` command to extract the binary distribution files from the zip archive.

    1. Move the downloaded zip file from `My Downloads` to the folder created in the previous step.
    2. Open the Windows Command Prompt and navigate to the folder that contains the archive using the `cd` command
    3. Execute the `jar -xvf` command to extract the archive contents into the current folder.

    ```
    C:\Users\<user>\My Downloads\jbcp-ss-jboss-<version>>jar -xvf "jbcp-
    ss-1.0.GA-apache-tomcat-6.0.14-0904221257.zip"
    ```

4. It is recommended that the folder holding the JBCP SIP Servlets for JBoss files (in this example, the folder named `jbcp-ss-jboss-<version>`) is moved to a user-defined location for storing executable programs. For example, the `Program Files` folder.
5. Consider deleting the archive, if free disk space is an issue.

    ```
    C:\Users\<user>\My Downloads\jbcp-ss-jboss-<version>>delete "jbcp-
    ss-1.0.GA-apache-tomcat-6.0.14-0904221257.zip"
    ```

2.1.5. Setting the JBOSS_HOME Environment Variable

Configuring JBCP SIP Servlets for JBoss consists of setting the `JBOSS_HOME` environment variable and optionally customizing the JBCP SIP Servlets for JBoss server by adding SIP Connectors, configuring the application router, and logging.

After setting `JBOSS_HOME` according to the instructions in the following section, refer to *Section 2.3, "Configuring"* to learn how to configure JBCP SIP Servlets for JBoss.

Alternatively, after having set `JBOSS_HOME`, the JBCP SIP Servlets for JBoss server can be run. Return to this section to configure it later.

2.1.5.1. Setting the JBOSS_HOME Environment Variable

The **JBoss Communications Platform** (**JBoss Communications**) is built on top of the **JBoss Application Server** (**JBoss AS**). You do not need to set the `JBOSS_HOME` environment variable to run any of the **JBoss Communications 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 **JBoss Communications Platform** and most JBoss Communications servers are built on top of the **JBoss Application Server** (**JBoss AS**). When the **JBoss Communications Platform** or JBoss Communications servers are built *from source*, then `JBOSS_HOME` *must* be set, because the JBoss Communications 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 JBoss Communications files.

This guide does not detail building the **JBoss Communications Platform** or any JBoss Communications servers from source. It is nevertheless useful to understand the role played by **JBoss AS** and `JBOSS_HOME` in the JBoss Communications 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

Important

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 **JBoss Communications Platform** binary distribution.
- ...you have installed a JBoss Communications server binary distribution *which bundles* ***JBoss AS***.

You *MUST* set JBOSS_HOME if...

- ...you are installing the **JBoss Communications Platform** or any of the JBoss Communications servers *from source*.
- ...you are installing the **JBoss Communications Platform** binary distribution, or one of the JBoss Communications server binary distributions, which *do not* bundle **JBoss AS**.

Naturally, if you installed the **JBoss Communications Platform** or one of the JBoss Communications server binary releases which *do not* bundle **JBoss AS**, yet requires it to run, then you should *install **JBoss AS*** [http://docs.jboss.org/jbossas/docs/Installation_And_Getting_Started_Guide/5/html_single/index.html] 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 **JBoss Communications Platform** or individual JBoss Communications 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 2.5. 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].

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

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

[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.

Note

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

```
~]$ echo $JBOSS_HOME
/home/silas/jbcp-1.2.1/jboss-eap-4.3/jboss-as/
```

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 JBoss Communications Platform or individual JBoss Communications 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*.

2.1.6. Configuring

To configure JBCP SIP Servlets for JBoss, refer to *Section 2.3, "Configuring"*.

2.1.7. Running

To start the server, execute one of the startup scripts in the `bin` directory (on Linux or Windows), or by double-clicking the `run.bat` executable batch file in that same directory (on Windows only). It is recommended that the JBoss Application Server is started using the terminal or Command Prompt because the messages displayed during startup can be used to debug, and subsequently correct, any problems. In the Linux terminal or Command Prompt, a successfully started server will return the following information :

```
22:39:07,598 INFO  [SipApplicationDispatcherImpl] Mobicents Sip Servlets <version> - revision
 <rev number> started.
```

Detailed instructions are given below, arranged by platform.

Procedure 2.6. Running JBCP SIP Servlets for JBoss on Linux

1. Change the working directory to JBCP SIP Servlets for JBoss's installation directory (the one in which the zip file's contents was extracted to)

```
downloads]$ cd "jbcp-ss-jboss-<version>"
```

2. (Optional) Ensure that the `bin/run.sh` start script is executable.

```
jbcp-ss-jboss-<version>]$ chmod +x bin/run.sh
```

3. Execute the `run.sh` Bourne shell script.

```
jbcp-ss-jboss-<version>]$ ./bin/run.sh
```

Note

Instead of executing the Bourne shell script to start the server, the `run.jar` executable Java archive can be executed from the `bin` directory:

```
jbcp-ss-jboss-<version>]$ java -jar bin/run.jar
```

Procedure 2.7. Running JBCP SIP Servlets for JBoss on Windows

There are several ways to start JBCP SIP Servlets for JBoss on Windows. All of the following methods accomplish the same task.

1. Using Windows Explorer, navigate to the `bin` subdirectory in the installation directory.
2. The preferred way to start JBCP SIP Servlets for JBoss from the Command Prompt. The command line interface displays details of the startup process, including any problems encountered during the startup process.

   Open the Command Prompt via the **Start** menu and navigate to the correct folder:

```
C:\Users\<user>My Downloads> cd "jbcp-ss-jboss-<version>"
```

3. Start the JBoss Application Server by executing one of the following files:

   - `run.bat` batch file:

```
C:\Users\<user>My Downloads\jbcp-ss-jboss-<version>>bin\run.bat
```

   - `run.jar` executable Java archive:

```
C:\Users\<user>My Downloads\jbcp-ss-jboss-<version>>java -jar bin
\run.jar
```

2.1.8. Using

Once the server is running, access the SIP Servlets Management Console by opening *http://localhost:8080/sip-servlets-management/*.

2.1.9. Testing

After installation, there should be one pre-configured sample application deployed in the `default` server onfiguration. You can use it to verify that the server is installed and running correctly. The application name is “org.mobicents.servlet.sip.example.SimpleApplication”. From the Sip Servlets Management Console you can make sure it is subscribed to receive `INVITE` and `REGISTER` SIP requests. It is a simple `Click2Call` application allowing SIP registration and calling phones from the Web user interface.

The scenario for this example consists of the following steps:

1. Alice and Bob each register a SIP Softphone
2. Alice clicks on the "Call" link to place a call to Bob
3. Alice's phone rings
4. When Alice picks up her phone, Bob's phone rings
5. When Bob answers his phone, the call is connected
6. When one of them hangs up, the other one is also disconnected

Procedure 2.8. Testing the Click2Call sample application

1. Open up a browser to *http://localhost:8080/click2call/*. If you have no registered SIP clients you will be asked to register at least two.
2. Configure your SIP clients to use the sip servlets server as a register and proxy. (IP address : 127.0.0.1, port: 5080) By default it will accept any password
3. After the registration you will see a table where each cell will initiate a call between the corresponding clients.
4. Close the calls.
5. Navigate to *http://localhost:8080/click2call/simplecall.html*, which is a simplified version that doesn't require registered clients.
6. Enter the URIs of the two SIP phones you just started and click "Submit"

7. The phones should be ringing again. You can pick them up and you will know that the SIP and the HTTP containers are working properly.

2.1.10. Stopping

Detailed instructions for stopping the JBoss Application Server are given below, arranged by platform. If the server is correctly stopped, the following three lines are displayed as the last output in the Linux terminal or Command Prompt:

```
[Server] Shutdown complete
Shutdown complete
Halting VM
```

Procedure 2.9. Stopping JBCP SIP Servlets for JBoss on Linux

1. Change the working directory to the binary distribution's install directory.

```
~]$ cd "jbcp-ss-jboss-<version>"
```

2. (Optional) Ensure that the bin/shutdown.sh start script is executable:

```
jbcp-ss-jboss-<version>]$ chmod +x bin/shutdown.sh
```

3. Run the `shutdown.sh` executable Bourne shell script with the `-S` option (the short option for `--shutdown`) as a command line argument:

```
jbcp-ss-jboss-<version>]$ ./bin/shutdown.sh -S
```

Note

The `shutdown.jar` executable Java archive with the `-S` option can also be used to shut down the server:

```
jbcp-ss-jboss-<version>]$ java -jar bin/shutdown.jar -S
```

Procedure 2.10. Stopping jbcp-ss for JBoss on Windows

- Stopping the JBoss Application Server on Windows consists in executing either the `shutdown.bat` or the `shutdown.jar` executable file in the `bin` subdirectory of the JBCP SIP

Servlets for JBoss binary distribution. Ensure the `-S` option (the short option for `--shutdown`) is included in the command line argument.

```
C:\Users\<user>\My Downloads\jbcp-ss-jboss-<version>>bin\shutdown.bat -S
```

- The `shutdown.jar` executable Java archive with the `-S` option can also be used to shut down the server:

```
C:\Users\<user>\My Downloads\jbcp-ss-jboss-<version>>java -jar bin
\shutdown.jar -S
```

2.1.11. Uninstalling

To uninstall JBCP SIP Servlets for JBoss, delete the directory containing the binary distribution.

2.2. SIP Servlet-Enabled Tomcat Servlet Container: Installing, Configuring and Running

You can also run JBoss Communications SIP Servlets on top of the Apache Tomcat Servlet Container. This section provides information on the requirements and prerequisites for running JBCP SIP Servlets for Tomcat, as well as instructions on how to download, install, configure, run, use, stop, test and uninstall it.

Keep in mind that not all capabilities provided by running JBoss Communications SIP Servlets Server on top of the JBoss Application Server are available with JBCP SIP Servlets for Tomcat. In particular, JBCP SIP Servlets for Tomcat lacks support for both clustering and failover; JBCP SIP Servlets for Tomcat nodes can utilize the SIP load balancer, however.

If you are interested in clustering and failover support, or would rather run the JBoss Communications SIP Servlets Server on top of the JBoss Application Server, go to *Section 2.1, "SIP Servlet-Enabled JBoss Application Server: Installing, Configuring and Running"*.

Differences Between the Standard Tomcat Installation and One Customized for the SIP Servlets Server. Provided here is a list of differences between a standard Tomcat Servlet Container installation and the SIP Servlets Server for Tomcat installation. The differences include:

- The `server.xml` configuration file has been modified to provide extended classes to the standard Tomcat container classes, in order to allow SIP applications to be loaded and the SIP stack started.
- A `dars` directory containing the default applications' router properties files for using the SIP Servlet Click-to-Call application (which comes bundled with the release) has been added to the `conf` directory.
- Additional JAR files which can be found in the `lib` directory have been added to enable SIP Servlet functionality.

Installing the Java Development Kit

2.2.1. Java Development Kit (JDK): Installing, Configuring and Running

The **JBoss Communications Platform** is written in Java; therefore, before running any **JBoss Communications** server, you must have a working Java Runtime Environment (JRE) or Java Development Kit (JDK) installed on your system. In addition, the JRE or JDK you are using to run **JBoss Communications** must be version 5 or higher[3].

Should I Install the JRE or JDK? Although you can run **JBoss Communications** servers using the Java Runtime Environment, we assume that most users are developers interested in developing Java-based, **JBoss Communications**-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 <*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.

[3] At this point in time, it is possible to run most **JBoss Communications** 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 JBoss Communications web site, forums or discussion pages if you need to inquire about the status of running the XML Document Management Server with Java 6.

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 **JBoss Communications** in a production environment.

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

Procedure 2.11. Installing the JDK on Linux

- 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 *ftp://jpackage.hmdc.harvard.edu/JPackage/1.7/generic/RPMS.non-free/*.

Important

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 2.12. Installing the JDK on Windows

- 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 the `JAVA_HOME` Environment Variable on Linux. You can determine whether `JAVA_HOME` is set on your system by `echo`ing it on the command line:

```
~]$ echo $JAVA_HOME
```

If `JAVA_HOME` is not set already, then you must set its value to the location of the JDK installation on your system. You can do this by adding two lines to your personal `~/.bashrc` configuration file. Open `~/.bashrc` (or create it if it doesn't exist) and add a line similar to the following one anywhere inside the file:

```
export JAVA_HOME="/usr/lib/jvm/jdk1.5.0_<version>"
```

You should also set this environment variable for any other users who will be running **JBoss Communications** (any environment variables `exported` from `~/.bashrc` files are local to that user).

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

Selecting the Correct System JVM on Linux using `alternatives` . On systems with the `alternatives` command, including Red Hat Enterprise Linux and Fedora, you can easily choose which JDK (or JRE) installation you wish to use, as well as which `java` and `javac` executables should be run when called.

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

```
root@localhost ~]$ /usr/sbin/alternatives --config java

There are 3 programs which provide 'java'.

  Selection    Command
-----------------------------------------------
   1           /usr/lib/jvm/jre-1.5.0-gcj/bin/java
   2           /usr/lib/jvm/jre-1.6.0-sun/bin/java
*+ 3           /usr/lib/jvm/jre-1.5.0-sun/bin/java
```

```
Enter to keep the current selection[+], or type selection number:
```

In our case, we want to use the Sun JDK, version 5, that we downloaded and installed, to run the `java` executable. In the `alternatives` information printout above, a plus (+) next to a number indicates the one currently being used. As per `alternatives`' instructions, pressing **Enter** will simply keep the current JVM, or you can enter the number corresponding to the JVM you would prefer to use.

Repeat the procedure above for the `javac` command and the `java_sdk_1.5.0` environment variable, *as the root user*:

```
~]$ /usr/sbin/alternatives --config javac
```

```
~]$ /usr/sbin/alternatives --config java_sdk_1.5.0
```

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`.

2.2.2. Pre-Install Requirements and Prerequisites

Hardware Requirements

Sufficient Disk Space

You must have sufficient disk space in order to install the JBCP SIP Servlets for Tomcat binary release. Once unzipped, version 1.6.0.FINAL of the JBCP SIP Servlets for Tomcat binary release requires *at least* 20MB of free disk space. Keep in mind that disk space requirements may change from release to release.

Anything Java Itself Will Run On
: JBCP SIP Servlets for Tomcat is 100% Java. It will run on the same hardware that the Tomcat Servlet Container runs on.

Software Prerequisites

JDK 5 or Higher
: A working installation of the Java Development Kit (JDK) version 5 or higher is required in order to run JBCP SIP Servlets for Tomcat.

2.2.3. Downloading

You can download the latest version of JBCP SIP Servlets for Tomcat from *http://www.mobicents.org/mss-downloads.html*. The top row of the table holds the latest version. Note that each release of the JBoss Communications SIP Servlets Server is comprised of two separate binary distribution files: one which is JBCP SIP Servlets for JBoss, and the other which is JBCP SIP Servlets for Tomcat. Download JBoss Communications SIP Servlets Server for Tomcat and continue with the following instructions.

2.2.4. Installing

Once the requirements and prerequisites have been met and you have downloaded the binary distribution zip file, you are ready to install JBCP SIP Servlets for Tomcat. Follow the instructions below for your platform, whether Linux or Windows.

Use Version Numbers Relevant to Your Installation!

For clarity, the command line instructions presented in this chapter use specific version numbers and directory names. Remember to replace them with version numbers and file names relevant to those you are actually working with.

Procedure 2.13. Installing the JBCP SIP Servlets for Tomcat Binary Distribution on Linux

1. For this example, we'll assume you're currently in your home directory, which is where you downloaded the zip file to. First, create a subdirectory to hold the unzipped JBCP SIP Servlets for Tomcat files. It is good practice to include the version number in this directory name; if you do so, remember to correctly match it with the version of the JBCP SIP Servlets for Tomcat distribution you downloaded.

```
~]$ cd downloads
```

2. In `downloads`, create a subdirectory to hold the unzipped JBCP SIP Servlets for Tomcat files. It is good practice to include the version number in this directory name; if you do so, remember

to correctly match it with the version of the JBCP SIP Servlets for Tomcat binary distribution you downloaded.

```
~]$ mkdir "jbcp-ss-tomcat-<version>"
```

3. Move the downloaded zip file into the directory you have just created:

```
~]$ mv "jbcp-ss-1.0.GA-apache-tomcat-6.0.14-0904221257.zip" "jbcp-ss-
tomcat-<version>"
```

4. Move into that directory:

```
~]$ cd "jbcp-ss-tomcat-<version>"
```

5. Finally, use Java's `jar -xvf` command to extract the contents of the zip file into the current directory, thus completing the install:

```
jbcp-ss-tomcat-<version>]$ jar -xvf "jbcp-ss-1.0.GA-apache-
tomcat-6.0.14-0904221257.zip"
```

- Alternatively, if Linux's `unzip` utility is present on your system or is installable, you can use it in lieu of Java's `jar -xvf` command:

```
jbcp-ss-tomcat-<version>]$ unzip "jbcp-ss-1.0.GA-apache-
tomcat-6.0.14-0904221257.zip"
```

Note

You can also use `unzip`'s `-d` *`<unzip_to_location>`* option to extract the zip file's contents to a location other than the current directory.

6. To free disk space, you may want to delete the zip file once you've extracted its contents:

```
jbcp-ss-tomcat-<version>]$ rm "jbcp-ss-1.0.GA-apache-
tomcat-6.0.14-0904221257.zip"
```

Procedure 2.14. Installing the JBCP SIP Servlets for Tomcat Binary Distribution on Windows

1. For this example, we'll assume that you downloaded the binary distribution zip file to the `My Downloads` folder. First, using Windows Explorer, create a subdirectory in `My Downloads` to extract the zip file's contents into. When you name this folder, it is good practice to include the version number; if you do so, remember to correctly match it with the version of the MSS for

Tomcat binary distribution you downloaded. In these instructions, we will refer to this folder as `jbcp-ss-tomcat-<version>`.

2. Double-click the downloaded zip file, selecting as the destination folder the one you just created to hold the zip file's contents.

   - Alternatively, it is also possible to use Java's `jar -xvf` command to extract the binary distribution files from the zip archive. To use this method instead, first move the downloaded zip file from `My Downloads` to the folder that you just created to hold the SIP Servlets Server files.

   - Then, open the Windows Command Prompt and navigate to the folder holding the archive using the `cd` command.

     Opening the Command Prompt from Windows Explorer

     If you are using Windows Vista®, you can open the Command Prompt directly from Explorer. Hold down the **Shift** key and right-click on either a folder, the desktop, or inside a folder. This will cause an **Open Command Window Here** context menu item to appear, which can be used to open the Command Prompt with the current working directory set to either the folder you opened, or opened it from.

   - Finally, use the `jar -xvf` command to extract the archive contents into the current folder.

     ```
     C:\Users\Me\My Downloads\jbcp-ss-tomcat-<version>>jar -xvf "jbcp-
     ss-1.0.GA-apache-tomcat-6.0.14-0904221257.zip"
     ```

3. At this point, you may want to move the folder holding the JBCP SIP Servlets for Tomcat binary files (in this example, the folder named `jbcp-ss-tomcat-<version>`) to another location. This step is not strictly necessary, but it is probably a good idea to move the installation folder from `My Downloads` to a user-defined location for storing runnable programs. Any location will suffice, however.

4. You may want to delete the zip file after extracting its contents in order to free disk space:

   ```
   C:\Users\Me\My Downloads\jbcp-ss-tomcat-<version>>delete "jbcp-
   ss-1.0.GA-apache-tomcat-6.0.14-0904221257.zip"
   ```

2.2.5. Setting the CATALINA_HOME Environment Variable

Before running the JBoss Communications server you are installing, you should consider if you need to set the `CATALINA_HOME` environment variable. Setting it (or re-setting it) will always work. Whether or not you *need* to set `CATALINA_HOME` depends on the following factors:

- If you are installing a binary JBoss Communications server and `CATALINA_HOME` is not set on your system, then you *do not need* to set it, but doing so will do no harm.
- If you are installing a binary JBoss Communications server and `CATALINA_HOME` is (already) set on your system, then you need to make sure it points to the location of the new JBoss Communications server.
- If you are installing a JBoss Communications server from source which uses the Tomcat servlet container, then you *must* set `CATALINA_HOME`.

The following instructions detail how to set `CATALINA_HOME` on both Linux and Windows.

Procedure 2.15. Setting the `CATALINA_HOME` Environment Variable on Linux

1. The `CATALINA_HOME` environment variable must point to the location of your Tomcat installation. Any JBoss Communications 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 JBoss Communications 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 `<mobicents_server>` for the topmost directory of the JBoss Communications server you installed:

   ```
   export CATALINA_HOME="/home/<username>/<path>/<to>/<mobicents_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 2.16. Setting the CATALINA_HOME Environment Variable on Windows

- The CATALINA_HOME environment variable must point to the location of your Tomcat installation. Any JBoss Communications 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 JBoss Communications 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*.

2.2.6. Configuring

Configuring JBCP SIP Servlets for Tomcat consists in setting the CATALINA_HOME environment variable and then, optionally, customizing your JBCP SIP Servlets for Tomcat container by adding SIP Connectors, configuring the application router, and configuring logging. See *Section 2.3, "Configuring"* to learn what and how to configure JBCP SIP Servlets for Tomcat.

Alternatively, you can simply run your JBCP SIP Servlets for Tomcat container now and return to this section to configure it later.

2.2.7. Running

Once installed, you can run the Tomcat Servlet Container by executing the one of the startup scripts in the bin directory (on Linux or Windows), or by double-clicking the run.bat executable batch file in that same directory (on Windows only). However, we suggest always starting Tomcat using the terminal or Command Prompt because you are then able to read—and act upon—any startup messages, and possibly debug any problems that may arise. In the Linux terminal or Command Prompt, you will be able to tell that the container started successfully if the last line of output is similar to the following:

```
Using CATALINA_BASE: /home/silas/temp/apps/mobicents/sip_servlets_server/mss-tomcat-1.0
Using CATALINA_HOME: /home/silas/temp/apps/mobicents/sip_servlets_server/mss-tomcat-1.0
Using CATALINA_TMPDIR: /home/silas/temp/apps/mobicents/sip_servlets_server/mss-tomcat-1.0/temp
Using JRE_HOME: /etc/java-config-2/current-system-vm
```

Detailed instructions are given below, arranged by platform.

Procedure 2.17. Running JBCP SIP Servlets for Tomcat on Linux

1. Change your working directory to the SIP Servlets-customized Tomcat's topmost directory (the one in which you extracted the zip file's contents to):

```
~]$ cd "jbcp-ss-tomcat-<version>"
```

2. (Optional) Ensure that the `bin/startup.sh` start script is executable:

```
jbcp-ss-tomcat-<version>]$ chmod +x bin/startup.sh
```

3. Finally, execute the `startup.sh` Bourne shell script:

```
jbcp-ss-tomcat-<version>]$ ./bin/startup.sh
```

Procedure 2.18. Running JBCP SIP Servlets for Tomcat on Windows

1. There are several different ways to start the Tomcat Servlet Container on Windows. All of the following methods accomplish the same task.

   Using Windows Explorer, change your folder to the one in which you unzipped the downloaded zip file, and then to the `bin` subdirectory.

2. Although not the preferred way (see below), it is possible to start the Tomcat Servlet Container by double-clicking on the `startup.bat` executable batch file.

   - As mentioned above, the best way to start the Tomcat Servlet Container is by using the Command Prompt. Doing it this way will allow you to view all of the server startup details, which will enable you to easily determine whether any problems were encountered during the startup process. You can open the Command Prompt directly from the `<topmost_directory>\bin` folder in Windows Explorer, or you can open the Command Prompt via the **Start** menu and navigate to the correct folder:

     ```
     C:\Users\Me\My Downloads> cd "jbcp-ss-tomcat-<version>"
     ```

   - Start the Tomcat Servlet Container by running the executable `startup.bat` batch file:

     ```
     C:\Users\Me\My Downloads\jbcp-ss-tomcat-<version>>bin\startup.bat
     ```

2.2.8. Stopping

Detailed instructions for stopping the Tomcat Servlet Container are given below, arranged by platform. Note that if you properly stop the server, you will see the following three lines as the last output in the Linux terminal or Command Prompt (both running and stopping the Tomcat Servlet Container produces the same output):

```
Using CATALINA_BASE:   /home/silas/temp/apps/mobicents/sip_servlets_server/
mss-tomcat-1.0
Using CATALINA_HOME:   /home/silas/temp/apps/mobicents/sip_servlets_server/
mss-tomcat-1.0
Using CATALINA_TMPDIR: /home/silas/temp/apps/mobicents/sip_servlets_server/
mss-tomcat-1.0/temp
Using JRE_HOME:       /etc/java-config-2/current-system-vm
```

Procedure 2.19. Stopping JBCP SIP Servlets for Tomcat on Linux by Executing `shutdown.sh`

1. You can shut down the Tomcat Servlet Container by executing the `shutdown.sh` Bourne shell script in the `<topmost_directory>/bin` directory. To do so, first change your working directory to the binary distribution's topmost directory (the one to which you extracted the downloaded zip file's contents):

   ```
   downloads]$ cd "jbcp-ss-tomcat-<version>"
   ```

2. (Optional) Ensure that the bin/shutdown.sh start script is executable:

   ```
   jbcp-ss-tomcat-<version>]$ chmod +x bin/shutdown.sh
   ```

3. Finally, run the `shutdown.sh` executable Bourne shell script

   ```
   jbcp-ss-tomcat-<version>]$ ./bin/shutdown.sh
   ```

Procedure 2.20. Stopping JBCP SIP Servlets for Tomcat on Windows

- Stopping the Tomcat Servlet Container on Windows consists in executing the `shutdown.bat` executable batch script in the `bin` subdirectory of the SIP Servlets-customized Tomcat binary distribution:

  ```
  C:\Users\Me\My Downloads\jbcp-ss-tomcat-<version>>bin\shutdown.bat
  ```

2.2.9. Using

After starting the server successfully, you can access the default web applications included with JBCP SIP Servlets for Tomcat by opening the following URL in your browser: *http://localhost:8080/* .

You can also access the SIP Servlets Management Console by opening *http://localhost:8080/sip-servlets-management/* in your browser.

2.2.10. Testing

2.2.11. Uninstalling

To uninstall JBCP SIP Servlets for Tomcat, simply delete the directory you decompressed the binary distribution archive into.

2.3. Configuring

2.3.1. Configuring SIP Connectors

SIP Connectors are added by adding a <*Connector*> element under the <*Service*> element in the container's `server.xml` configuration file.

Three SIP Connectors are configured for SIP Servlets:

- UDP and TCP are running on the binding IP Address of the container and port 5080.
- TLS is running on the binding IP Address of the container and port 5081.

Example 2.1. Adding a SIP Connector to server.xml

To add a SIP Connector on the IP address `127.0.0.1`, on port `5080`, using the UDP transport protocol, you should insert the following XML element:

```
 <Connector port="5080"
ipAddress="127.0.0.1"
 protocol="org.mobicents.servlet.sip.startup.SipProtocolHandler"
 signalingTransport="udp"
 useStun="false"
 stunServerAddress="stun01.sipphone.com"
 stunServerPort="3478"
 staticServerAddress="122.122.122.122"
 staticServerPort="44"
 useStaticAddress="true"
 httpFollowsSip="false"/>
```

SIP <connector> Attributes

port
: The port number on which the container will be able to receive SIP messages.

ipAddress
: The IP address at which the container will be able to receive SIP messages. The container can be configured to listen to all available IP addresses by setting *`ipAddress`* to **`0.0.0.0`** **`<sipPathName>`**.

protocol
: Specifies the connector is a SIP Connector and not an HTTP Connector. There is no need to change this property.

signalingTransport
: Specifies the transport on which the container will be able to receive SIP messages. For example, "udp".

useStun
: Enables Session Traversal Utilities for NAT (STUN) support for this Connector. The attribute defaults to "false". If set to "true", ensure that the `ipAddress` attribute is *not* set to `127.0.0.1`. Refer to *Section 7.4, "STUN Support"* for more information about STUN.

stunServerAddress
: Specifies the STUN server address used to discover the public IP address of the SIP Connector. This attribute is only required if the `useStun` attribute is set to "true". Refer to *Section 7.4, "STUN Support"* for more information about STUN and public STUN servers.

stunServerPort
: Specifies the STUN server port of the STUN server used in the `stunServerAddress` attribute. You should rarely need to change this attribute; also, it is only needed if the `useStun` attribute is set to "true". Refer to *Section 7.4, "STUN Support"* for more information about STUN.

useStaticAddress
: Specifies whether the settings in staticServerAddress and staticServerPort are activated. The default value is "false" (deactivated).

staticServerAddress
: Specifies what load-balancer server address is inserted in Contact/Via headers for server-created requests. This parameter is useful for cluster configurations where requests should be bound to a load-balancer address, rather than a specific node address.

staticServerPort
: Specifies the port of the load-balancer specified in staticServerAddress. This parameter is useful in cluster configurations where requests should be bound to a load-balancer address rather than a specific node address.

httpFollowsSip
: Makes the application server aware of how the SIP Load Balancers assign request affinity, and stores this information in the application session.

  When the HTTP Load Balancer sends HTTP requests that are not associated with the application session, the application server will force the HTTP request to be repeated until it lands on the correct node.

Note

A comprehensive list of implementing classes for the SIP Stack is available from the *Class SipStackImpl page on nist.gov* [http://ci.jboss.org/jenkins/job/jain-sip/lastSuccessfulBuild/artifact/javadoc/javax/sip/SipStack.html].

2.3.2. Application Routing and Service Configuration

The application router is called by the container to select a SIP Servlet application to service an initial request. It embodies the logic used to choose which applications to invoke. An application router is required for a container to function, but it is a separate logical entity from the container.

The application router is responsible for application selection and must not implement application logic. For example, the application router cannot modify a request or send a response.

For more information about the application router, refer to the following sections of the *JSR 289 specification* [http://jcp.org/en/jsr/detail?id=289]: Application Router Packaging and Deployment, Application Selection Process, and Appendix C.

In order to configure the application router, you should edit the `Service` element in the container's `server.xml` configuration file:

Example 2.2. Configuring the Service Element in the Container's server.xml

```
 <Service name="Sip-Servlets"
className="org.mobicents.servlet.sip.startup.SipStandardService"
sipApplicationDispatcherClassName="org.mobicents.servlet.sip.core.SipApplicationDispatcherImpl"
usePrettyEncoding="false"
additionalParameterableHeaders="Header1,Header2"
bypassResponseExecutor="false"
bypassRequestExecutor="false"
baseTimerInterval="500"
t2Interval="4000"
t4Interval="5000"
timerDInterval="32000"
dispatcherThreadPoolSize="4"
darConfigurationFileLocation="file:///home/silas/workspaces/mobicents-sip-servlets/
sip-servlets-examples/reinvite-demo/reinvite-dar.properties"
sipStackPropertiesFile="conf/mss-sip-stack.properties"
dialogPendingRequestChecking="false">
```

SIP Service element attributes

className
: This attribute specifies that the servlet container is a *converged* (i.e. SIP + HTTP) servlet container.

sipApplicationDispatcherClassName
: This attribute specifies the class name of the `org.mobicents.servlet.sip.core.SipApplicationDispatcher` implementation to use. The routing algorithm and application selection process is performed in that class.

darConfigurationFileLocation
: The default application router file location. This is used by the default application router to determine the application selection logic. Refer to Appendix C of the JSR 289 specification for more details.

sipStackPropertiesFile
: Specifies the location of the file containing key value pairs corresponding to the SIP Stack configuration properties. This attribute is used to further tune the JAIN SIP Stack. If this property is omitted, the following default values are assumed:

  - gov.nist.javax.sip.LOG_MESSAGE_CONTENT=true
  - gov.nist.javax.sip.TRACE_LEVEL=32
  - gov.nist.javax.sip.DEBUG_LOG=logs/mss-jsip-debuglog.txt
  - gov.nist.javax.sip.SERVER_LOG=logs/mss-jsip-messages.xml
  - javax.sip.STACK_NAME=Mobicents-SIP-Servlets
  - javax.sip.AUTOMATIC_DIALOG_SUPPORT=off
  - gov.nist.javax.sip.DELIVER_UNSOLICITED_NOTIFY=true
  - gov.nist.javax.sip.THREAD_POOL_SIZE=64
  - gov.nist.javax.sip.REENTRANT_LISTENER=true
  - gov.nist.javax.sip.MAX_FORK_TIME_SECONDS=0. Dialog forking is not enabled by default as it has an impact on memory. If set to a value greater than 0, Dialog Forking will be enabled on Mobicents Sip Servlets.
  - gov.nist.javax.sip.AUTOMATIC_DIALOG_ERROR_HANDLING=false. Merged requests Loop Detection is turned off by default

  SIP Servlets also adds its own properties to allow for even more configuration and flexibility:

  - If the property `org.mobicents.servlet.sip.SERVER_HEADER` is set, a server header will be added to all SIP Responses leaving the container.

- If the property `org.mobicents.servlet.sip.USER_AGENT_HEADER` is set, a server header will be added to all SIP Requests leaving the container.

usePrettyEncoding
: Allows Via, Route, and RecordRoute header field information to be split into multiple lines, rather than each header field being separating with a comma. The attribute defaults to "true". Leaving this attribute at the default setting may assist in debugging non-RFC3261 compliant SIP servers.

additionalParameterableHeaders
: Comma separated list of header names that are treated as parameterable by the container. The specified headers are classed as valid, in addition to the standard parameterable headers defined in the Sip Servlets 1.1 Specification.

  Setting and getting parameters is allowed for both the standard and the additional parameters. Parameters that are not specified in `additionalParameterableHeaders` will result in a ServletParseException error.

bypassRequestExecutor/bypassResponseExecutor
: If set to false, the SIP Servlets server uses a ThreadPoolExecutor linked to a LinkedBlockingQueue to dispatch the request/response threads. The container can then handle two different responses (for example a 180 Ringing and a 200 OK) concurrently. However, a race condition can occur where the second response overtakes the first one (200 OK dispatched to the application before the 180 Ringing) on UDP.

  These flags are set to true by defauilt, Jain sip serializing is used per transaction to ensure such race conditions don't occur in Mobicents Sip Servlets even though they can still happen on UDP at jain sip level.

baseTimerInterval
: Specifies the `T1` Base Timer Interval, which allows the SIP Servlets container to adjust its timers depending on network conditions. The default interval is 500 (milliseconds).

  For more information about available timers, refer to the *RFC326 "Table of Timer Values"1* [http://tools.ietf.org/html/rfc3261#appendix-A], and the document contained in the *3GPP-IMS TS 24.229 v9.1.0 specification ZIP archive* [http://www.3gpp.org/ftp/Specs/html-info/24229.htm].

  All of the timers present in the tables depend on `T1`, `T2`, `T4`, and `Timer D`.

t2Interval
: Specifies the `T2` Interval, which allows the SIP Servlets container to adjust its timers depending on network conditions. The default interval is 4000 (milliseconds).

  For more information about available timers, refer to the *RFC326 "Table of Timer Values"1* [http://tools.ietf.org/html/rfc3261#appendix-A], and the document contained in the *3GPP-IMS TS 24.229 v9.1.0 specification ZIP archive* [http://www.3gpp.org/ftp/Specs/html-info/24229.htm].

All of the timers present in the tables depend on `T1`, `T2`, `T4`, and `Timer D`.

t4Interval

Specifies the `T4` Interval, which allows the SIP Servlets container to adjust its timers depending on network conditions. The default interval is 5000 (milliseconds).

For more information about available timers, refer to the *RFC326 "Table of Timer Values"1* [http://tools.ietf.org/html/rfc3261#appendix-A], and the document contained in the *3GPP-IMS TS 24.229 v9.1.0 specification ZIP archive* [http://www.3gpp.org/ftp/Specs/html-info/24229.htm].

All of the timers present in the tables depend on `T1`, `T2`, `T4`, and `Timer D`.

timerDInterval

Specifies the `Timer D` Interval, which allows the SIP Servlets container to adjust its timers depending on network conditions. The default interval is 32000 (milliseconds).

For more information about available timers, refer to the *RFC326 "Table of Timer Values"1* [http://tools.ietf.org/html/rfc3261#appendix-A], and the document contained in the *3GPP-IMS TS 24.229 v9.1.0 specification ZIP archive* [http://www.3gpp.org/ftp/Specs/html-info/24229.htm].

All of the timers present in the tables depend on `T1`, `T2`, `T4`, and `Timer D`.

dispatcherThreadPoolSize

The number of threads used for processing SIP messages inside the Sip Servlets container by the dispatcher. The default value is 4.

dialogPendingRequestChecking

This property enables and disables error checking when SIP transactions overlap. If within a single dialog an INVITE request arrives while there is antoher transaction proceeding, the container will send a 491 error response. The default value is false.

dnsServerLocatorClass

Specifies the `org.mobicents.ext.javax.sip.dns.DNSServerLocator` implementation class that will be used by the container to perform DNS lookups compliant with RFC 3263 : Locating SIP Servers and E.164 NUmber Mapping. The default class used by the container is `org.mobicents.ext.javax.sip.dns.DefaultDNSServerLocator`, but any class implementing the `org.mobicents.ext.javax.sip.dns.DNSServerLocator` interface. To disable DNS lookups, this attribute should be left empty.

addressResolverClass

Specifies the `gov.nist.core.net.AddressResolver` implementation class that will be used by the container to perform DNS lookups. The default class used by the container is `org.mobicents.servlet.sip.core.DNSAddressResolver`, but any class implementing the `gov.nist.core.net.AddressResolver` NIST SIP Stack interface and having a constructor with a `org.mobicents.servlet.sip.core.SipApplicationDispatcher` parameter can be used. To disable DNS lookups, this attribute should be left empty.

This attribute has been deprecated in favor of `dnsServerLocatorClass` attribute which provides better compliance with RFC 3263 : Location SIP Servers and ENUM support

2.3.3. SIP Servlets Server Logging

There are two separate levels of logging:

- Logging at the container level, which can be configured using the `log4j.xml` configuration file, usually located in the container's `lib` directory.
- Logging of the NIST SIP stack, which is configured in the `Connector` element of the container's `server.xml` configuration file.

You can setup the logging so that the NIST SIP Stack will log into the container logs.

To use LOG4J in NIST SIP Stack, you need to define a category in `JBOSS_HOME/server/default/conf/jboss-log4j.xml` and set it to `off`.

Example 2.3. Configuring the NIST SIP Stack to log into the Container's logs

```
<category name="gov.nist">
<priority value="OFF"/>
</category>
```

For this category to be used in MSS, you need to specify it in `JBOSS_HOME/server/default/conf/mss-sip-stack.properties`, add the `gov.nist.javax.sip.LOG4J_LOGGER_NAME=gov.nist` property, and set the `gov.nist.javax.sip.TRACE_LEVEL=LOG4J` property.

Application Router

Application Routing is performed within the JBoss Communications Sip Servlets container by the Default Application Router. The following sections describe the Default Application Router, and how other JSR 289 compliant Application Router implementations can be installed.

3.1. Default Application Router

The Application Router is called by the container to select a SIP Servlet application to service an initial request. It embodies the logic used to choose which applications to invoke.

3.1.1. Role of the Application Router

An Application Router is required for a container to function, but it is a separate logical entity from the container. The Application Router is responsible for application selection and does not implement application logic. For example, the Application Router cannot modify a request or send a response.

There is no direct interaction between the Application Router and applications, only between the SIP Servlets Container and the Application Router.

The SIP Servlets container is responsible for passing the required information to the Application Router within the initial request so the Application Router can make informed routing decisions. The Application Router is free to make use of any information or data stores, except for the information passed by the container. It is up to the individual implementation how the Application Router makes use of the information or data stores.

The deployer in a SIP Servlet environment controls application composition by defining and deploying the Application Router implementation. Giving the deployer control over application composition is desirable because the deployer is solely responsible for the services available to subscribers.

Furthermore, the SIP Servlets specification intentionally allows the Application Router implementation to consult arbitrary information or data stores. This is because the deployer maintains subscriber information and this information is often private and valuable.

3.1.2. JBoss Communications Default Application Router

JBoss Communications SIP Servlets provides an implementation of the Default Application Router (DAR) as defined in the SIP Servlets 1.1 specification, Appendix C.

3.1.2.1. The DAR Configuration File

The Default Application Router (DAR) obtains its operational parameters from a configuration text file that is modeled as a Java properties file. The configuration file contains the information needed by the Application Router to select which SIP Servlet application will handle an incoming initial request.

In the case of JBoss Communications SIP Servlets, it is also possible to configure the DAR through the `server.xml` configuration file (see *Example 2.2, "Configuring the Service Element in the Container's server.xml"* and *Section 1.2, "Working with the SIP Servlets Management Console"*).

The properties file has the following characteristics and requirements:

- It must be made available to the DAR.
- It must allow the contents and file structure to be accessible from a hierarchical URI supplied as a system property `javax.servlet.sip.ar.dar.configuration`.
- It is first read by the container when it loads up and is refreshed each time an application is deployed and undeployed.
- It has a simple format in which the name of the property is the SIP method and the value is a comma-separated string value for the `SipApplicationRouterInfo` object.

```
INVITE: (sip-router-info-1), (sip-router-info-2)..
SUBSCRIBE: (sip-router-info-3), (sip-router-info-4)..
ALL: (sip-router-info-5), (sip-router-info-6)..
```

  JBoss Communications SIP Servlets defines a new keyword called `ALL`. The keyword allows mapping between the sip-router-info data, and all methods supported by the container (for example, INVITE, REGISTER, SUBSCRIBE). This mapping can save time when configuring an application that listens to all incoming methods.

Note

If `ALL`, and a specific method are defined in the DAR file, the specific method takes precedence over `ALL`. When the specific method no longer has applications to serve, `ALL` is enabled again.

The sip-router-info data specified in the properties file is a string value version of the `SipApplicationRouterInfo` object. It consists of the following information:

- The name of the application as known to the container. The application name can be obtained from the <app-name> element of the `sip.xml` deployment descriptor of the application, or the `@SipApplication` annotation.
- The identity of the subscriber that the DAR returns. The DAR can return any header in the SIP request using the DAR directive `DAR:SIP_HEADER`. For example, `DAR:From` would return the SIP URI in the `From` header. The DAR can alternatively return any string from the SIP request.
- The routing region, which consists of one of the following strings: `ORIGINATING`, `TERMINATING` or `NEUTRAL`. This information is not currently used by the DAR to make routing decisions.

- A SIP URI indicating the route as returned by the Application Router, which can be used to route the request externally. The value may be an empty string.
- A route modifier, which consists of one of the following strings: `ROUTE`, `ROUTE_BACK` or `NO_ROUTE`. The route modifier is used in conjunction with the route information to route a request externally.
- A string representing the order in which applications must be invoked (starts at 0). The string is removed later on in the routing process, and substituted with the order positions of sip-router-info data.
- An optional string that contains JBoss Communications-specific parameters. Currently, only the `DIRECTION` and `REGEX` parameters are supported.

Note

The field can contain unsupported `key=value` properties that may be supported in future releases. The unsupported properties will be ignored during parsing, until support for the attributes is provided.

The syntax is demonstrated in *Example 3.1, "DIRECTION Example"*.

- The `DIRECTION` parameter specifies whether an application serves external(INBOUND) requests or initiates (OUTBOUND) requests.

  If an application is marked `DIRECTION=INBOUND`, it will not be called for requests initiated by applications behaving as UAC. To mark an application as UAC, specify `DIRECTION=INBOUND` in the optional parameters in the DAR.

  Applications that do not exist in the DAR list for the container are assumed to be `OUTBOUND`. Because undefined applications are incapable of serving external requests, they must have self-initiated the request. The Sip Servlets Management Console can be used to specify the `DIRECTION` parameter.

- The `REGEX` parameter specifies a regular expression to be matched against the initial request passed to the Application Router.

  If the regular expression matches a part of the initial request, the application is called. If it does not, it is skipped.

  For example, in the following sip-router-info data:

```
INVITE - ("org.mobicents.servlet.sip.testsuite.SimpleApplication",
 "DAR:From", "ORIGINATING", "", "NO_ROUTE", "0",
 "REGEX=From:.*sip:.*@sip-servlets\.com")
```

  - only incoming initial requests with a From Header with a SIP URI that belongs to the sip-servlets.com domain will be passed to the SimpleApplication.

Example 3.1. DIRECTION Example

In this example, two applications are declared for the `INVITE` request. The `LocationServiceApplication` is called for requests coming from outside the container, but it will not be called for the requests initiated by the UAC application `Click2DialApplication`.

```
INVITE: ("org.mobicents.servlet.sip.testsuite.Click2DialApplication",
 "DAR:From",
"ORIGINATING", "", "NO_ROUTE", "0", "DIRECTION=OUTBOUND"), \
("org.mobicents.servlet.sip.testsuite.LocationServiceApplication", "DAR
\:From",
"ORIGINATING", "", "NO_ROUTE", "0", "DIRECTION=INBOUND")
```

This type of configuration is useful in cases where different application must be responsible for both requests initiated by the container, and external requests received by the container.

Example 3.2. ORIGINATING/TERMINATING DAR Example

In this example, the DAR is configured to invoke two applications on receipt of an INVITE request; one each in the originating and the terminating halves. The applications are identified by their application deployment descriptor names.

```
INVITE: ("OriginatingCallWaiting", "DAR:From", "ORIGINATING", "",
 "NO_ROUTE", "0"), ("CallForwarding", "DAR:To", "TERMINATING",
 "","NO_ROUTE", "1")
```

For this example, the returned subscriber identity is the URI from each application's `From` and `To` headers respectively. The DAR does not return any route to the container, and maintains the invocation state in the `stateInfo` as the index of the last application in the list.

3.1.2.2. Routing of SIP Messages to Applications

3.1.2.2.1. Initial Requests and Application Selection Process

Initial Requests are those that can essentially be dialog creating (such as, `INVITE`, `SUBSCRIBE` and `NOTIFY`), and not part of an already existing dialog.

Initial requests are routed to applications deployed in the container according to the SIP Servlets 1.1 specification, Section 15.4.1 Procedure for Routing an Initial Request.

Note

There are some other corner cases that apply to initial requests. Refer to Appendix B, Definition of an Initial Request in the SIP Servlets 1.1 specification.

Example 3.3. INVITE Routing

The following example describes how the DAR routes an INVITE to two applications deployed in a container. The applications in this example are a Location Service and a Call Blocking application.

In the example, the assumption of a request coming to the server is described. However, applications can act as a UAC, and generate initial requests on their own. For routing purposes, it is not necessary for the specified application initiating the request to have an entry in the DAR file.

The DAR file contains the required information for the two applications to be invoked in the correct order.

```
INVITE: ("LocationService", "DAR:From", "ORIGINATING", "", "NO_ROUTE", "0"),
 ("CallBlocking", "DAR:To", "TERMINATING", "","NO_ROUTE", "1")
```

Processing occurs in the following order:

1. A new `INVITE` (not a re-INVITE) arrives at the container.

   The `INVITE` is a dialog creating request, and is not part of any dialog.

2. The Application Router is called.

   From the `INVITE` information, the first application to invoke is the Location Service.

3. The Application Router returns the application invocation order information to the container (along with the rest of the sip-router-info data) so the container knows which application to invoke.

4. The container invokes the LocationService that proxies the `INVITE`.

   The proxied `INVITE` is considered as a new `INVITE` to the known IP Address of the registered user for the Request URI

   For further information regarding `INVITE` handling, refer to "Section 15.2.2 Sending an Initial Request" in the SIP Servlets 1.1 Specification.

5. Because the `INVITE` has been proxied, the container invokes the Application Router for the proxied `INVITE` to see if any more applications are interested in the event.

6. From the proxied invite, the Application Router determines that the second application to invoke is the Call Blocking application.

7. The Application Router returns information regarding the Call Blocking application to the container (along with the rest of the sip-router-info data) so the container knows which application to invoke.

8. The container routes the `INVITE` for the Call Blocking application to the next application in the chain.

9. The Call Blocking application determines that the user that initiated the call is black listed. The application rejects the call with a "Forbidden" response.

10. Because the Call Blocking application acts as a UAS, the Application Selection Process is stopped for the original `INVITE`.

The path the `INVITE` has taken (that is, LocationService to CallBlocking) is called the application path. The routing of the responses will now occur as explained in the next section.

3.1.2.2.2. Response Routing

Responses always follow the reverse of the path taken by the corresponding request. In our case, the Forbidden response will first go back to the LocationService, and then back to the caller. This is true for responses to both initial and subsequent requests. The application path is a logical concept and as such may or may not be explicitly represented within containers.

Another possible outcome could have been that the Call Blocking application, instead of sending a Forbidden response, allowed the call and proxied the INVITE to the same Request URI chosen by the Location Service. Then when the callee sends back the 200 OK Response, this response goes back the same way through the application path (so in the present case Call Blocking, then Location Service, then back to the caller).

Note

The Call Blocking application cannot just do nothing with the request and expect the container to route the request in its place (either to a next application in chain if another one is present or to the outside world if none is present). The Application has to do something with request (either proxy it or act as a UAS).

3.1.2.2.3. Subsequent Requests

Subsequent requests are all requests that are not Initial.

The second scenario, where the Call Blocking application allowed the call, will be used in this section to showcase subsequent requests. The caller has received the 200 OK response back. Now, according to the SIP specification (RFC 3261), it sends an ACK. The ACK arrives at the container, and is not a dialog creating request and is already part of an ongoing dialog (early dialog) so the request is detected as a Subsequent request and will follow the application path created by the initial request. The ACK will go through Location Service, Call Blocking, and finally to the callee.

3.1.3. Limitations of the Default Application Router

The DAR is a minimalist Application Router implementation that is part of the reference implementation. While it could be used instead of a production Application Router, it offers no processing logic except for the declaration of the application order.

In real world deployments, the Application Router plays an extremely important role in application orchestration and composition. It is likely that the Application Router would make use of complex rules and diverse data repositories in future implementations.

3.2. DFC Application Router

3.2.1. Description of DFC Application Router

Instead of using the JBoss Communications Default Application Router, any SIP Servlets 1.1 compliant Application Router can be used, including the eCharts *DFC Application Router* [http://echarts.org/EChartsSipServletManual/sip-echartsse4.html#x6-140004.1].

3.2.2. Installing the DFC Application Router

Detailed instructions are available from *the eCharts website* [http://echarts.org/Blog/Running-E4SS-apps-on-Mobicents-SIP-Servlets.html]. The following procedure describe how to install the eCharts DFC Application Router (DFCAR) on a variety of SIP Servlet Server platforms.

Procedure 3.1. Installing DFCAR on Tomcat

1. **Deploy the DFCAR**

   Drop the `dfcar.jar` from the ECharts distribution package in the `TOMCAT_HOME/lib` directory.

2. **Remove the DAR**

   Remove the JBoss Communications Default Application Router located in `TOMCAT_HOME/lib/sip-servlets-application-router-*.jar`.

Procedure 3.2. Installing DFCAR on JBoss 5.1.x

1. **Deploy the DFCAR**

   Drop the `dfcar.jar` from the ECharts distribution package in the `JBOSS_HOME/server/default/deploy/jbossweb.sar/` directory.

2. **Remove the DAR**

   Remove the JBoss Communications Default Application Router located in `JBOSS_HOME/server/default/deploy/jbossweb.sar/sip-servlets-application-router-*.jar`.

SIP Servlet Example Applications

The SIP Servlet Server has a selection of examples that demonstrate particular capabilities of the server. *Table 4.1, "Available Examples"* lists the available examples, their location, and a brief description about the functionality each example demonstrates. The examples can also provide a useful starting point for developing SIP Applications, therefore it is encouraged to experiment and adapt the base examples. Each example is available in both binary and source formats.

Table 4.1. Available Examples

Example	Description
Section 4.1.3, "The Call-Blocking Service"	Demonstrates how to block calls by specifying that the INVITE SIP Extension checks the `From` address to see if it is specified in the block list. If the blocked SIP address matches, the Call Blocking application send a FORBIDDEN response.
Section 4.1.4, "The Call-Forwarding Service"	Demonstrates how to forward calls by specifying that the INVITE SIP Extension checks the `To` address to see if it is specified in the forward list. If the SIP address matches, the application acts as a back-to-back user agent (B2BUA).
Section 4.1.5, "The Call-Controller Service"	Call Blocking and Call Forwarding are merged to create a new service.
Speed Dial [http://www.mobicents.org/speed_dial.html]	Demonstrates how to implement speed dialing for SIP addresses. The demonstration uses a static list of speed dial numbers. The numbers are translated into a complete address based on prior configuration. The SIP addresses are proxied without record-routing, or supervised mode.
Section 4.1.1, "The Location Service"	Demonstrates a location service that performs a lookup based on the request URI, into a hard-coded list of addresses. The request is proxied to the set of destination addresses associated with that URI.
Composed Speed Dial and Location [http://www.mobicents.org/speeddial_locationservice.html]	Speed Dial and Location are merged to create a new service. Speed Dial proxies the speed dial number to a SIP address, then Location Service proxies the call to the actual location of the call recipient.

Example	Description
Click to Call [http://www.mobicents.org/click2call.html]	Demonstrates how SIP Servlets can be used along with HTTP servlets as a converged application to place calls from a web portal. The example is a modified version of the click to dial example from the Sailfin project, but has been reworked to comply with JSR 289.
Chat Server [http://www.mobicents.org/chatserver.html]	Demonstrates `MESSAGE` SIP Extension support. This example is based on the chatroom server demonstration from the BEA dev2dev project, and has been modified to meet JSR 289 requirements.
Media JSR 309 Example [http://www.mobicents.org/mss-jsr309-demo.html]	Demonstrates how the Sip Servlets Application Developers can leverage the JSR-309 API, which provides to application developers multimedia capabilities with a generic media server (MS) abstraction interface. This example is only compatible with JBoss AS5. The solution is know to work with Twinkle and linphone SIP soft-phones.
Shopping [http://www.mobicents.org/shopping-demo-jsr309.html]	Demonstrates integration with Seam and Java Enterprise Edition (JEE), and JSR 309 Media integration with text to speech (TTS) and dual-tone multi-frequency (DTMF) tones. The demonstration builds on the Converged Demo example, and adds support for the SIP Servlets v1.1 specification.
JSLEE/SIP Servlets Interoperability [http://www.mobicents.org/jslee-sips-interop-demo.html]	Demonstrates how the JBoss Communications platform components can work in concert with each other to provide a integrated solution. All major components of the platform are used in this example, which was created to demonstrate to JavaOne 2008 delegates a possible use case scenario for the platform.
Facebook Click to Call [http://www.mobicents.org/facebook-c2c.html]	Demonstrates how SIP Servlets and HTTP Servlets can be used can be used to create a Facebook plug-in that allows user to call POTS phones through a SIP-PSTN gateway provider. This demonstration is only available from the source repository; no binary is available.

Example	Description
Section 4.1.2, "The Diameter Event-Changing Service"	Demonstrates how the Diameter Event Charging, and the Location service, can be used to perform fixed-rated charging of calls (event charging). When a call is initiated, a debit of ten euros is applied to the `A Party` account. If the call is rejected by the `B Party`, or `A Party` hangs up before `B Party` can answer the call, the ten euro charge is credited to the `A Party` account.
Diameter Sh OpenIMS Integration [http://www.mobicents.org/mss-diameter_sh.html]	Demonstrates the integration between JBoss Communications and OpenIMS, using the Diameter Sh interface to receive profile updates and SIP.
Diameter Ro/Rf IIntegration [http://www.mobicents.org/mss-diameter_rorf.html]	A Diameter Ro/Rf service that performs online call charging.
Conference [http://www.mobicents.org/conference-demo-jsr309.html]	Demonstrates the capabilities of the Media Server, such as endpoint composition and conferencing, as well as proving that SIP Servlets are capable of working seamlessly with any third-party web framework, without repackaging or modifying the deployment descriptors. The demonstration uses Google's GWT Ajax framework with server-push updates to provide a desktop-like user interface experience and JSR 309 for Media Control.
Media IPBX [http://www.mobicents.org/mss-ipbx.html]	Demonstrates how a SIP PBX solution can be deployed using the JBoss Communications platform. For more information, refer to *Section 4.1.6, "Media IPBX"*.
JRuby on Rails SIP Servlets [http://www.mobicents.org/mss-jruby-example.html]	Demonstrates how JRuby on Rails can be used by the JBoss Communications platform to provide a multi-language application that can initiate phone calls to customers after they log a complaint through a web portal.
Pure JRuby on Rails Telco [http://www.mobicents.org/mss-pure-jruby-telco.html]	Builds on the JRuby on Rails SIP Servlets demonstration, but adds the ability to call the application rather that initially interact through the web portal. The application has the ability to set up and tear down the call.

Example	Description
Alerting Application [http://www.mobicents.org/alerting-app.html]	This application was developed so that the JBoss RHQ/Jopr Enterprise Management Solution would be able to notify system administrators when a monitoring alert is fired by Jopr/RHQ.
Alerting Application [http://www.mobicents.org/mss-presence-client-example.html]	A Call Blocking application interoperating with the Mobicents SIP Presence Service to fetch the blocked contacts through XCAP.

4.1. Operating the Example Applications

4.1.1. The Location Service

The JBoss Communications Location Service contains a list of mappings of request URIs to destination addresses. When the Location Service receives a request, it performs a lookup on that mapping and proxies the request simultaneously to the destination address (or addresses) associated with that URI.

The Location Service Mappings Cannot Currently Be Configured

The Location Service currently performs a lookup on a hard-coded list of addresses. This model is evolving toward the eventual use of a database.

Regardless of whether you are using the JBoss Application Server or the Tomcat Servlet Container as the Servlets Server, the application, container and Location Service perform the following steps:

- A user—let us call her Alice—makes a call to `sip:receiver@sip-servlets.com`. The `INVITE` is received by the servlet container, which then starts the Location Service.
- The Location Service, using non-SIP means, determines that the callee (i.e. the receiver) is registered at two locations, identified by the two SIP URIs, `sip:receiver@127.0.0.1:5090` and `sip:receiver@127.0.0.1:6090`.
- The Location Service proxies to those two destinations in parallel, without record-routing, and without making use of supervised mode.
- One of the destinations returns a `200 OK` status code; the second proxy is then canceled.
- The `200 OK` is forwarded to Alice, and call setup is completed as usual.

Here is the current list of hard-coded contacts and their location URIs:

sip:receiver@sip-servlets.com

- `sip:receiver@127.0.0.1:5090`
- `sip:receiver@127.0.0.1:6090`

4.1.1.1. The Location Service: Installing, Configuring and Running

Pre-Install Requirements and Prerequisites. The following requirements must be met before installation can begin.

Software Prerequisites

Either an JBCP SIP Servlets for JBoss or an JBCP SIP Servlets for Tomcat Installation

The Location Service requires either an JBCP SIP Servlets for JBoss or an JBCP SIP Servlets for Tomcat binary installation.

You can find detailed instructions on installing JBCP SIP Servlets for JBoss here: *Section 2.1, "SIP Servlet-Enabled JBoss Application Server: Installing, Configuring and Running"*.

You can find detailed instructions on installing JBCP SIP Servlets for Tomcat here: *Section 2.2, "SIP Servlet-Enabled Tomcat Servlet Container: Installing, Configuring and Running"*.

Downloading. The Location Service is comprised of two archive files, a Web Archive (WAR) and a Default Application Router (DAR) configuration file, which you need to add to your installed SIP Servlets Server. For more information about WAR files, refer to the *JBoss Application Server Administration and Development Guide* [http://www.jboss.org/file-access/default/members/jbossas/freezone/docs/Server_Configuration_Guide/beta422/html/index.html]. For more information about DAR files, refer to the *JSR 289 spec, Appendix C* [http://jcp.org/en/jsr/detail?id=289].

Download the Location Service's WAR file from here: *https://repository.jboss.org/nexus/content/groups/public-jboss/org/mobicents/servlet/sip/example/location-service/1.6.0.FINAL/location-service-1.6.0.FINAL.war*.

Download the Location Service's DAR file from here: *http://www.mobicents.org/locationservice-dar.properties*.

Installing. Both the `location-service-1.6.0.FINAL.war` WAR file and the `locationservice-dar.properties` DAR file that you downloaded should be placed into different directories in your SIP Servlet Server installation hierarchy. Which directory depends on whether you are using the Location Service with JBCP SIP Servlets for JBoss or with JBCP SIP Servlets for Tomcat:

JBCP SIP Servlets for JBoss

Place `location-service-1.6.0.FINAL.war` into the `JBOSS_HOME/server/default/deploy/` directory, and `locationservice-dar.properties` into the `JBOSS_HOME/server/default/conf/dars/` directory.

JBCP SIP Servlets for Tomcat

Place `location-service-1.6.0.FINAL.war` into the `CATALINA_HOME/webapps/` directory, and `locationservice-dar.properties` into the `CATALINA_HOME/conf/dars/` directory.

Configuring. The `darConfigurationFileLocation` attribute of the `Service` element must be set to the value **`conf/dars/locationservice-dar.properties`**. The instructions are given below by SIP Servlets Server type:

JBCP SIP Servlets for JBoss

Open the `JBOSS_HOME/server/default/deploy/jboss-web.deployer/server.xml` configuration file and find the `Service` element. Add an attribute to it called **`darConfigurationFileLocation`**, and set it to *`conf/dars/locationservice-dar.properties`*:

Example 4.1. Editing JBCP SIP Servlets for JBoss's server.xml for the Location Service

```
<Service
 name="jboss.web"
 className="org.mobicents.servlet.sip.startup.SipStandardService"
 sipApplicationDispatcherClassName="org.mobicents.servlet.sip.core.SipApplicationDispatcherImpl"
 darConfigurationFileLocation="conf/dars/locationservice-dar.properties"
 sipStackPropertiesFile="conf/mss-sip-stack.properties">
```

Make sure that the configuration file only contains one `darConfigurationFileLocation` attribute: your new one.

JBCP SIP Servlets for Tomcat

Open the `CATALINA_HOME/conf/server.xml` configuration file and find the `Service` element. Add an attribute to it called **`darConfigurationFileLocation`**, and set it to **`conf/dars/locationservice-dar.properties`**:

Example 4.2. Editing JBCP SIP Servlets for Tomcat's server.xml for the Location Service

```
<Service
 name="Sip-Servlets"
 className="org.mobicents.servlet.sip.startup.SipStandardService"
```

```
sipApplicationDispatcherClassName="org.mobicents.servlet.sip.core.SipApplicationDispatcherImpl"
darConfigurationFileLocation="conf/dars/locationservice-dar.properties"
sipStackPropertiesFile="conf/mss-sip-stack.properties">
```

Make sure that the configuration file only contains one `darConfigurationFileLocation` attribute: your new one.

Running. Once the WAR and DAR files have been placed in the right directories, and the JBoss Application Server or Tomcat Servlet Container knows where to find them (which you specified in a `server.xml` file), then you should go ahead and run the SIP Servlets Server.

To learn how to run the SIP Servlets-enabled JBoss Application Server, refer to *Section 2.1.7, "Running"*.

To learn how to run the SIP Servlets-enabled Tomcat Container, refer to *Section 2.2.7, "Running"*.

Testing. The following procedure shows how to test the Location Service.

Procedure 4.1.

1. Start two SIP soft-phones. The first phone should be set up as **`sip:receiver@sip-servlets.com`** at the IP address **`127.0.0.1`** on port **`5090`**. The second phone can be set up in any way you like. Note that the SIP phones do not have to be registered.
2. Using the second phone, make a call to `sip:receiver@sip-servlets.com`. If the Location Service has been set up correctly and is running, the first phone—as the receiver or callee—should now be ringing.

Stopping. To learn how to stop the SIP Servlets-enabled JBoss Application Server, refer to *Section 2.1.10, "Stopping"*.

To learn how to stop the SIP Servlets-enabled Tomcat Container, refer to *Section 2.2.8, "Stopping"*.

Uninstalling. Unless disk space is at a premium, there is usually no need to uninstall the Location Service. However, if you will not be using it again, you may want to unset or reset the `darConfigurationFileLocation` attribute of the `Service` element, which you set in the `server.xml` configuration file in *Configuring*.

You may also wish to delete the WAR and DAR files for the Location Service, which you installed in *Installing*.

4.1.2. The Diameter Event-Changing Service

The Diameter Event-Changing Service is based on the Location Service, which performs call-charging at a fixed rate. Upon the initiation of a call, a debit of €10.00 occurs. In the cases of

a call being rejected or the caller disconnecting (hanging up) before an answer is received, the caller's account is refunded.

Note that an JBCP SIP Servlets for JBoss installation is required to run this example; it will not work with JBCP SIP Servlets for Tomcat.

Provided here is a step-by-step description of the procedure as performed by the application and container:

Procedure 4.2. Diameter Event-Changing Service Step-By-Step

1. A user, Alice, makes a call to `sip:receiver@sip-servlets.com`. The `INVITE` is received by the servlet container, which sends a request to debit Alice's account to the Charging Server. The servlet container then invokes the location service.
2. The Location Service determines, without using the SIP protocol itself, where the callee—or receiver—is registered. The callee may be registered at two locations identified by two SIP URIs: `sip:receiver@127.0.0.1:5090` and `sip:receiver@127.0.0.1:6090`.
3. The Location Service proxies to those two destinations simultaneously, without record-routing and without using the supervised mode.
4. One of the destinations returns `200 (OK)`, and so the container cancels the other.
5. The `200 (OK)` is forwarded upstream to Alice, and the call setup is carried out as usual.
6. If none of the registered destinations accepts the call, a Diameter Accounting-Request for refund is sent to the Diameter Charging Server in order to debit the already-credited €10.00

4.1.2.1. Diameter Event-Changing Service: Installing, Configuring and Running

Preparing your JBCP SIP Servlets for JBoss server to run the Diameter Event-Changing example requires downloading a WAR archive, a DAR archive, the Ericsson Charging Emulator, setting an attribute in JBoss's `server.xml` configuration file, and then running JBoss AS. Detailed instructions follow.

Pre-Install Requirements and Prerequisites. The following requirements must be met before installation can begin.

Software Prerequisites

One JBCP SIP Servlets for JBoss Installation
: Before proceeding, you should follow the instructions for installing, configuring, running and testing JBCP SIP Servlets for JBoss from the binary distribution.

Downloading. The following procedure describes how to download the required files.

1. First, download the latest Web Application Archive (WAR) file corresponding to this example, the current version of which is named `diameter-event-charging-*.war`, from *https://repository.jboss.org/nexus/content/groups/public-jboss/org/mobicents/servlet/sip/example/diameter-event-charging/1.6.0.FINAL/diameter-event-charging-1.6.0.FINAL.war*.
2. Secondly, download the corresponding Disk Archive (DAR) configuration file here: *http://www.mobicents.org/diametereventcharging-dar.properties*.
3. Finally, you will need to download the Ericsson Charging Emulator, version 1.0, from *http://mobicents.googlecode.com/files/ChargingSDK-1_0_D31E.zip*.

Installing. The following procedure describes how to install the downloaded files.

1. Place the `diameter-event-charging-1.6.0.FINAL.war` WAR archive into the `$JBOSS_HOME/server/<profile>/deploy` directory.
2. Place the `diametereventcharging-dar.properties` DAR file in your `$JBOSS_HOME/server/<profile>/conf/dars` directory.
3. Finally, open the terminal, move into the directory to which you downloaded the Ericsson Charging SDK (for the sake of this example, we will call this directory `charging_sdk`), and then unzip the downloaded zip file (you can use Java's `jar -xvf` command for this:

```
~]$ cd charging_sdk
charging_sdk]$ jar -xvf ChargingSDK-1_0_D31E.zip
```

Alternatively, you can use Linux's `unzip` command to do the dirty work:

```
charging_sdk]$ unzip ChargingSDK-1_0_D31E.zip
```

Configuring. To configure the server for the Event-Changing example, simply open the `server.xml` configuration file in your server's `$JBOSS_HOME/server/<profile>/deploy/jboss-web.deployer/` directory, and edit the value of the `darConfigurationFileLocation` attribute of the `Service` element so that it is `conf/dars/mobicents-dar.properties`.

Example 4.3. Editing the darConfigurationFileLocation Attribute of the Service Tag

```
...
```

```
<Service name="jboss.web"
    className="org.mobicents.servlet.sip.startup.SipStandardService"
   sipApplicationDispatcherClassName="org.mobicents.servlet.sip.core.SipApplicationDispatcherImpl"
   sipApplicationRouterClassName="org.mobicents.servlet.sip.router.DefaultApplicationRouter"
   darConfigurationFileLocation="conf/dars/mobicents-dar.properties"
   sipStackPropertiesFile="conf/mss-sip-stack.properties">

...
```

Running. The following procedure describes how to run the Diameter Event-Changing Service.

Procedure 4.3. Diameter Event-Changing Service

1. First, you should run your JBCP SIP Servlets for JBoss server. For instructions on doing so, refer to *Section 2.1.7, "Running"*.

2. Then, run the Ericsson Charging Emulator. Open a terminal, change the working directory to the location of the unzipped Charging Emulator files (in `ChargingSDK-1_0_D31E` or a similarly-named directory), and run it with the `java -jar PPSDiamEmul.jar` command:

   ```
   ~]$ java -jar PPSDiamEmul.jar
   ```

Using. Using the Event-Changing service means, firstly, inserting some parameters into the Charging Emulator, and then, by using two SIP (soft)phones, calling one with the other. The following sequential instructions show you how.

SIP (Soft)Phone? Which?

The JBoss Communications team recommends one of the following SIP phones, and has found that they work well: the 3CX Phone, the SJ Phone or the WengoPhone.

Procedure 4.4. Using the Diameter Event-Changing Service

1. **Configure the Ericsson SDK Charging Emulator**

   Once you have started the Charging Emulator, you should configure it exactly as portrayed in *Figure 4.1, "Configuring the Charging Emulator"*.

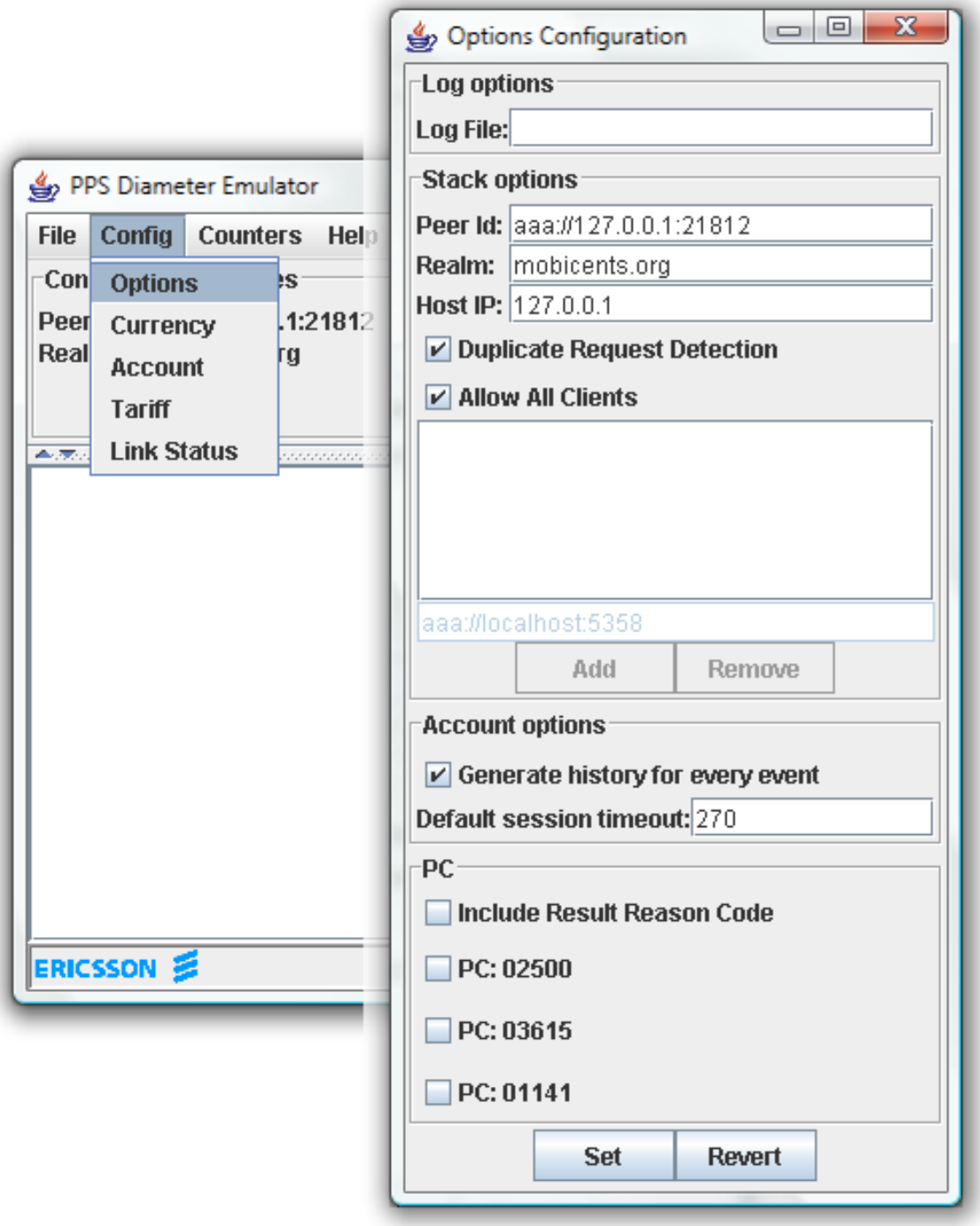


Figure 4.1. Configuring the Charging Emulator

1. Set the `Peer ID` to: **`aaa://127.0.0.1:21812`**
2. Set the `Realm` to: **`mobicents.org`**

3. Set the `Host IP` to: **`127.0.0.1`**

2. Start two SIP (soft)phones. You should set the first phone up with the following parameters: **`sip:receiver@sip-servlets`** on IP address **`127.0.0.1`** on port **`5090`**. The other phone can be set up any way you like.

3. Before making a call, open the **Config** → **Options** dialog window, as shown in the image.

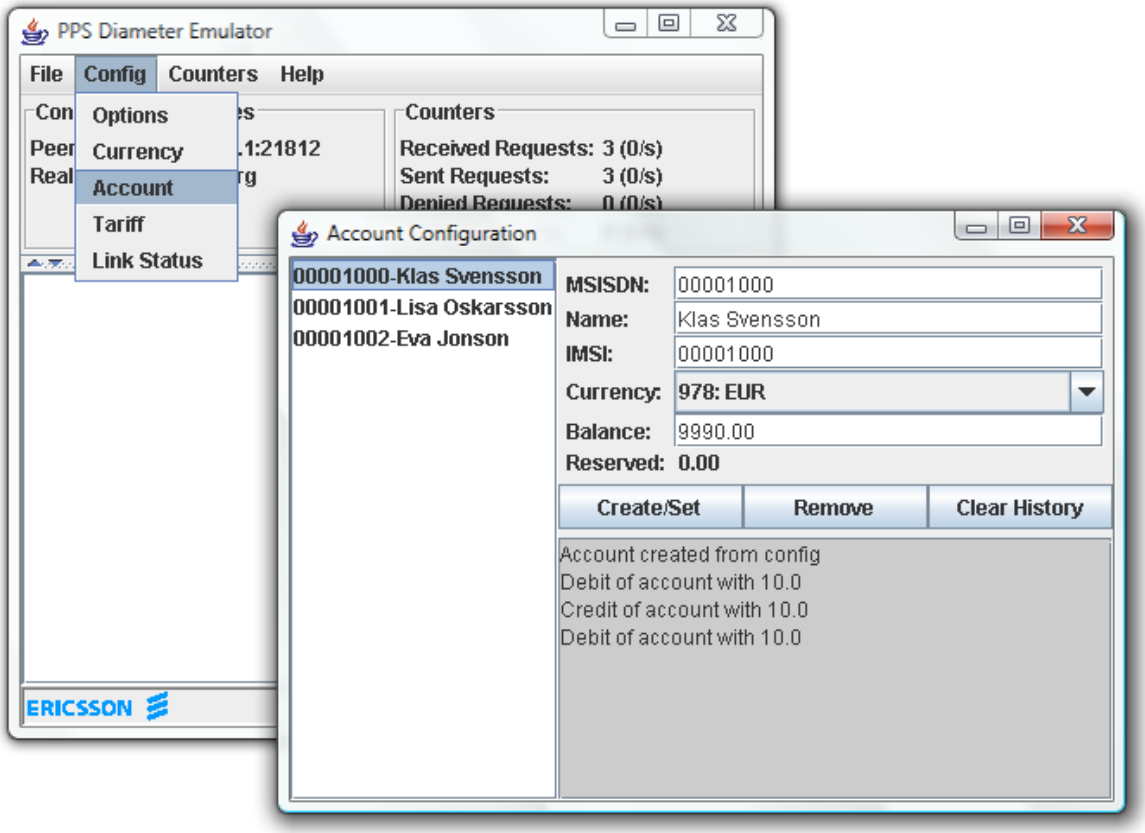


Figure 4.2. Configuring Accounts in the Charging Emulator

In the **Account Configuration** window of the Charging Emulator, you can see the user's balances. Select a user to watch the balance. You can also stretch the window lengthwise to view the user's transaction history.

4. Time to call! From the second, “any-configuration” phone, make a call to `sip:receiver@sip-servlets.com`. Upon doing so, the other phone should ring or signal that it is being contacted .

5. You should be able to see a request—immediately following the invite and before the other party (i.e. you) accepts or rejects the call—sent to the Charging Emulator. That is when the debit of the user's account is made. In the case that the call is rejected, or the caller gives up, a second, new Diameter request is sent to refund the initial amount charged by the call.

On the other hand, if the call is accepted, nothing else related to Diameter happens, and no second request takes place.

Please note that this is not the correct way to do charging, as Diameter provides other means, such as unit reservation. However, for the purpose of a demonstration it is sufficient to show the debit and follow-up credit working. Also, this is a fixed-price call, regardless of the duration. Charging can, of course, be configured so that it is time-based.

4.1.3. The Call-Blocking Service

The JBoss Communications Call-Blocking Service, upon receiving an `INVITE` request, checks to see whether the sender's address is a blocked contact. If so, it returns a `FORBIDDEN` reply; otherwise, call setup proceeds as normal.

Blocked Contacts Cannot Currently Be Configured

Blocked contacts are currently hard-coded addresses. This model is evolving towards the eventual use of a database.

Here is the current hard-coded list of blocked contacts:

- `sip:blocked-sender@sip-servlets.com`
- `sip:blocked-sender@127.0.0.1`

4.1.3.1. The Call-Blocking Service: Installing, Configuring and Running

Pre-Install Requirements and Prerequisites. The following requirements must be met before installation can begin.

Software Prerequisites

Either an JBCP SIP Servlets for JBoss or an JBCP SIP Servlets for Tomcat Installation

The Call-Blocking Service requires either an JBCP SIP Servlets for JBoss or an JBCP SIP Servlets for Tomcat binary installation.

You can find detailed instructions on installing JBCP SIP Servlets for JBoss here: *Section 2.1, “SIP Servlet-Enabled JBoss Application Server: Installing, Configuring and Running”*.

You can find detailed instructions on installing JBCP SIP Servlets for Tomcat here: *Section 2.2, “SIP Servlet-Enabled Tomcat Servlet Container: Installing, Configuring and Running”*.

Downloading. The Call-Blocking Service is comprised of two archive files, a Web Archive (WAR) and a Default Application Router (DAR) configuration file, which you need to add to your installed SIP Servlets Server. For more information about WAR files, refer to the *JBoss Application Server Administration and Development Guide* [http://www.jboss.org/file-access/default/members/jbossas/freezone/docs/Server_Configuration_Guide/beta422/html/index.html]. For more information about DAR files, refer to the *JSR 289 spec, Appendix C* [http://jcp.org/en/jsr/detail?id=289].

Download the Call-Blocking Service's WAR file from here: *https://repository.jboss.org/nexus/content/groups/public-jboss/org/mobicents/servlet/sip/example/call-blocking/1.6.0.FINAL/call-blocking-1.6.0.FINAL.war*.

Download the Call-Blocking Service's DAR file from here: *http://www.mobicents.org/call-blocking-servlet-dar.properties*.

Installing. Both the `call-blocking-1.6.0.FINAL.war` WAR file and the `call-blocking-servlet-dar.properties` DAR file that you downloaded should be placed into different directories in your SIP Servlet Server installation hierarchy. Which directory depends on whether you are using the Call-Blocking Service with JBCP SIP Servlets for JBoss or with JBCP SIP Servlets for Tomcat:

JBCP SIP Servlets for JBoss
: Place `call-blocking-1.6.0.FINAL.war` into the `JBOSS_HOME/server/default/deploy/` directory, and `call-blocking-servlet-dar.properties` into the `JBOSS_HOME/server/default/conf/dars/` directory.

JBCP SIP Servlets for Tomcat
: Place `call-blocking-servlet-dar.properties` into the `CATALINA_HOME/webapps/` directory, and `call-blocking-servlet-dar.properties` into the `CATALINA_HOME/conf/dars/` directory.

Configuring. The `darConfigurationFileLocation` attribute of the `Service` element must be set to the value **`conf/dars/call-blocking-servlet-dar.properties`**. The instructions for doing so are given below by SIP Servlets Server type:

JBCP SIP Servlets for JBoss
: Open the `JBOSS_HOME/server/default/deploy/jboss-web.deployer/server.xml` configuration file and find the `Service` element. Add an attribute to it called **`darConfigurationFileLocation`**, and set it to *`conf/dars/call-blocking-servlet-dar.properties`*:

Example 4.4. Editing JBCP SIP Servlets for JBoss's server.xml for the Call-Blocking Service

```
<Service
```

```
name="jboss.web"
className="org.mobicents.servlet.sip.startup.SipStandardService"
sipApplicationDispatcherClassName="org.mobicents.servlet.sip.core.SipApplicationDispatcherImpl"
darConfigurationFileLocation="conf/dars/call-blocking-servlet-dar.properties"
sipStackPropertiesFile="conf/mss-sip-stack.properties">
```

Make sure that the configuration file only contains one `darConfigurationFileLocation` attribute: your new one.

JBCP SIP Servlets for Tomcat

Open the `CATALINA_HOME/conf/server.xml` configuration file and find the `Service` element. Add an attribute to it called **`darConfigurationFileLocation`**, and set it to **`conf/dars/call-blocking-servlet-dar.properties`**:

Example 4.5. Editing JBCP SIP Servlets for Tomcat's server.xml for the Call-Blocking Service

```
<Service
name="Sip-Servlets"
className="org.mobicents.servlet.sip.startup.SipStandardService"
sipApplicationDispatcherClassName="org.mobicents.servlet.sip.core.SipApplicationDispatcherImpl"
darConfigurationFileLocation="conf/dars/call-blocking-servlet-dar.properties"
sipStackPropertiesFile="conf/mss-sip-stack.properties">
```

Make sure that the configuration file only contains one `darConfigurationFileLocation` attribute: your new one.

Running. Once the WAR and DAR files have been placed in the right directories, and the JBoss Application Server or Tomcat Servlet Container knows where to find them (which you specified in a `server.xml` file), then you should go ahead and run the SIP Servlets Server.

To learn how to run the SIP Servlets-enabled JBoss Application Server, refer to *Section 2.1.7, "Running"*.

To learn how to run the SIP Servlets-enabled Tomcat Container, refer to *Section 2.2.7, "Running"*.

Testing. The following procedure shows how to test the Call-Blocking Service.

Procedure 4.5. Testing the Call Blocking Service

1. Start a SIP softphone of your choice. The account name should be **`blocked-sender`**. The `From Header` should list one of the following addresses: **`sip:blocked-sender@sip-`**

servlets.com or **sip:blocked-sender@127.0.0.1**. The SIP softphone does not need to be registered.

2. Make a call to any address, and you should receive a FORBIDDEN response.

Stopping. To learn how to stop the SIP Servlets-enabled JBoss Application Server, refer to *Section 2.1.10, "Stopping"*.

To learn how to stop the SIP Servlets-enabled Tomcat Container, refer to *Section 2.2.8, "Stopping"*.

Uninstalling. Unless disk space is at a premium, there is usually no need to uninstall the Call-Blocking Service. However, if you will not be using it again, you may want to unset or reset the darConfigurationFileLocation attribute of the Service element, which you set in the server.xml configuration file in *Configuring*.

You may also wish to delete the WAR and DAR files for the Call-Blocking Service, which you installed in *Installing*.

4.1.4. The Call-Forwarding Service

The JBoss Communications Call-Forwarding Service, upon receiving an INVITE request, checks to see whether the sender's address is among those in a list of addresses which need to be forwarded. If so, then the Call-Forwarding Service acts as a Back-to-Back User Agent (B2BUA), and creates a new call leg to the destination. When the response is received from the new call leg, it sends it an acknowledgment (ACK) and then responds to the original caller. If, on the other hand, the server does not receive an ACK, then it tears down the new call leg with a BYE. Once the BYE is received, then it answers OK directly and sends the BYE to the new call leg.

Contacts to Forward Cannot Currently Be Configured

Contacts to forward are currently hard-coded addresses. This model is evolving toward the eventual use of a database.

Here is the current hard-coded list of contacts to forward:

- sip:receiver@sip-servlets.com
- sip:receiver@127.0.0.1

4.1.4.1. The Call-Forwarding Service: Installing, Configuring and Running

Pre-Install Requirements and Prerequisites. The following requirements must be met before installation can begin.

Software Prerequisites

Either an JBCP SIP Servlets for JBoss or an JBCP SIP Servlets for Tomcat Installation

The Call-Forwarding Service requires either an JBCP SIP Servlets for JBoss or an JBCP SIP Servlets for Tomcat binary installation.

You can find detailed instructions on installing JBCP SIP Servlets for JBoss here: *Section 2.1, "SIP Servlet-Enabled JBoss Application Server: Installing, Configuring and Running"*.

You can find detailed instructions on installing JBCP SIP Servlets for Tomcat here: *Section 2.2, "SIP Servlet-Enabled Tomcat Servlet Container: Installing, Configuring and Running"*.

Downloading. The Call-Forwarding Service is comprised of two archive files, a Web Archive (WAR) and a Data Archive (DAR), which you need to add to your installed SIP Servlets Server. For more information about WAR and DAR files, refer to the *JBoss Application Server Administration and Development Guide* [http://www.jboss.org/file-access/default/members/jbossas/freezone/docs/Server_Configuration_Guide/beta422/html/index.html].

Download the Call-Forwarding Service's WAR file from here: *https://repository.jboss.org/nexus/content/groups/public-jboss/org/mobicents/servlet/sip/example/call-forwarding/1.6.0.FINAL/call-forwarding-1.6.0.FINAL.war*.

Download the Call-Forwarding Service's DAR file from here: *http://www.mobicents.org/call-forwarding-servlet-dar.properties*.

Installing. Both the `call-forwarding-1.6.0.FINAL.war` WAR file and the `call-forwarding-servlet-dar.properties` DAR file that you downloaded should be placed into different directories in your SIP Servlet Server installation hierarchy. Which directory depends on whether you are using the Call-Forwarding Service with JBCP SIP Servlets for JBoss or with JBCP SIP Servlets for Tomcat:

JBCP SIP Servlets for JBoss

Place `call-forwarding-1.6.0.FINAL.war` into the `JBOSS_HOME/server/default/deploy/` directory, and `call-forwarding-servlet-dar.properties` into the `JBOSS_HOME/server/default/conf/dars/` directory.

JBCP SIP Servlets for Tomcat

Place `call-forwarding-1.6.0.FINAL.war` into the `CATALINA_HOME/webapps/` directory, and `call-forwarding-servlet-dar.properties` into the `CATALINA_HOME/conf/dars/` directory.

Configuring. The `darConfigurationFileLocation` attribute of the `Service` element must be set to the value **`conf/dars/call-forwarding-b2bua-servlet-dar.properties`**. The instructions for doing so are given below by SIP Servlets Server type:

JBCP SIP Servlets for JBoss

Open the `JBOSS_HOME/server/default/deploy/jboss-web.deployer/server.xml` configuration file and find the `Service` element. Add an attribute to it called **`darConfigurationFileLocation`**, and set it to *`conf/dars/call-forwarding-b2bua-servlet-dar.properties`*:

Example 4.6. Editing JBCP SIP Servlets for JBoss's server.xml for the Call-Forwarding Service

```
<Service
 name="jboss.web"
 className="org.mobicents.servlet.sip.startup.SipStandardService"
 sipApplicationDispatcherClassName="org.mobicents.servlet.sip.core.SipApplicationDispatcherImpl"
 darConfigurationFileLocation="conf/dars/call-forwarding-b2bua-servlet-dar.properties"
 sipStackPropertiesFile="conf/mss-sip-stack.properties">
```

Make sure that the configuration file only contains one `darConfigurationFileLocation` attribute: your new one.

JBCP SIP Servlets for Tomcat

Open the `CATALINA_HOME/conf/server.xml` configuration file and find the `Service` element. Add an attribute to it called **`darConfigurationFileLocation`**, and set it to **`conf/dars/call-forwarding-b2bua-servlet-dar.properties`**:

Example 4.7. Editing JBCP SIP Servlets for Tomcat's server.xml for the Call-Forwarding Service

```
<Service
 name="Sip-Servlets"
 className="org.mobicents.servlet.sip.startup.SipStandardService"
 sipApplicationDispatcherClassName="org.mobicents.servlet.sip.core.SipApplicationDispatcherImpl"
 darConfigurationFileLocation="conf/dars/call-forwarding-b2bua-servlet-dar.properties"
 sipStackPropertiesFile="conf/mss-sip-stack.properties">
```

Make sure that the configuration file only contains one `darConfigurationFileLocation` attribute: your new one.

Running. Once the WAR and DAR files have been placed in the right directories, and the JBoss Application Server or Tomcat Servlet Container knows where to find them (which you specified in a `server.xml` file), then you should go ahead and run the SIP Servlets Server.

To learn how to run the SIP Servlets-enabled JBoss Application Server, refer to *Section 2.1.7, "Running"*.

To learn how to run the SIP Servlets-enabled Tomcat Container, refer to *bbssswticar-binary-SIP_Servlets_Server_with_Tomcat-Running*.

Testing. The following procedure shows how to test the Call-Forwarding Service.

Procedure 4.6.

1. Start two SIP soft-phones of your choice. Set the account settings of the first SIP softphone to:

   - Account name: **`forward-receiver`**
   - IP address: **`127.0.0.1`**
   - Port: **`5090`**

   Neither of the SIP soft-phones needs to be registered.

2. From the second phone, make a call to `sip:receiver@sip-servlets.com`. The first phone, "forward-receiver", should now be ringing.

Stopping. To learn how to stop the SIP Servlets-enabled JBoss Application Server, refer to *Section 2.1.10, "Stopping"*.

To learn how to stop the SIP Servlets-enabled Tomcat Container, refer to *Section 2.2.8, "Stopping"*.

Uninstalling. Unless disk space is at a premium, there is usually no need to uninstall the Call-Forwarding Service. However, if you will not be using it again, you may want to unset or reset the `darConfigurationFileLocation` attribute of the `Service` element, which you set in the `server.xml` configuration file in *Configuring*.

You may also wish to delete the WAR and DAR files for the Call-Forwarding Service, which you installed in *Installing*.

4.1.5. The Call-Controller Service

The Call-Controller service is a composition of two other services: Call-Blocking and Call-Forwarding. Essentially, it performs the services of both call-forwarding and call-blocking.

- To learn about how the Call-Blocking service works, refer to *Section 4.1.3, "The Call-Blocking Service"*.
- To learn about how the Call-Forwarding service works, refer to *Section 4.1.4, "The Call-Forwarding Service"*.

Blocked Contacts and Contacts to Forward Cannot Currently Be Configured

Both the list of blocked contacts and the list of contacts to forward are currently both hard-coded. However, both of those models are evolving toward the eventual use of databases.

4.1.5.1. The Call-Controller Service: Installing, Configuring and Running

The Call-Controller service requires the two WAR files for the Call-Blocking and Call-Forwarding services to be placed in the correct directory inside your JBoss Communications SIP Servlets Server binary installation. However, the Call-Controller service does *not* require their corresponding DAR files: you need only to download and install a DAR file customized for the Call-Controller service. The instructions below show you how to do precisely this; there is no need, therefore, to first install either the Call-Blocking or the Call-Forwarding services, though it is helpful to at least be familiar with them.

Pre-Install Requirements and Prerequisites. The following requirements must be met before installation can begin.

Software Prerequisites

Either an JBCP SIP Servlets for JBoss or an JBCP SIP Servlets for Tomcat Installation

The Call-Controller Service requires either an JBCP SIP Servlets for JBoss or an JBCP SIP Servlets for Tomcat binary installation.

You can find detailed instructions on installing JBCP SIP Servlets for JBoss here: *Section 2.1, “SIP Servlet-Enabled JBoss Application Server: Installing, Configuring and Running”*.

You can find detailed instructions on installing JBCP SIP Servlets for Tomcat here: *Section 2.2, “SIP Servlet-Enabled Tomcat Servlet Container: Installing, Configuring and Running”*.

Downloading. The Call-Controller Service is comprised of two WAR files, one for the Call-Forwarding service and one for Call-Blocking, and a customized Call-Controller DAR file. You do not need to install the DAR files for the Call-Forwarding or the Call-Blocking services. For more information about WAR files, refer to the *JBoss Application Server Administration and Development Guide* [http://www.jboss.org/file-access/default/members/jbossas/freezone/docs/Server_Configuration_Guide/beta422/html/index.html]. For more information about DAR files, refer to the *JSR 289 spec, Appendix C* [http://jcp.org/en/jsr/detail?id=289]

Download the Call-Blocking Service's WAR file from here: *https://repository.jboss.org/nexus/content/groups/public-jboss/org/mobicents/servlet/sip/example/call-blocking/1.6.0.FINAL/call-blocking-1.6.0.FINAL.war*.

Download the Call-Forwarding Service's WAR file from here: *https://repository.jboss.org/nexus/content/groups/public-jboss/org/mobicents/servlet/sip/example/call-forwarding/1.6.0.FINAL/call-forwarding-1.6.0.FINAL.war*.

Download the Call-Controller Service's DAR file from here: *http://www.mobicents.org/call-controller-servlet-dar.properties*.

Installing. The `call-blocking-1.6.0.FINAL.war`, `call-forwarding-1.6.0.FINAL.war` and `call-controller-servlet-dar.properties` archive files that you downloaded should be placed into different directories in your SIP Servlet Server installation hierarchy. Which directory depends on whether you are using the Call-Controller Service with JBCP SIP Servlets for JBoss or with JBCP SIP Servlets for Tomcat:

JBCP SIP Servlets for JBoss

Place `call-blocking-1.6.0.FINAL.war` and `call-forwarding-1.6.0.FINAL.war` into the `JBOSS_HOME/server/default/deploy/` directory, and `call-controller-servlet-dar.properties` into the `JBOSS_HOME/server/default/conf/dars/` directory.

JBCP SIP Servlets for Tomcat

Place `call-blocking-1.6.0.FINAL.war` and `call-forwarding-1.6.0.FINAL.war` into the `CATALINA_HOME/webapps/` directory, and `call-controller-servlet-dar.properties` into the `CATALINA_HOME/conf/dars/` directory.

Configuring. The `darConfigurationFileLocation` attribute of the `Service` element must be set to the value **`conf/dars/call-controller-servlet-dar.properties`**. Instructions for doing so are given below by SIP Servlets Server type:

JBCP SIP Servlets for JBoss

Open the `JBOSS_HOME/server/default/deploy/jboss-web.deployer/server.xml` configuration file and find the `Service` element. Add an attribute to it called **`darConfigurationFileLocation`**, and set it to *`conf/dars/call-controller-servlet-dar.properties`* :

Example 4.8. Editing JBCP SIP Servlets for JBoss's server.xml for the Call-Controller Service

```
<Service
 name="jboss.web"
 className="org.mobicents.servlet.sip.startup.SipStandardService"
 sipApplicationDispatcherClassName="org.mobicents.servlet.sip.core.SipApplicationDispatcherImpl"
 darConfigurationFileLocation="conf/dars/call-controller-servlet-dar.properties "
 sipStackPropertiesFile="conf/mss-sip-stack.properties">
```

Make sure that the configuration file only contains one `darConfigurationFileLocation` attribute: your new one.

JBCP SIP Servlets for Tomcat

Open the `CATALINA_HOME/conf/server.xml` configuration file and find the `Service` element. Add an attribute to it called **`darConfigurationFileLocation`**, and set it to **`conf/dars/call-controller-servlet-dar.properties`** :

Example 4.9. Editing JBCP SIP Servlets for Tomcat's server.xml for the Call-Controller Service

```
<Service
 name="Sip-Servlets"
 className="org.mobicents.servlet.sip.startup.SipStandardService"
 sipApplicationDispatcherClassName="org.mobicents.servlet.sip.core.SipApplicationDispatcherImpl"
 darConfigurationFileLocation="conf/dars/call-controller-servlet-dar.properties "
 sipStackPropertiesFile="conf/mss-sip-stack.properties">
```

Make sure that the configuration file only contains one `darConfigurationFileLocation` attribute: your new one.

Running. Once the WAR and DAR files have been placed in the right directories, and the JBoss Application Server or Tomcat Servlet Container knows where to find them (which you specified in a `server.xml` file), then you should go ahead and run the SIP Servlets Server.

To learn how to run the SIP Servlets-enabled JBoss Application Server, refer to *Section 2.1.7, "Running"*.

To learn how to run the SIP Servlets-enabled Tomcat Container, refer to *Section 2.2.7, "Running"*.

Testing. Two use-cases can be distinguished for the Call-Controller service: one in which a call is blocked, and another in which a call is forwarded. Therefore, we have two cases for which we can test the Call-Controller.

Procedure 4.7. Blocking a Call with Call-Controller

1. Start two SIP soft-phones of your choice. Set the account settings of the SIP soft-phones to:

Relevant First Softphone Settings

- Account name: **`forward-receiver`**
- IP address: **`127.0.0.1`**

- Port: **`5090`**

Relevant Second Softphone Settings

- Account name: **`blocked-sender`**

Neither of the SIP soft-phones needs to be registered.

2. From the second phone, `blocked-sender`, make a call to `sip:receiver@sip-servlets.com`. You should receive a `FORBIDDEN` response.

Procedure 4.8. Forwarding a Call with Call-Controller

1. Start two SIP soft-phones of your choice. Set the account settings of the SIP soft-phones to:

Relevant First Softphone Settings

- Account name: **`forward-receiver`**
- IP address: **`127.0.0.1`**
- Port: **`5090`**

Relevant Second Softphone Settings

- Account name: **`forward-sender`**

Neither of the SIP soft-phones needs to be registered.

2. From the second softphone, `forward-sender`, make a call to `sip:receiver@sip-servlets.com`. The first phone, `forward-receiver`, should now be ringing.

Stopping. To learn how to stop the SIP Servlets-enabled JBoss Application Server, refer to *Section 2.1.10, "Stopping"*.

To learn how to stop the SIP Servlets-enabled Tomcat Container, refer to *Section 2.2.8, "Stopping"*.

Uninstalling. Unless disk space is at a premium, there is usually no need to uninstall the Call-Controller Service. However, if you will not be using it again, you may want to unset or reset the `darConfigurationFileLocation` attribute of the `Service` element, which you set in the `server.xml` configuration file in *Configuring*.

You may also wish to delete the WAR and DAR files for the Call-Controller Service, which you installed in *Installing*.

Note

Chapter 4, SIP Servlet Example Applications provides more information about other service examples available.

4.1.6. Media IPBX

The Media IPBX provides an extensible and customizable SIP PBX solution, based on the Seam Telco Framework (STF). While the PBX is currently provided as a capability demonstration, the ultimate goal is to transition Media IPBX into a fully-fledged SIP PBX solution.

Media IPBX terminates all calls to JBoss Communications Media Server conference endpoints, which provides flexibility in manipulating established calls including server-side conferencing and ring-back tones. The PBX can also be implemented as a Session Border Controller.

Note

Media IPBX is compatible with JBoss Communications SIP Servlets with JBoss AS 4.2.3. Versions prior to this release do not support Media IPBX.

Media IPBX provides the following major features:

- User authentication.
- SIP phone registration.
- System configuration.
- Individual user views.
- Call monitoring and management.
- Multiple SIP phone instances per user.
- Status-based SIP phone assignment for incoming calls.
- Public Switched Telephone Network (PSTN) support, including administrative functions.
- Support for SIP REGISTER requests to automatically add phones by matching the username, or username and hostname (in 'strict mode' only).
- Optionally specify local or online sources for announcements and ringback tones.
- Session Border Controller capability.
- Full conferencing support, including:

- Privacy functions, including mute and closed-calls.
- Call status announcement.
- Ringback tones when waiting for other participants to join the conference.
- Joining incoming calls to a conference.
- Parking calls and isolating a single speaker using dual-tone multi-frequency (DTMF) tones. This feature is currently experimental.

Many of the features in Media IPBX are presented to the user as hints on the GUI portal pages. It is recommended that you install Media IPBX and experiment with the demonstration to gain an understanding of how the solution works.

For information about installing and running Media IPBX, including binary and source code locations, visit the *Media IPBX homepage* [http://www.mobicents.org/mss-ipbx.html].

Clustering and High Availability

5.1. JBoss Communications SIP Servlets for JBoss: Clustering Support

JBoss Communications supports the clustering of SIP Servlets-enabled JBoss Application Servers for performance, reliability and failover purposes. Note that only JBCP SIP Servlets for JBoss Servers can be used as cluster nodes; JBCP SIP Servlets for Tomcat Containers are not supported.

The SIP Servlets Server uses the JBoss Application Server as its servlet container, and takes advantage of its capabilities, including clustering and failover. For detailed background information about JBoss Application Server clustering refer to the *JBoss Application Server Clustering Guide* [http://www.jboss.org/file-access/default/members/jbossas/freezone/docs/Clustering_Guide/beta422/html/index.html].

5.1.1. SIP Servlets Server Cluster: Installing, Configuring and Running

5.1.1.1. Pre-Install Requirements and Prerequisites

Software Prerequisites

A SIP Servlets-enabled JBoss Application Server

Before proceeding, ensure you have correctly configured your JBoss Application Server, according to SIP Servlet Server requirements:

- *Section 2.1, "SIP Servlet-Enabled JBoss Application Server: Installing, Configuring and Running"*

The easiest way to set up a cluster of SIP Servlets-enabled JBoss Application Servers is to install, configure and test the binary distribution on one machine, and then copy the entire installation (directory) to the other machines in the cluster. This is the approach taken in this chapter.

Install a SIP Servlets Server with JBoss by following the instructions detailed in *Section 2.1, "SIP Servlet-Enabled JBoss Application Server: Installing, Configuring and Running"*.

Afer meeting the requirement you can begin to configure the server *Section 5.1.1.2, "Configuring"* below.

5.1.1.2. Configuring

Once installed, the JBCP SIP Servlets for JBoss binary distribution requires only minor configuration in order to enable clustering.

SIP, and HTTP session state clustering is pre-configured straight out of the binary distribution. However, to enable session replication in a node, you must tag it as `<distributable/>` both in the `web.xml` and `sip.xml` descriptors. This can be done only individually (per application).

5.1.1.3. Running

Clustering with JBCP SIP Servlets for JBoss nodes requires running all of the nodes using the "all" Server Configuration Profile, which is specified when you invoke `run.sh` or `run.bat`.

Running JBCP SIP Servlets for JBoss with the "all" Configuration Profile, on Linux. To run the server on Linux using the "all" Configuration Profile, start the server with the following command:

```
JBCP SIP Servlets-jboss-<version>]$ ./bin/run.sh -c all
```

Running JBCP SIP Servlets for JBoss with the "all" Configuration Profile, on Windows. To run the server on Windows using the "all" Configuration Profile, open the Command Prompt, change your folder to the topmost folder of your JBCP SIP Servlets for JBoss installation, and issue the following command:

```
C:Usersuser\<username>My DownloadsJBCP SIP Servlets-jboss-
<version>>binrun.bat -c all
```

Distributing requests between nodes. Together with the application server nodes, it is advised to run a SIP load-balancer or an IP load-balancer. The IP load balancer will distribute the traffic evenly between the nodes. A load-balancer is a single entry-point to all nodes. All calls should be made through the load balancer if High Availability is required. For more information about load balancing, refer to *???*.

By default, the servers are configured with one SIP load-balancer set to the IP address `127.0.0.1`. This is specified in the `balancers` attribute in the `server.xml` configuration file as follows:

```
<Service name="jboss.web"
    className="org.mobicents.servlet.sip.startup.failover.SipStandardBalancerNodeService"
  balancers="127.0.0.1"
  sipPathName="org.mobicents.ha"
  sipApplicationDispatcherClassName="org.mobicents.servlet.sip.core.SipApplicationDispatcherImpl"
concurrencyControlMode="None"
  darConfigurationFileLocation="conf/dars/mobicents-dar.properties"
  sipStackPropertiesFile="conf/mss-sip-stack.properties">
```

Multiple load balancers can be specified and all of them will be updated on the health status of the node. The complete syntax for the balancers string is the following:

```
<Service name="jboss.web"
```

```
    ...
 balancers="ipAddress1:sipPort1:rmiPort1;ipAddress2:sipPort2:rmiPort2;..3...4.."
 ...>
```

If the RMI port is omitted, port 2000 is assumed, and if the SIP port is omitted, 5065 is assumed.

Warning

The SIP port specified in the balancers string for each balancer refers to the internal SIP port of the SIP balancer. That is because the internal port faces the cluster nodes directly. Requests coming through the internal port will go to the external port and vice versa. If you put the external port in the `balancers` string, then the SIP LB will assume that the requests comes from outside the cluster and it will route it back to some cluster node instead of routing it outside the cluster as expected. Always use the SIP internal port in the `balancers` string. The exception to this rule is when a single port is used for internal and external ports in the SIP load balancer. In that case, the direction analysis is done based on `Via` headers and the requests are routed correctly without extra settings.

When multiple SIP load balancers are specified, the outgoing requests will always go through the first one, or an IP load balancer can be used and the requests will be distributed based on the IP balancer policy. To route the outgoing requests to a particular IP address (the IP load balancer address for example) the `outboundProxy` property can be used:

```
<Service name="jboss.web"
     ...
  balancers="127.0.0.1:5060:2000;127.0.0.1:5160:2100"
outboundProxy="127.0.0.1:5500"
  ...>
```

In this example configuration, all outbound requests will go through 127.0.0.1:5500, while the node will perform the health checks against two SIP load balancers. If the 127.0.0.1:5500 machine is an IP load balancer, it should be configured to spray the SIP load balancers, and they will route the requests outside the cluster reliably.

The `outboundProxy` attribute overrides the default effect of specifying a SIP port for SIP load balancers in the `balancers` string.

5.1.1.4. Stopping

Individual nodes in the cluster should be stopped one-by-one, followed by the SIP load balancer. Refer to:

- Stopping the SIP load balancer: *Section 5.3.1.7, “Stopping”*
- Stopping JBCP SIP Servlets for JBoss: *Section 2.1.10, “Stopping”*
- Stopping JBCP SIP Servlets for Tomcat: *Section 2.2.8, “Stopping”*

5.1.1.5. Testing

To test your JBoss cluster setup for mid-call failover (Established SIP Dialog), refer to:

- *Section 5.2, “JBoss Communications SIP Servlets for JBoss: Transparent Failover”*

5.1.1.6. Uninstalling

Uninstalling a SIP Servlets Cluster would entail deleting the Mobicents SIP Servlets Servers directories, the SIP Load Balancer directory, and possibly removing the unused `JBOSS_HOME` environment variable.

5.1.2. SIP Sessions Passivation/Activation

5.1.2.1. Description of SIP Sessions passivation

Passivation is the process of controlling memory usage by removing relatively unused sessions from memory while storing them in persistent storage. If a passivated session is requested by a client, it can be "activated" back into memory and removed from the persistent store. Mobicents Sip Servlets supports passivation of SipSessions (and by extension SIP Dialogs) and SipApplicationSessions sipapps whose sip.xml includes the `distributable` tag (i.e. clustered sipapps).

Passivation occurs at 2 points during the lifecycle of a sip application:

- When the container requests the creation of a new SIP Session or SIP Application Session. If the number of currently active session exceeds a configurable limit, an attempt is made to passivate sessions to make room in memory.
- Periodically (by default, every ten seconds) as the JBoss Web background thread runs.

A session will be passivated if one of the following holds true:

- The session hasn't been used in greater than a configurable maximum idle time.
- The number of active sessions exceeds a configurable maximum and the session hasn't been used in greater than a configurable minimum idle time.

In both cases, sessions are passivated on a Least Recently Used (LRU) basis.

5.1.2.2. Configuring Distributable SIP Session Passivation in your SIP Application

5.1.2.2.1. Step 1: Add the following to jboss-web.xml for your sip application

```
<max-active-sessions>20</max-active-sessions>
<passivation-config>
  <use-session-passivation>true</use-session-passivation>
  <passivation-min-idle-time>60</passivation-min-idle-time>
  <passivation-max-idle-time>600</passivation-max-idle-time>
</passivation-config>
```

- `max-active-sessions` Determines the maximum number of active sessions allowed. If the number of sessions managed by the session manager exceeds this value and passivation is enabled, the excess will be passivated based on the configured passivation-min-idle-time. If after passivation is completed (or if passivation is disabled), the number of active sessions still exceeds this limit, attempts to create new sessions will be rejected. If set to -1 (the default), there is no limit.
- `passivation-min-idle-time` Determines the minimum time (in seconds) that a session must have been inactive before the container will consider passivating it in order to reduce the active session count below max-active-sessions. A value of -1 (the default) disables passivating sessions before passivation-max-idle-time. Neither a value of -1 nor a high value are recommended if max-active-sessions is set.
- `passivation-max-idle-time` Determines the maximum time (in seconds) that a session can be inactive before the container should attempt to passivate it to save memory. Passivation of such sessions will take place regardless of whether the active session count exceeds max-active-sessions. Should be less than the sip.xml session-timeout setting. A value of -1 (the default) disables passivation based on maximum inactivity.
- `use-session-passivation` Determines whether session passivation will be enabled for the web application. Default is false.

5.1.2.2.2. Step 2 : Configure the JBoss Cache cache loader

In most cases you don't need to do anything for Step 2; the standard JBoss AS configuration for distributable session caching should suit your needs. The following is a bit more detail in case you're interested or want to change from the defaults.

SIP Session passivation relies on JBoss Cache's Cache Loader passivation for storing and retrieving the passivated sessions. Therefore the cache instance used by your sipapps clustered session manager must be configured to enable Cache Loader passivaton. The Cache Loader configuration for the `standard-session-cache` config is as follows:

```
<property name="cacheLoaderConfig">
 <bean class="org.jboss.cache.config.CacheLoaderConfig">
      <!-- Do not change these -->
      <property name="passivation">true </property>
      <property name="shared">false </property>

      <property name="individualCacheLoaderConfigs">
       <list>
         <bean class="org.jboss.cache.loader.FileCacheLoaderConfig">
           <!-- Where passivated sessions are stored -->
        <property name="location">${jboss.server.data.dir}${/}session </property>
           <!-- Do not change these -->
           <property name="async">false </property>
           <property name="fetchPersistentState">true </property>
           <property name="purgeOnStartup">true </property>
           <property name="ignoreModifications">false </property>
           <property name="checkCharacterPortability">false </property>
         </bean>
       </list>
      </property>
 </bean>
</property>
```

- `passivation` property MUST be true.
- `shared` property MUST be false. Do not passivate sessions to a shared persistent store, otherwise if another node activates the session, it will be gone from the persistent store and also gone from memory on other nodes that have passivated it. Backup copies will be lost.
- `individualCacheLoaderConfigs` property accepts a list of Cache Loader configurations. JBC allows you to chain cache loaders; see the JBC docs. For the session passivation use case a single cache loader is sufficient.
- `class` attribute on a cache loader config bean MUST refer to the configuration class for a cache loader implementation (e.g. org.jboss.cache.loader.FileCacheLoaderConfig, org.jboss.cache.loader.JDBCCacheLoaderConfig). See the JBoss Cache documentation for more on the available CacheLoader implementations. If you wish to use JDBCCacheLoader (to

persist to a database rather than the filesystem used by FileCacheLoader) note the comment above about the shared property. Don't use a shared database, or at least not a shared table in the database. Each node in the cluster must have its own storage location.

- `location` property for FileCacheLoaderConfig defines the root node of the filesystem tree where the passivated session should be stored. The default is to store them in your JBoss AS configuration's data directory.
- `async` MUST be false to ensure passivated sessions are promptly written to the persistent store.
- `fetchPersistentState` MUST be true to ensure passivated sessions are included in the set of session backup copies transferred over from other nodes when the cache starts.
- `purgeOnStartup` should be true to ensure out-of-date session data left over from a previous shutdown of a server doesn't pollute the current data set.
- `ignoreModifications` should be false.
- `checkCharacterPortability` should be false as a minor performance optimization.

Refer to the JBossCache documentation for further details about Cache Loader configuration.

5.1.2.3. Configuring Server-Wide Passivation Defaults

The section on jboss-web.xml configuration above described how to configure passivation on a sipapp-by-sipapp basis. If you are deploying several sipapps and each uses the same set of passivation configurations, you may wish to skip the jboss-web.xml configuration and just change the server-wide defaults. This can be done by uncommenting the following section in the $JBOSS_HOME/server/xxx/deployers/jbossweb.deployer/META-INF/war-deployers-jboss-beans.xml file's WebAppClusteringDefaultsDeployer and setting the values you desire:

```
<!--
 <property name="useSessionPassivation">false </property>
 <property name="passivationMaxIdleTime">-1 </property>
 <property name="passivationMinIdleTime">-1 </property>
-->
```

5.2. JBoss Communications SIP Servlets for JBoss: Transparent Failover

A JBoss Communications SIP Servlets Server for JBoss cluster does not employ any standby nodes. Typically, proxies and registrars must share the user location table by using a database cluster.

The JBoss Communications SIP Load Balancer, which is a SIP Call ID-aware load balancer, is used as the intermediary. The SIP Load Balancer forwards stateful transaction requests to cluster nodes based on its provisioning algorithm. The SIP Load Balancer acts as an entry-point to the cluster, and distributes the incoming requests between nodes. It is always advised to use a SIP load balancer or an IP load balancer in a cluster configuration.

This choice of implementation has many benefits:

- There is no need for standby nodes, because the remaining nodes in a degraded cluster automatically and transparently (to the user) take on the load of the failed node. This can be done because both the SIP Load Balancer and SIP Servlet-enabled JBoss Application Servers support mid-call or call setup failover (respectively Established SIP Dialog and Early SIP Dialog).
- There is no need to ensure that requests are directed to the correct node, because in a SIP Servlets-enabled JBoss Application Server (or JBoss Communications JAIN SLEE server) cluster, any node can serve any request to any User Agent (UA).
- All hardware is in use, reducing costs.
- Maintenance is easier, due to all nodes having nearly-identical configurations.

5.2.1. JBoss Communications Failover Capabilities

The SIP Stack used by the JBoss Communications SIP Servlets for JBoss supports two modes:

- `ESTABLISHED SIP DIALOG` failover. This means that failover can occur only on established calls (SIP Dialogs which are in the CONFIRMED state as per RFC 3261) and calls that are in the process of being setup will not be failed over (SIP Dialogs which are in the EARLY state as per RFC 3261).

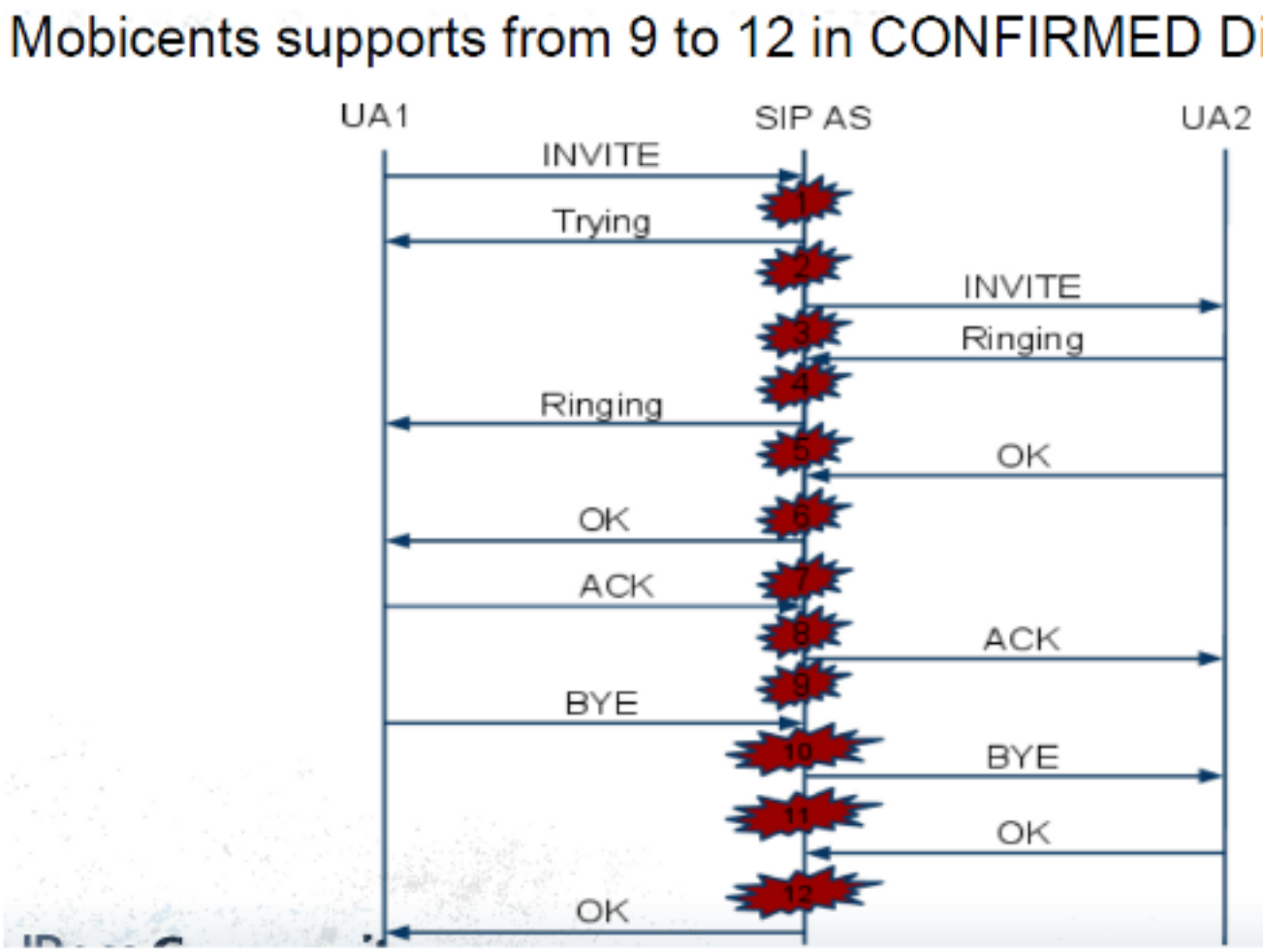


Figure 5.1. Established Dialog Failover

This is the default configuration.

Optimization Note

To maximize performance it is recommended to use a load balancer with SIP affinity enabled, so that all messages related to the same call go to the

same node even though Mobicents Sip Servlets supports the case where SIP transactions for a given SIP Dialog go to different nodes.

- `EARLY SIP DIALOG` failover. This means that failover can occur after an informational response (1xx) is received with a To Tag. For example, calls that are in the process of being setup will be failed over (SIP Dialogs which are in the EARLY state as per RFC 3261).

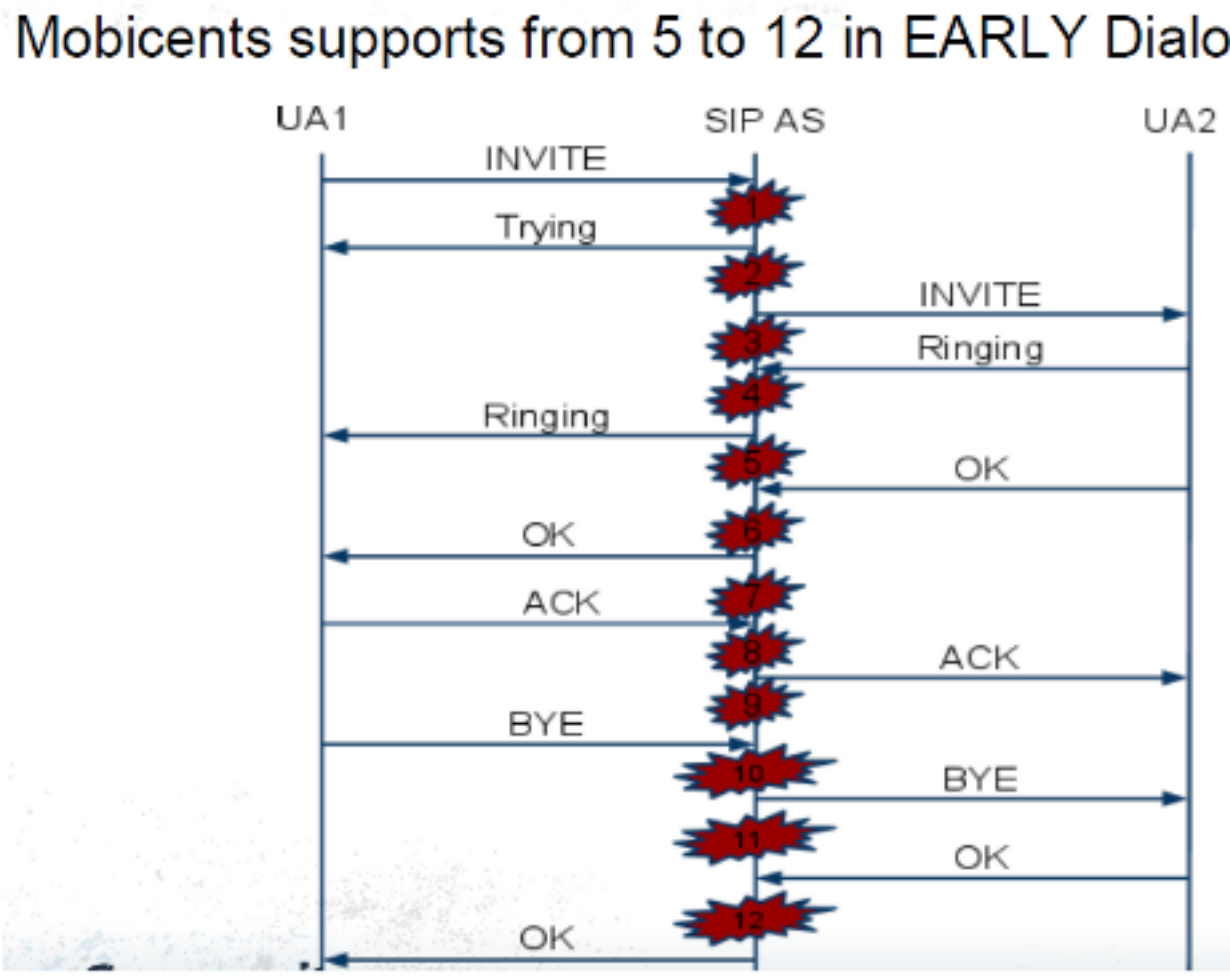


Figure 5.2. Early Dialog Failover

As Early Dialog failover means replicating some transaction states, it can introduce some overhead in terms of network replication as well. The failover granularity is configurable to

best fit applications on their usage and optimize the performance. The following property, `org.mobicents.ha.javax.sip.REPLICATION_STRATEGY=EarlyDialog`, needs to be added to the SIP Stack Stack properties, defined in the external properties file specified by the *sipStackPropertiesFile* attribute as described in *Section 2.3.1, "Configuring SIP Connectors"*.

Optimization Note

To maximize performance it is recommended to use a Load Balancer with SIP affinity enabled so that all messages related to the same SIP transaction (and even SIP call) go to the same node even though Mobicents Sip Servlets supports the case where SIP transactions for a given SIP Dialog go to different nodes.

5.2.2. JBCP SIP Servlets for JBoss Cluster: Installing, Configuring and Running

There are a number of options you can specify for JBCP SIP Servlets clustering. By default, most of them are configured in the "all" server configuration, which is ready to use. In this chapter we will cover the most common configuration options you might need.

5.2.2.1. Downloading

5.2.2.2. Installing

5.2.2.3. Configuring

5.2.2.4. Running

5.2.2.5. Using

5.2.2.6. Testing

5.2.2.7. Uninstalling

5.3. Load Balancer

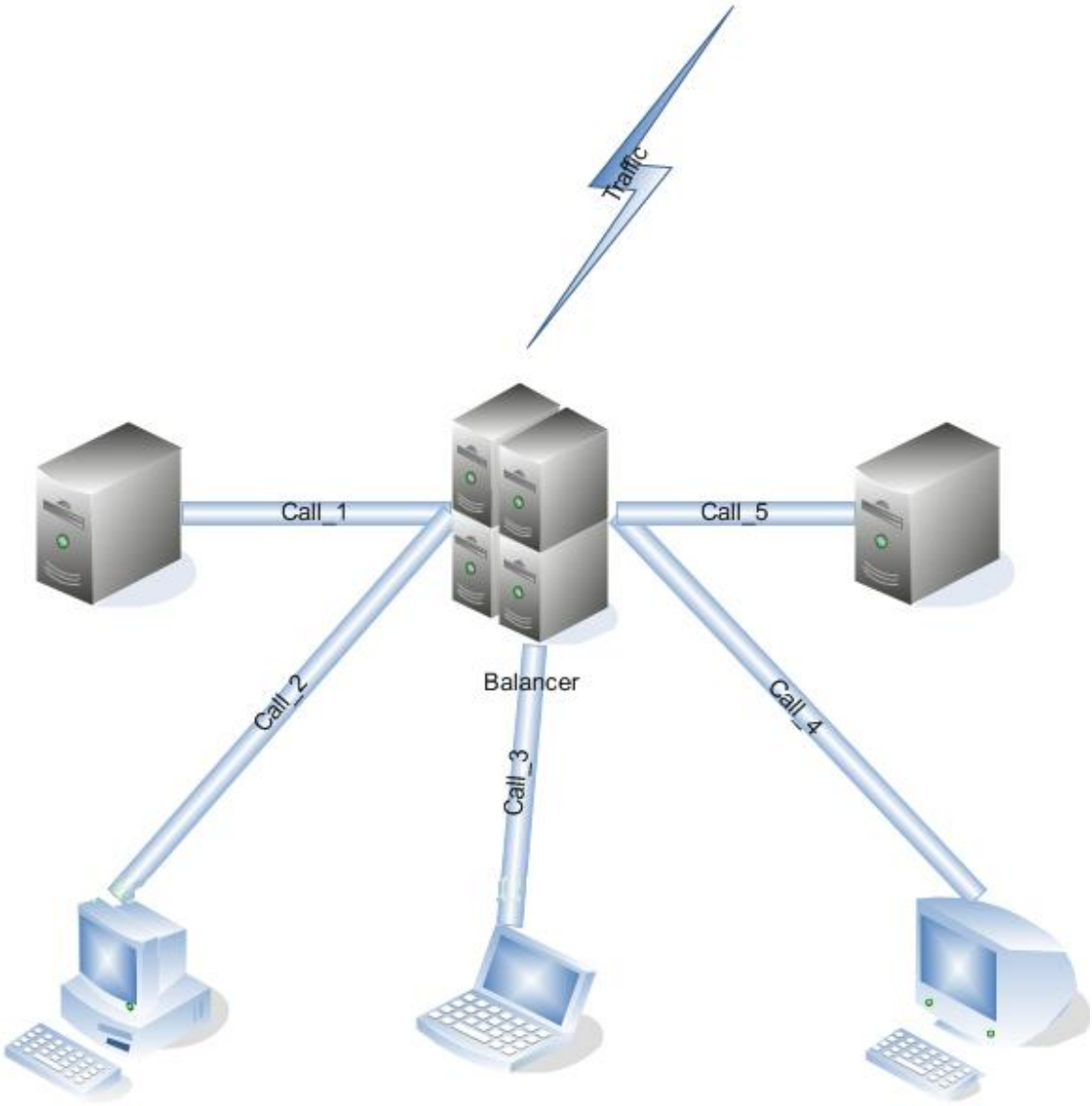


Figure 5.3. Star Cluster Topology.

The SIP Load Balancer is used to balance the load of SIP service requests and responses between nodes in a SIP Servlets Server cluster. Both JBCP SIP Servlets for JBoss and JBCP SIP Servlets for Tomcat servers can be used in conjunction with the SIP Load Balancer to increase the performance and availability of SIP services and applications.

In terms of functionality, the SIP Load Balancer is a simple stateless proxy server that intelligently forwards SIP session requests and responses between User Agents (UAs) on a Wide Area Network (WAN), and SIP Servlets Server nodes, which are almost always located on a Local Area Network (LAN). All SIP requests and responses pass through the SIP Load Balancer.

5.3.1. SIP Load Balancer: Installing, Configuring and Running

5.3.1.1. Pre-Install Requirements and Prerequisites

Software Prerequisites

A JAIN SIP HA-enabled application server such as JBoss Communications JAIN SLEE or JBoss Communications SIP Servlets is required.

Running the SIP Load Balancer requires at least two instances of the application server as cluster nodes nodes. Therefore, before configuring the SIP Load Balancer, we should make sure we've installed a the SIP application server first. The JBoss Communications SIP load balancer will work with a SIP Servlets-enabled JBoss Application Server *or* a JAIN SLEE application server with SIP RA.

SIP Servlets containers based on Tomcat are also supported but the session replication is not available there, thus mid-call failover will not work.

- To install a SIP Servlet-enabled JBoss Application Server, follow the instructions here: *Section 2.1, “SIP Servlet-Enabled JBoss Application Server: Installing, Configuring and Running”*.
- To install a SIP Servlet-enabled Tomcat Servlet Container, follow these instructions: *Section 2.2, “SIP Servlet-Enabled Tomcat Servlet Container: Installing, Configuring and Running”*.

5.3.1.2. Downloading

The load balancer is located in the `sip-balancer` top-level directory of the JBCP SIP Servlets distribution. You will find the following files in the directory:

SIP load balancer executable JAR file
: This is the binary file with all dependencies

SIP load balancer Configuration Properties file
: This is the properties files with various settings

5.3.1.3. Installing

The SIP load balancer executable JAR file can be extracted anywhere in the file system. It is recommended that the file is placed in the directory containing other JAR executables, so it can be easily located in the future.

5.3.1.4. Configuring

Configuring the SIP load balancer and the two SIP Servlets-enabled Server nodes is described in *Configuring the JBoss Communications SIP Load Balancer and SIP Server Nodes*.

Procedure 5.1. Configuring the JBoss Communications SIP Load Balancer and SIP Server Nodes

1. **Configure lb.properties Configuration Properties File**

   Configure the SIP Load Balancer's Configuration Properties file by substituting valid values for your personal setup. *Example 5.1, "Complete Sample lb.properties File"* shows a sample `lb.properties` file, with key element descriptions provided after the example. The lines beginning with the pound sign are comments.

Example 5.1. Complete Sample lb.properties File

```
# Mobicents Load Balancer Settings
# For an overview of the Mobicents Load Balancer visit http://docs.google.com/present/view?id=dc5jp5vx_89cxdvtxcm
# The Load balancer will listen for both TCP and UDP connections



# The binding address of the load balancer. This also specifies the
# default value for both internalHost and externalHost if not specified separately.
host=127.0.0.1

# The binding address of the load balancer where clients should connect (if the host property is not specified)
#externalHost=127.0.0.1

# The SIP port from where servers will receive messages
# delete if you want to use only one port for both inbound and outbound)
internalPort=5065

# The SIP port used where clients should connect
externalPort=5060

# The binding address of the load balancer where SIP application servers should connect (if the host property is not specified)
#internalHost=127.0.0.1

# The RMI port used for heartbeat signals
rmiRegistryPort=2000

# The HTTP port for HTTP forwarding
# if you like to activate the integrated HTTP load balancer, this is the entry point
```

```
httpPort=2080
#If no nodes are active the LB can redirect the traffic to the unavailableHost specified in this
 property,
#otherwise, it will return 503 Service Unavailable
#unavailableHost=google.com

# If you are using IP load balancer, put the IP address and port here
#externalIpLoadBalancerAddress=127.0.0.1
#externalIpLoadBalancerPort=111

# Requests initited from the App Servers can route to this address (if you are using 2 IP load
 balancers for bidirectional SIP LB)
#internalIpLoadBalancerAddress=127.0.0.1
#internalIpLoadBalancerPort=111

# The addresses in the SIP LB Via headers can be either the real addresses or those specified
 in the external and internal IP LB addresses
useIpLoadBalancerAddressInViaHeaders=false

# Designate extra IP addresses as serer nodes
#extraServerNodes=222.221.21.12:21,45.6.6.7:9003,33.5.6.7,33.9.9.2

# Call-ID affinity algortihm settings. This algorithm is the default. No need to uncomment it.
#algorithmClass=org.mobicents.tools.sip.balancer.CallIDAffinityBalancerAlgorithm
# This property specifies how much time to keep an association before being evitcted.
# It is needed to avoid memory leaks on dead calls. The time is in seconds.
#callIdAffinityMaxTimeInCache=500
#The following attribute specified the policy after failover. If set to true all calls from the failed
 node
#will go to a new healthy node (all calls to the same node). If set to false the calls will go to
 random new nodes.
#callIdAffinityGroupFailover=false

# Uncomment to enable the consistent hash based on Call-ID algorithm.
#algorithmClass=org.mobicents.tools.sip.balancer.HeaderConsistentHashBalancerAlgorithm
# This property is not required, it defaults to Call-ID if not set, cna be "from.user" or "to.user"
 when you want the SIP URI username
#sipHeaderAffinityKey=Call-ID
#specify the GET HTTP parameter to be used as hash key
#httpAffinityKey=appsession

# Uncomment to enable the persistent consistent hash based on Call-ID algorithm.
#algorithmClass=org.mobicents.tools.sip.balancer.PersistentConsistentHashBalancerAlgorithm
# This property is not required, it defaults to Call-ID if not set
```

```
#sipHeaderAffinityKey=Call-ID
#specify the GET HTTP parameter to be used as hash key
#httpAffinityKey=appsession

#This is the JBoss Cache 3.1 configuration file (with jgroups), if not specified it will use default
#persistentConsistentHashCacheConfiguration=/home/config.xml

# Call-ID affinity algortihm settings. This algorithm is the default. No need to uncomment it.
#algorithmClass=org.mobicents.tools.sip.balancer.CallIDAffinityBalancerAlgorithm
# This property specifies how much time to keep an association before being evitcted.
# It is needed to avoid memory leaks on dead calls. The time is in seconds.
#callIdAffinityMaxTimeInCache=500

# Uncomment to enable the consistent hash based on Call-ID algorithm.
#algorithmClass=org.mobicents.tools.sip.balancer.HeaderConsistentHashBalancerAlgorithm
# This property is not required, it defaults to Call-ID if not set, cna be "from.user" or "to.user"
 when you want the SIP URI username
#sipHeaderAffinityKey=Call-ID
#specify the GET HTTP parameter to be used as hash key
#httpAffinityKey=appsession

# Uncomment to enable the persistent consistent hash based on Call-ID algorithm.
#algorithmClass=org.mobicents.tools.sip.balancer.PersistentConsistentHashBalancerAlgorithm
# This property is not required, it defaults to Call-ID if not set
#sipHeaderAffinityKey=Call-ID
#specify the GET HTTP parameter to be used as hash key
#httpAffinityKey=appsession

#This is the JBoss Cache 3.1 configuration file (with jgroups), if not specified it will use default
#persistentConsistentHashCacheConfiguration=/home/config.xml


#If a node doesnt check in within that time (in ms), it is considered dead
nodeTimeout=5100
#The consistency of the above condition is checked every heartbeatInterval milliseconds
heartbeatInterval=150


#JSIP stack configuration.....
javax.sip.STACK_NAME = SipBalancerForwarder
javax.sip.AUTOMATIC_DIALOG_SUPPORT = off
# You need 16 for logging traces. 32 for debug + traces.
# Your code will limp at 32 but it is best for debugging.
gov.nist.javax.sip.TRACE_LEVEL = 0
```

```
// Specify if message contents should be logged.
gov.nist.javax.sip.LOG_MESSAGE_CONTENT=false

gov.nist.javax.sip.DEBUG_LOG = logs/sipbalancerforwarderdebug.txt
gov.nist.javax.sip.SERVER_LOG = logs/sipbalancerforwarder.xml
gov.nist.javax.sip.THREAD_POOL_SIZE = 64
gov.nist.javax.sip.REENTRANT_LISTENER = true
```

host
: Local IP address, or interface, on which the SIP load balancer will listen for incoming requests.

externalPort
: Port on which the SIP load balancer listens for incoming requests from SIP User Agents.

internalPort
: Port on which the SIP load balancer forwards incoming requests to available, and healthy, SIP Server cluster nodes.

rmiRegistryPort
: Port on which the SIP load balancer will establish the RMI heartbeat connection to the application servers. When this connection fails or a disconnection instruction is received, an application server node is removed and handling of requests continues without it by redirecting the load to the lie nodes.

httpPort
: Port on which the SIP load balancer will accept HTTP requests to be distributed across the nodes.

internalTransport
: Transport protocol for the internal SIP connections associated with the internal SIP port of the load balancer. Possible choices are `UDP`, `TCP` and `TLS`.

externalTransport
: Transport protocol for the external SIP connections associated with the external SIP port of the load balancer. Possible choices are `UDP`, `TCP` and `TLS`. It must match the transport of the internal port.

externalIpLoadBalancerAddress
: Address of the IP load balancer (if any) used for incoming requests to be distributed in the direction of the application server nodes. This address may be used by the SIP load

balancer to be put in SIP headers where the external address of the SIP load balancer is needed.

externalIpLoadBalancerPort
: The port of the external IP load balancer. Any messages arriving at this port should be distributed across the external SIP ports of a set of SIP load balancers.

internalIpLoadBalancerAddresst
: Address of the IP load balancer (if any) used for outgoing requests (requests initiated from the servers) to be distributed in the direction of the clients. This address may be used by the SIP load balancer to be put in SIP headers where the internal address of the SIP load balancer is needed.

internalIpLoadBalancerPort
: The port of the internal IP load balancer. Any messages arriving at this port should be distributed across the internal SIP ports of a set of SIP load balancers.

extraServerNodes
: Comma-separated list of hosts that are server nodes. You can put here alternative names of the application servers here and they will be recognized. Names are important, because they might be used for direction-analysis. Requests coming from these server will go in the direction of the clients and will not be routed back to the cluster.

algorithmClass
: The fully-qualified Java class name of the balancing algorithm to be used. There are three algorithms to choose from and you can write your own to implement more complex routing behaviour. Refer to the sample configuration file for details about the available options for each algorithm. Each algorithm can have algorithm-specific properties for fine-grained configuration.

nodeTimeout
: In milliseonds. Default value is 5100. If a server node doesnt check in within this time (in ms), it is considered dead.

heartbeatInterval
: In milliseconds. Default value is 150 milliseonds. The hearbeat interval must be much smaller than the interval specified in the JAIN SIP property on the server machines - `org.mobicents.ha.javax.sip.HEARTBEAT_INTERVAL`

Note

The remaining keys and properties in the configuration properties file can be used to tune the JAIN SIP stack, but are not specifically required for load balancing. To assist with tuning, a comprehensive list of implementing classes for the SIP Stack is available from the *Interface SIP Stack page on nist.gov* [http://snad.ncsl.nist.gov/proj/iptel/jain-sip-1.2/

javadoc/javax/sip/SipStack.html]. For a comprehensive list of properties associated with the SIP Stack implementation, refer to *Class SipStackImpl page on nist.gov* [http://snad.ncsl.nist.gov/proj/iptel/jain-sip-1.2/javadoc/gov/nist/javax/sip/SipStackImpl.html].

2. **Configure logging**

   The SIP load balancer uses *Java Logging* [http://java.sun.com/j2se/1.4.2/docs/guide/util/logging/overview.html] as a logging mechanism. You can configure it through a property file and specify the property file to be used by using the following command `-Djava.util.logging.config.file=./lb-logging.properties`. Please refer to JDK logging for more informationon how to configure the Java logging.

3. **Configure the `server.xml` configuration file**

   Ensure the following attributes are configured for the `<service>` element in `server.xml`.

   - The `balancers` attribute must contain a IP address (or list of addresses) of the SIP load balancer(s) to which heartbeat information will be sent.
   - The `sipPathName` attribute must contain the following value `org.mobicents.ha` to indicate that the server will be using the Mobicents JAIN SIP HA SIP Stack which is an extension of the JAIN SIP Stack offering transparent replication.

4. **Configure the `mss-sip-stack.properties` configuration file**

   - The `org.mobicents.ha.javax.sip.cache.MobicentsSipCache.cacheName` property must contain the name of the cache that will be responsible for holding the replicated data of the SIP Stack layer (namely the established SIP dialog data). The value has to be one of the cache name present in the jboss-cache-manager-jboss-beans.xml file of the jboss-cache-manager JBoss Service of the container. The default value is `standard-session-cache`
   - The `org.mobicents.ha.javax.sip.BALANCERS` property must be configured with the list of load balancer IP address and internal ports. As an example, suppose a single SIP Load Balancer is running with IP `192.168.0.1` and internal port `5065`, the property would be set with value `192.168.0.1:5065`. To specify multiple balancers use `;` as separator. If this property is used the balancers attribute located in server.xml should not be used as it is a replacement for it.
   - The `org.mobicents.ha.javax.sip.LoadBalancerHeartBeatingServiceClassName` property is optional, it defines the class name of the HeartBeating service implementation, currently the only one available is `org.mobicents.ha.javax.sip.LoadBalancerHeartBeatingServiceImpl`

- The `org.mobicents.ha.javax.sip.LoadBalancerElector` property is optional, it defines the class of the load balancer elector from JAIN SIP HA Stack. The elector is used to define which load balancer will receive outgoing requests, which are out of dialog or in dialog with null state. Currently only one elector implementation is available, `org.mobicents.ha.javax.sip.RoundRobinLoadBalancerElector`, which, as the class name says, uses round robin algorythm to select the balancer.

Configuration File Locations

On JBCP SIP Servlets for Tomcat server installations, `server.xml` is located in `<install_directory>/conf`.

On JBCP SIP Servlets for JBoss server installations, the default `server.xml` configuration file is located in `server/default/deploy/jboss-web.sar` (or `server/default/deploy/jboss-web.deployer` for JBoss Application Server 4.x and EAP 4.x).

On JBCP SIP Servlets for JBoss installations, with JBoss clustering support enabled, the "all" `server.xml` file must be configured. It is located in `server/all/deploy/jboss-web.deployer`.

To determine what profile should be altered for each JBCP SIP Servlets for JBoss installation, refer to *Section 5.1, "JBoss Communications SIP Servlets for JBoss: Clustering Support"*.

Easy Node Configuration with JMX. Both SIP Servlet-enabled JBoss and Tomcat have JMX (Java Management Extensions) interfaces that allow for easy server configuration. The JMX Console is available once the server has been started by navigating to *http://localhost:8080/jmx-console/*.

Both the `balancers` and `heartBeatInterval` attribute values are available under `name=Mobicents-SIP-Servlets,type=load-balancer-heartbeat-service` in the JMX Console.

balancers
: Host names of the SIP load balancer(s) with corresponding `addBalancerAddress` and `removeBalancerAddress` methods.

heartBeatInterval
: Interval at which each heartbeat is sent to the SIP load balancer(s).

5.3.1.4.2. Converged Load Balancing

5.3.1.4.2.1. Apache HTTP Load Balancer

The JBCP SIP Servlets SIP Load Balancer can work in concert with HTTP load balancers such as `mod_jk`. Whenever an HTTP session is bound to a particular node, an instruction is sent to the SIP Load Balancer to direct the SIP calls from the same application session to the same node.

It is sufficient to configure `mod_jk` to work for HTTP in JBoss in order to enable cooperative load balancing. JBCP SIP Servlets will read the configuration and will use it without any extra configuration. You can read more about configuring `mod_jk` with JBoss in your JBoss Application Server documentation.

Alternatively you may disable this behaviour and make the HTTP load balancer follow the decisions made by the SIP load balancer with the httpFollowsSip flag. This is achieved by changing the jvmRoute part of the session ID cookie used internally by `mod_jk`.

5.3.1.4.2.1.1. The httpFollowsSip flag

The `httpFollowsSip` flag in the service configuration makes the application server aware of how different mod_jk and SIP load balancers have assigned request affinity for each application session. The application servers assign exactly one node to each Sip Servlets application session and this node is the node where the last SIP request associated with the application session has landed (decised by the SIP load balancer). Then the application server will actively update the session ID cookie (the jvmRoute part) of any HTTP request that arrives at the wrong node. The application server will do so with a specially composed HTTP redirect response or with a HTML refresh hint. As a backup strategy, if the request is bound to seek non-existing node forever and it will let the request be served by a new node. This avoids having a client stuck reloading the same page over and over.

One problem with this flag is that if you have two or more SIP sessions associated with the same application session and the load balancer has decided to send SIP requests to different nodes, which might happend if you use Call-ID based affinity, then the application server will have to change the jvmRoute very often for every SIP request resulting in significant overhead. It is generally not adviced to enable this flag if you have more than 1 SIP session per application session and the means to guarantee all SIP sessions from the application session will land on the same node.

This is an example how to enable the option. It is disabled by default.

```
<Connector port="5080"
    ipAddress = "${jboss.bind.address}"
    ...
    httpFollowsSip="true" />
```

5.3.1.4.2.2. Integrated HTTP Load Balancer

To use the integrated HTTP Load Balancer, no extra configuration is needed. If a unique `jvmRoute` is specified and enabled in each application server, it will behave exactly as the apache balancer. If `jvmRoute` is not present, it will use the session ID as a hash value and attempt to create a sticky session. The integrated balancer can be used together with the apache balancer at the same time.

In addition to the apache behavior, there is a consistent hash balancer algorithm that can be enabled for both HTTP and SIP messages. For both HTTP and SIP messages, there is a configurable affinity key, which is evaluated and hashed against each unassigned request. All requests with the same hash value will always be routed to the same application server node. For example, the SIP affinity key could be the callee user name and the HTTP affinity key could the “appsession” HTTP GET parameter of the request. If the desired behaviour group these requests, we can just make sure the affinity values (user name and GET parameter) are the same.

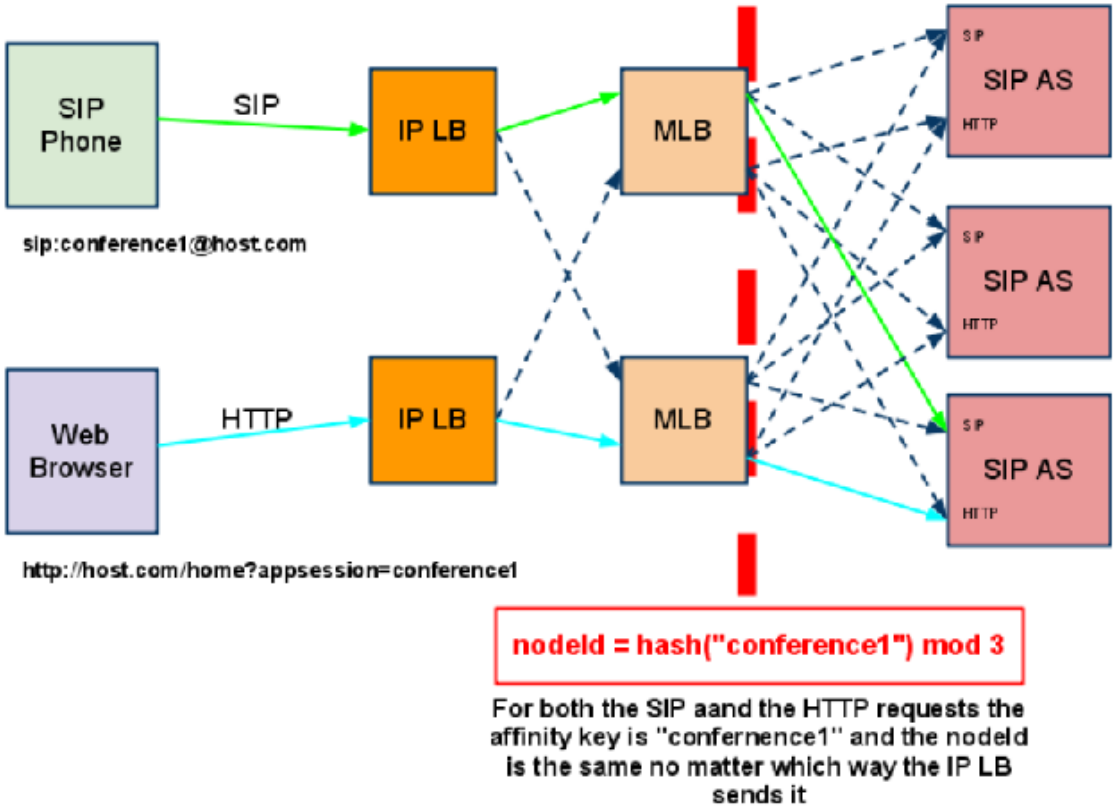


Figure 5.4. Ensuring SIP and HTTP requests are being grouped by common affinity value.

5.3.1.5. Running

Procedure 5.2. Running the SIP Load Balancer and SIP Server Nodes

1. **Start the SIP Load Balancer**

   Start the SIP load balancer, ensuring the Configuration Properties file (`lb.properties` in this example) is specified. In the Linux terminal, or using the Windows Command Prompt, the SIP Load Balancer is started by issuing a command similar to this one:

   ```
   java -jar sip-balancer-jar-with-dependencies.jar lb-
   configuration.properties
   ```

   Executing the SIP load balancer produces output similar to the following example:

   ```
   home]$ java -jar sip-balancer-jar-with-dependencies.jar lb-
   configuration.properties
   Oct 21, 2008 1:10:58 AM
    org.mobicents.tools.sip.balancer.SIPBalancerForwarder start
   INFO: Sip Balancer started on address 127.0.0.1, external port : 5060,
    port : 5065
   Oct 21, 2008 1:10:59 AM
    org.mobicents.tools.sip.balancer.NodeRegisterImpl startServer
   INFO: Node registry starting...
   Oct 21, 2008 1:10:59 AM
    org.mobicents.tools.sip.balancer.NodeRegisterImpl startServer
   INFO: Node expiration task created
   Oct 21, 2008 1:10:59 AM
    org.mobicents.tools.sip.balancer.NodeRegisterImpl startServer
   INFO: Node registry started
   ```

   The output shows the IP address on which the SIP Load Balancer is listening, as well as the external and internal listener ports.

2. **Configure SIP Server Nodes**

   SIP Servlets Server nodes can run on the JBoss Application Server, or the Tomcat Servlet Container. The SIP Servlets Server binary distributions define the type of SIP Servlets Server nodes used, and should already be installed from *Software Prerequisites*.

   The `server.xml` file specifies the nodes used. Because there is more then one client node specified, unique listener ports must be specified for each node to monitor HTTP and/or SIP connections. *Example 5.2, "Changing the SIP Connector Port for Servlet Server Nodes in server.xml"* describes the affected element in the `server.xml` file.

Configuration File Location

For the JBoss SIP Servlets Server binary distribution, `server.xml` is located in the `<install_directory>/server/all/deploy/jboss-web.deployer/` directory (for JBoss Application Server 4.x or EAP 4.x `<install_directory>/server/all/deploy/jboss-web.deployer/`). For the Tomcat binary distribution, `server.xml` is located in the `<install_directory>/conf/` directory.

Example 5.2. Changing the SIP Connector Port for Servlet Server Nodes in server.xml

```
<!-- Define a SIP Connector -->
<Connector port="5080"
```

3. **Start Load Balancer Client Nodes**

   Start all SIP load balancer client nodes.

5.3.1.6. Testing

To test load balancing, the same application must be deployed manually on each node, and two SIP Softphones must be installed.

Procedure 5.3. Testing Load Balancing

1. **Deploy an Application**

   Ensure that for each node, the DAR file location is specified in the `server.xml` file.

   Deploy the Location service manually on both nodes.

2. **Start the "Sender" SIP softphone**

   Start a SIP softphone client with the SIP address of **`sip:sender@sip-servlets-com`**, listening on port 5055. The outbound proxy must be specified as the sip-balancer (http://127.0.0.1:5060)

3. **Start the "Receiver" SIP softphone**

   Start a SIP softphone client with the SIP address of **`sip:receiver-failover@sip-servlets-com`**, listening on port 5090.

4. **Initiate two calls from "Sender" SIP softphone**

   Initiate one call from **`sip:sender@sip-servlets-com`** to **`sip:receiver-failover@sip-servlets-com`**. Tear down the call once completed.

   Initiate a second call using the same SIP address, and tear down the call once completed. Notice that the call is handled by the second node.

5.3.1.7. Stopping

Assuming that you started the JBoss Application Server as a foreground process in the Linux terminal, the easiest way to stop it is by pressing the **Ctrl**+**C** key combination in the same terminal in which you started it.

This should produce similar output to the following:

```
^COct 21, 2008 1:11:57 AM
 org.mobicents.tools.sip.balancer.SipBalancerShutdownHook run
INFO: Stopping the sip forwarder
```

5.3.1.8. Uninstalling

To uninstall the SIP load balancer, delete the JAR file you installed.

5.3.2. IP Load Balancing

5.3.2.1. IP Load Balancers

An IP load-balancer is a network appliance that distributes traffic to an application server (or actual servers) using a load-balancing algorithm. IP load-balancing is often used when the other load-balancers' capacity is exceeded and can not scale further without hardware upgrades.

Routing decisions are made based on OSI Layer 2, 3 or 4 data. This type of load balancer only examines low-level TCP, UDP or ethernet packet structures including MAC addresses, IP addresses, ports, and protocol types (TCP or UDP or other).

An IP load balancer never reads the payload of the TCP/IP packets and therefore never parses SIP or HTTP (or any protocol above OSI Layer 4). Because an IP load balancing device is not SIP or HTTP aware in any way, it is much more performant than `mod_jk` or the JBCP SIP Servlets SIP load-balancer.

5.3.2.2. Technical overview

In its simplest form, the IP load-balancer usually "owns" the public-facing IP address (known as a VIP). The traffic is routed to actual servers in it's private network similar to NAT. It is also possible to not change the IP address and just work on the MAC address by assuming that all actual servers are configured to accept packets for the VIP address. The features offered by the IP load balancer depend largely on the vendor.

Some examples of Linux-based software load balancers include *Red Hat Cluster Suite (RHCS)* [http://www.redhat.com/cluster_suite/] and *Linux Virtual Server (LVS)* [http://www.linuxvirtualserver.org/]. There are many hardware vendors as well.

One main drawback relating to IP load balancers is that they can not make routing decisions based on SIP messages and (with some exceptions) they can not work cooperatively with HTTP or other load balancers.

5.3.2.3. Configuring JBCP SIP Servlets Cluster for pure IP Load Balancing

Warning

Pure IP load balancing is not a recommented option. It is advised to use a distributed load balancer instead. Proper operation with pure IP load balancing depends on the ability of the IP load balancer to establish request affinity based on IP addresses and ports.

5.3.2.4.

First you need to remove the SIP load balancers from any configuration in JBCP SIP Servlets. In particular the `balancers` attribute in `server.xml`. and edit the jboss.web engine tag. You should remove the balancers attribute from the Service tag of jboss.web service. This simply removes the default load balancer from the system and the traffic bypasses the SIP load-balancer. Next you must configure JBCP SIP Servlets to put the IP load balancer IP address in the `Via`, `Contact` and other system headers where the IP address of the server machine is required. This will ensure that any responses or subsequent SIP requests follow the same path, but always hit the load balancer instead of particular cluster node that may fail. To specify the IP load balancer address in JBCP SIP Servlets your should edit this file `JBOSS_HOME/server/all/deploy/jboss-web.deployer/server.xml` and specify `staticServerAddress` such as:

```
<Connector port="5080"
    ipAddress = "${jboss.bind.address}"
    ...
staticServerAddress="122.122.122.122" staticServerPort="44"
useStaticAddress="true"/>
```

Note

Depending on your reliability requirements you can omit the configuration described in this section and let the servers use their own IP address in the SIP messages.

5.3.3. SIP Load Balancing Basics

All User Agents send SIP messages, such as `INVITE` and `MESSAGE`, to the same SIP URI (the IP address and port number of the SIP Load Balancer on the WAN). The Load Balancer then parses, alters, and forwards those messages to an available node in the cluster. If the message was sent as a part of an existing SIP session, it will be forwarded to the cluster node which processed that User Agent's original transaction request.

The SIP Server that receives the message acts upon it and sends a response back to the SIP Load Balancer. The SIP Load Balancer reparses, alters and forwards the message back to the original User Agent. This entire proxying and provisioning process is carried out independent of the User Agent, which is only concerned with the SIP service or application it is using.

By using the Load Balancer, SIP traffic is balanced across a pool of available SIP Servers, increasing the overall throughput of the SIP service or application running on either individual nodes of the cluster. In the case of a JBCP SIP Servlets server with `</distributed>` capabilities, load balancing advantages are applied across the entire cluster.

The SIP Load Balancer is also able to failover requests mid-call from unavailable nodes to available ones, thus increasing the reliability of the SIP service or application. The Load Balancer increases throughput and reliability by dynamically provisioning SIP service requests and responses across responsive nodes in a cluster. This enables SIP applications to meet the real-time demand for SIP services.

5.3.4. HTTP Load Balancing Basics

In addition to the SIP load balancing, there are several options for coordinated or cooperative load balancing with other protocols such as HTTP.

Typically, a JBoss Application Server will use apache HTTP server with mod_jk, mod_proxy, mod_cluster or similar extension installed as an HTTP load balancer. This apache-based load balancer will parse incoming HTTP requests and will look for the session ID of those requests in order to ensure all requests from the same session arrive at the same application server.

By default, this is done by examining the `jsessionid` HTTP cookie or GET parameter and looking for the `jvmRoute` assigned to the session. The typical `jsessionid` value is of the form `<sessionId>.<jvmRoute>`. The very first request for each new HTTP session does not have a session ID assigned; the apache routes the request to a random application server node.

When the node responds it assigns a session ID and `jvmRoute` to the response of the request in a HTTP cookie. This response goes back to the client through apache, which keeps track of

which node owns each `jvmRoute`. Once the very first request is served this way, the subsequent requests from this session will carry the assigned cookie, and the apache load balancer will always route the requests to the node, which advertised itself as the `jvmRoute` owner.

Instead of using apache, an integrated HTTP Load Balancer is also available. The SIP Load Balancer has a HTTP port where you can direct all incoming HTTP requests. The integrated HTTP load balancer behaves exactly like apache by default, but this behavior is extensible and can be overridden completely with the pluggable balancer algorithms. The integrated HTTP load balancer is much easier to configure and generally requires no effort, because it reuses most SIP settings and assumes reasonable default values.

Unlike the native apache, the integrated HTTP Load Balancer is written completely in Java, thus a performance penalty should be expected when using it. However, the integrated HTTP Balancer has an advantage when related SIP and HTTP requests must stick to the same node.

5.3.5. Pluggable balancer algorithms

The SIP/HTTP Load Balancer exposes an interface to allow users to customize the routing decision making for special purposes. By default there are three built-in algorithms. Only one algorithm is active at any time and it is specified with the `algorithmClass` property in the configuration file.

It is up to the algorithm how and whether to support distributed architecture or how to store the information needed for session affinity. The algorithms will be called for every SIP and HTTP request and other significant events to make more informed decisions.

Note

Users must be aware that by default requests explicitly addressed to a live server node passing through the load balancer will be forwarded directly to the server node. This allows for pre-specified routing use-cases, where the target node is known by the SIP client through other means. If the target node is dead, then the node selection algorithm is used to route the request to an available node.

The following is a list of the built-in algorithms:

org.mobicents.tools.sip.balancer.CallIDAffinityBalancerAlgorithm
: This algorithm is not distributable. It selects nodes randomly to serve a give Call-ID extracted from the requests and responses. It keeps a map with `Call-ID -> nodeId` associations and this map is not shared with other load balancers which will cause them to make different decisions. For HTTP it behaves like apache.

org.mobicents.tools.sip.balancer.HeaderConsistentHashBalancerAlgorithm
: This algorithm is distributable and can be used in distributed load balancer configurations. It extracts the hash value of specific headers from SIP and HTTP messages to decide which

application server node will handle the request. Information about the options in this algorithms is available in the balancer configuration file comments.

org.mobicents.tools.sip.balancer.PersistentConsistentHashBalancerAlgorithm
: This algorithm is distributable and is similar to the previous algorithm, but it attempts to keep session affinity even when the cluster nodes are removed or added, which would normally cause hash values to point to different nodes.

org.mobicents.tools.sip.balancer.ClusterSubdomainAffinityAlgorithm
: This algorithm is not distributable, but supports grouping server nodes to act as a subcluster. Any call of a node that belongs to a cluster group will be preferentially failed over to a node from the same group. To configure a group you can just add the `subclusterMap` property in the load balancer properties and listing the IP addresses of the nodes. The groups are enclosed in parentheses and the IP addresses are separate by commas as follows:

```
subclusterMap=( 192.168.1.1, 192.168.1.2 ) ( 10.10.10.10,  20.20.20.20,
 30.30.30.30)
```

The nodes specified in a group do not have to alive and nodes that are not specified are still allowed to join the cluster. Otherwise the algorthim behaves exactly as the default Call-ID affinity algorthim.

5.3.6. Distributed load balancing

When the capacity of a single load balancer is exceeded, multiple load balancers can be used. With the help of an IP load balancer the traffic can be distributed between all SIP/HTTP load balancers based on some IP rules or round-robin. With consistent hash and `jvmRoute`-based balancer algorithms it doesn't matter which SIP/HTTP load balancer will process the request, because they would all make the same decisions based on information in the requests (headers, parameters or cookies) and the list of available nodes. With consistent hash algorithms there is no state to be preserved in the SIP/HTTP balancers.

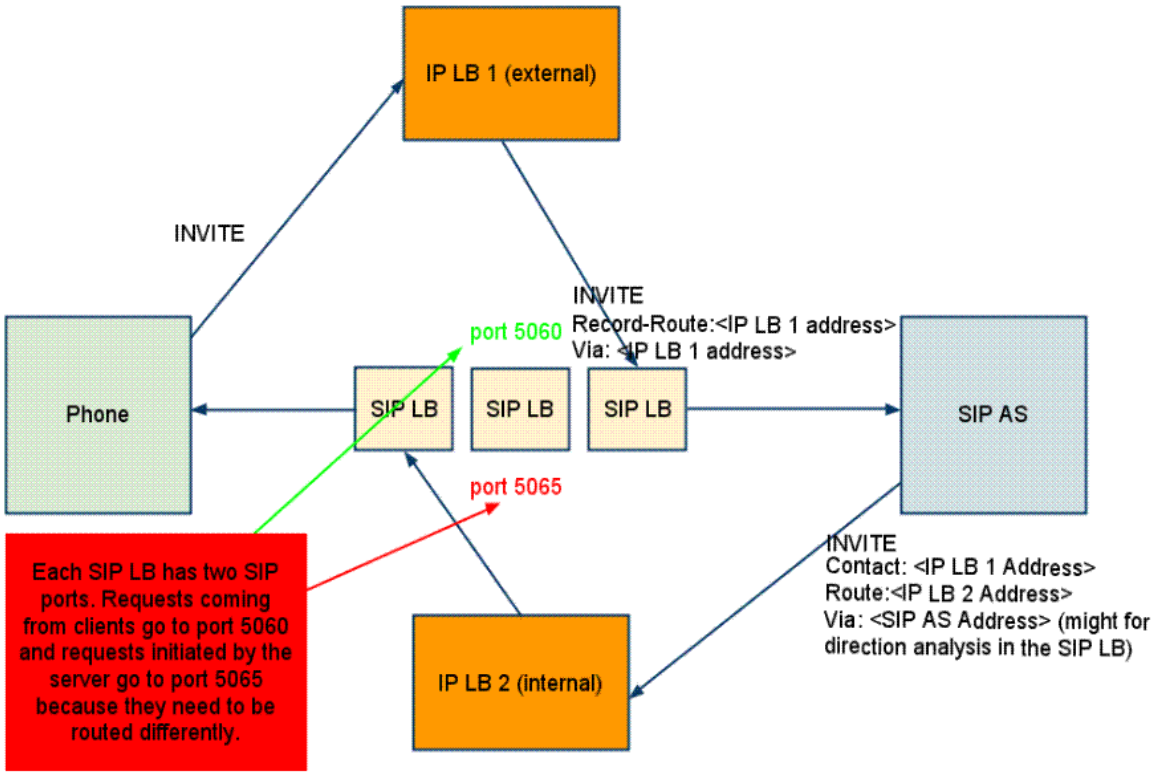


Figure 5.5. Example deployment: IP load balancers serving both directions for incoming/outgoing requests in a cluster

5.3.7. Implementation of the JBoss Communications Load Balancer

Each individual JBoss Communications SIP Server in the cluster is responsible for contacting the SIP load balancer and relaying its health status and regular "heartbeats".

From these health status reports and heartbeats, the SIP Load Balancer creates and maintains a list of all available and healthy nodes in the cluster. The Load Balancer forwards SIP requests between these cluster nodes, providing that the provisioning algorithm reports that each node is healthy and is still sending heartbeats.

If an abnormality is detected, the SIP Load Balancer removes the unhealthy or unresponsive node from the list of available nodes. In addition, mid-session and mid-call messages are failed over to a healthy node.

The SIP Load Balancer first receives SIP requests from endpoints on a port that is specified in its Configuration Properties configuration file. The SIP Load Balancer, using a round-robin algorithm, then selects a node to which it forwards the SIP requests. The Load Balancer forwards all same-session requests to the first node selected to initiate the session, providing that the node is healthy and available.

5.3.8. SIP Message Flow

The SIP Load Balancer appends itself to the `Via` header of each request, so that returned responses are sent to the SIP Balancer before they are sent to the originating endpoint.

The Load Balancer also adds itself to the path of subsequent requests by adding Record-Route headers. It can subsequently handle mid-call failover by forwarding requests to a different node in the cluster if the node that originally handled the request fails or becomes unavailable. The SIP load balancer immediately fails over if it receives an unhealthy status, or irregular heartbeats from a node.

In advanced configurations, it is possible to run more than one SIP Load Balancer. Simply edit the balancers connection string in your SIP Server - the list is separated with semi-colon.

Figure 5.6, "Basic IP and Port Cluster Configuration" describes a basic IP and Port Cluster Configuration. In the diagram, the SIP Load balancer is the server with the IP address of `192.168.1.1`.

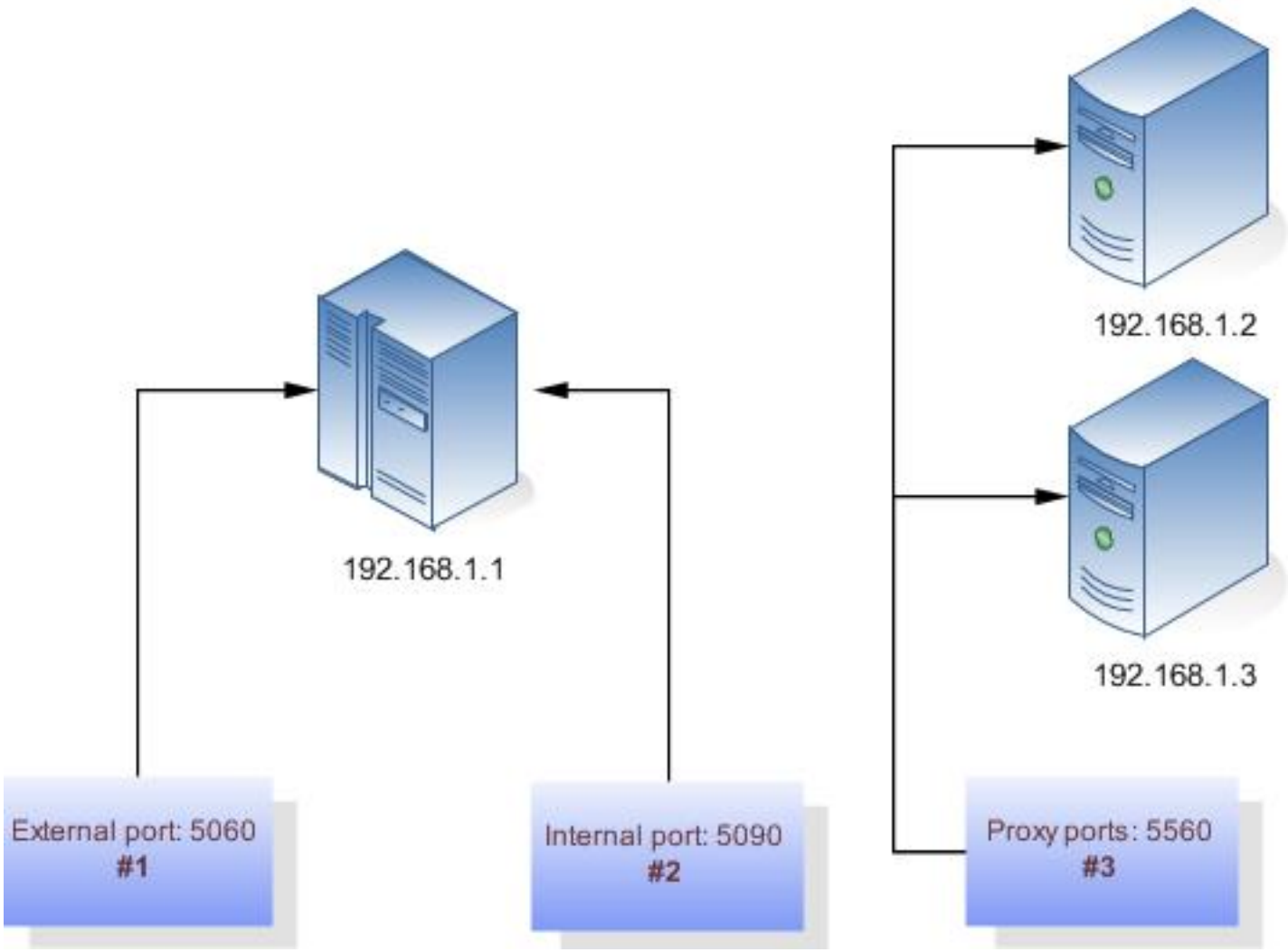


Figure 5.6. Basic IP and Port Cluster Configuration

Enterprise Monitoring and Management

There is two ways of monitoring Mobicents Sip Servlets :

- Through the industry standard Simple Network Management Protocol - SNMP
- Through Jopr is an enterprise management solution for JBoss middleware projects and other application technologies. This pluggable project provides administration, monitoring, alerting, operational control and configuration in an enterprise setting with fine-grained security and an advanced extension model.

  It provides support for monitoring base operating system information on six operating systems as well as management of Apache, JBoss Application Server (JBoss AS) and other related projects. See the *Jopr website* [http://www.jboss.org/jopr] or the *Jopr embedded website* [http://www.jboss.org/embjopr] for more information.

6.1. JBoss Communications SIP Servlets SNMP Monitoring and Management

This chapter provides information on how to enable the management of JBoss Communications SIP Servlets Servers through SNMP. It also allows applications to expose their metrics and management through SNMP

See *the full guide presents SNMP Monitoring for JBoss Application Server* [https://docs.jboss.org/author/display/MOBICENTS/SNMP+Adaptor]

6.2. JBoss Communications SIP Servlets Jopr Monitoring and Management

This chapter provides information on how to enable the management of JBoss Communications SIP Servlets Servers through Jopr with our custom Jopr plug in. Two versions of Jopr are available: an embedded version, which is better suited to development environments; and a full version, which is better suited to production environments.

The JBoss Communications SIP Servlet Jopr plugin provides a facility to view metrics related to the deployed applications, metrics related to the SIP Servlets Server. Additionally, the plugin provides the option to manage the various configuration settings of the SIP Servlets Server such as Congestion and Concurrency control.

6.2.1. Installation of the Enterprise Monitoring and Management Console

6.2.1.1. Jopr for Development

The Embedded Jopr (also known as the JBoss Administration Console) is pre-installed as an application deployed in the JBoss Communications SIP Servlets Server.

6.2.1.2. Jopr for Production

- Follow the *Jopr installation instructions* [http://jboss.org/community/docs/DOC-12828] to install the latest version of Jopr.
- Stop the Jopr server and agent if they are running.
- Get the JBoss Communications SIP Servlets Jopr Plug in from *jboss.com* [http://repository.jboss.com/maven2/org/mobicents/servlet/sip/jopr-mobicents-sip-servlets-plugin/1.3/jopr-mobicents-sip-servlets-plugin-1.3.jar].
- Copy the jopr-mobicents-sip-servlets-plugin-1.3.jar just downloaded to the `jopr-server/jbossas/server/default/deploy/rhq.ear/rhq-downloads/rhq-plugins/` directory
- Start the Jopr server then the agent.
- Start the JBoss Communications SIP Servlets Server on JBoss, and ensure the binding address is specified.

  `sh run.sh -b 192.168.0.10` (the Server will not get recognized on localhost)

6.2.2. Usage Instructions

6.2.2.1. Jopr for Development

- Log in to the Jopr console on *http://localhost:8080/admin-console* and login as admin (user=admin, password=admin).
- From the tree on the left side of the screen, under Services , click on the **MobicentsSipServlets** link, then click on the **jboss.web:type=SipApplicationDispatcher** link to view the incoming metrics.

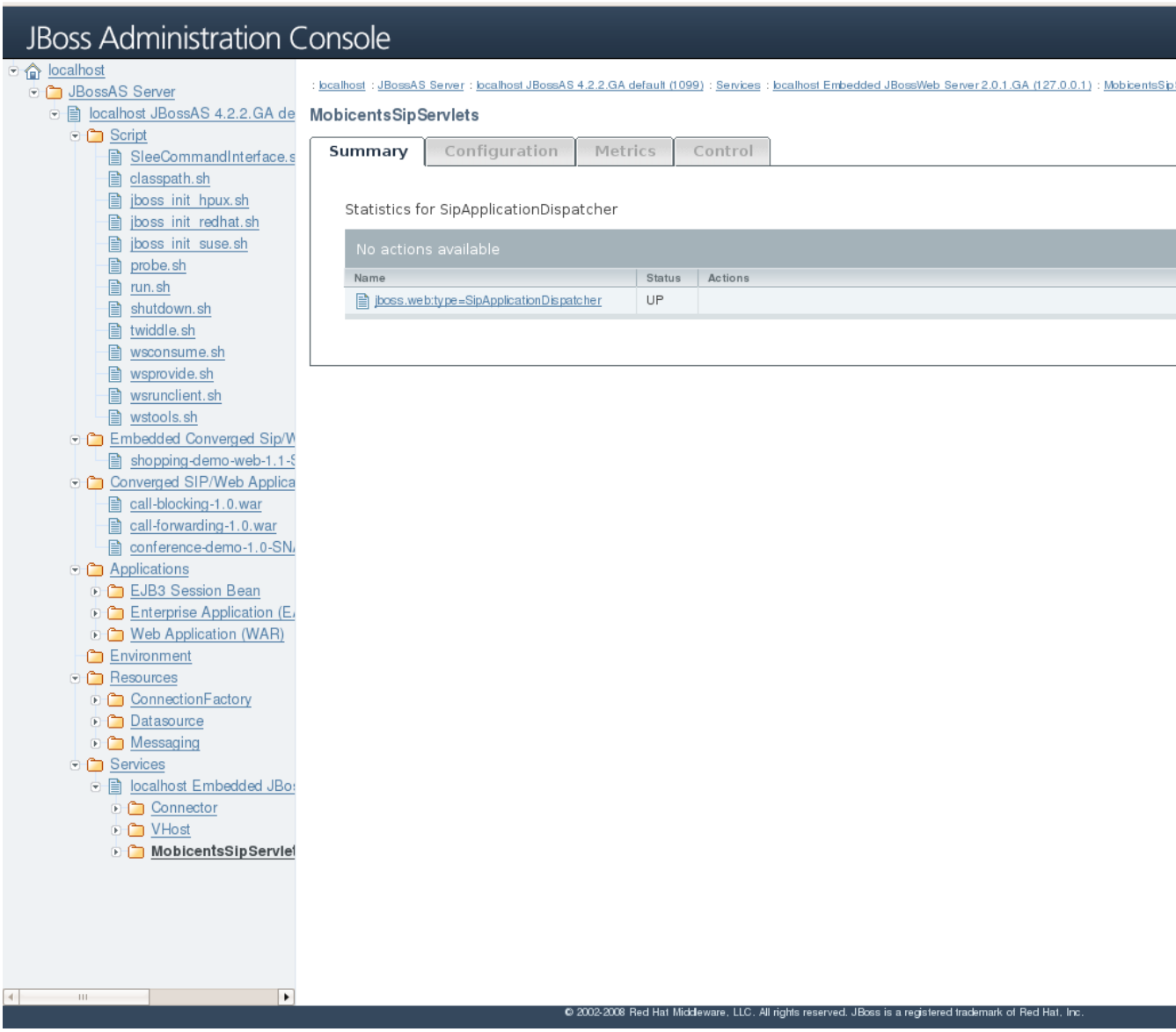


Figure 6.1. Mobicents SIP Servlets Server Metrics

- Click on the Configuration Tab to tune the Congestion Controls parameters as defined in *Section 7.2, "Concurrency and Congestion Control"*

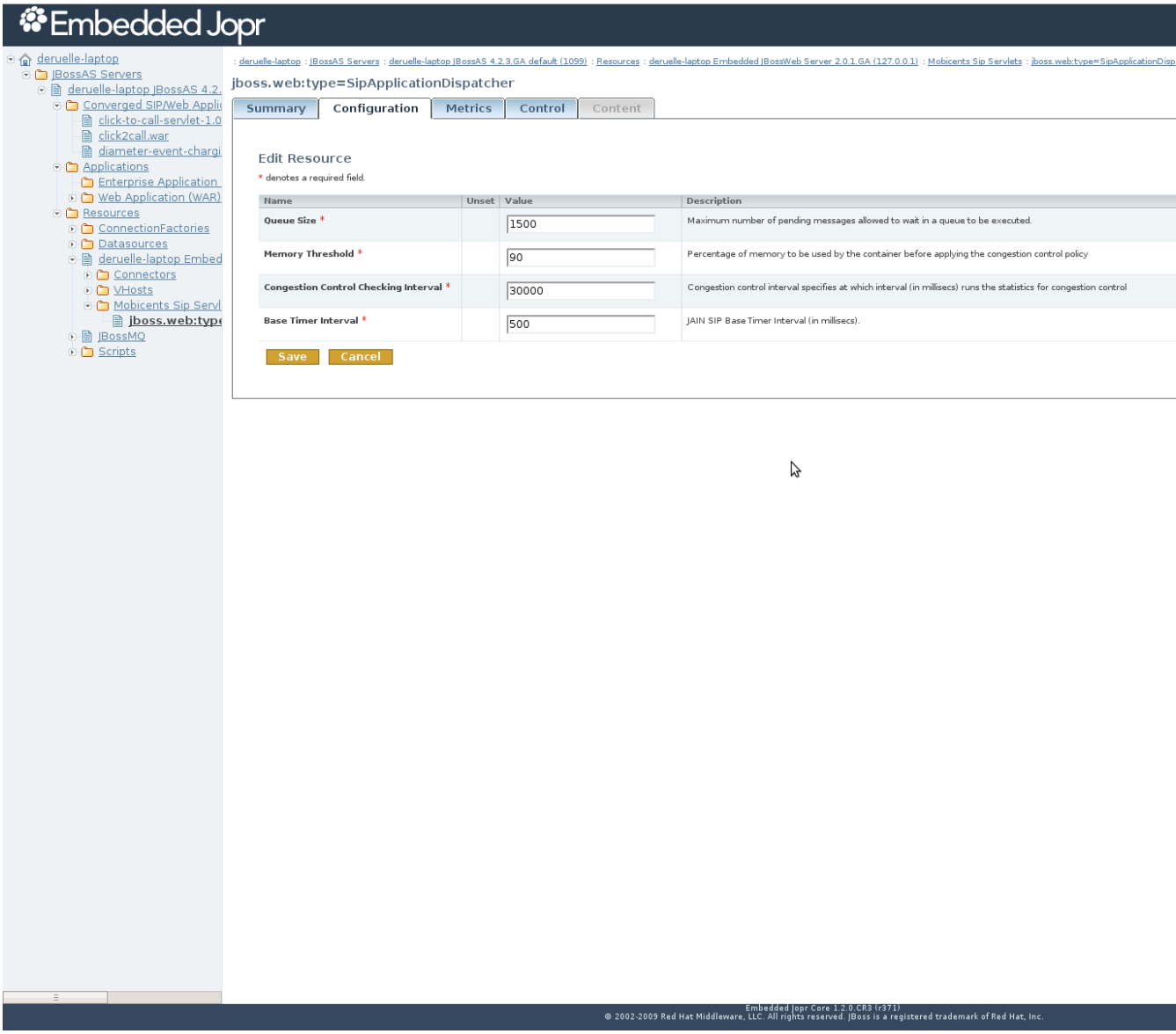


Figure 6.2. SIP Application Dispatcher Congestion Control Parameters Configuration

- Click on the **Control Tab** to set the Concurrency Control Mode and Congestion Control Policy as defined in *Section 7.2, "Concurrency and Congestion Control"*

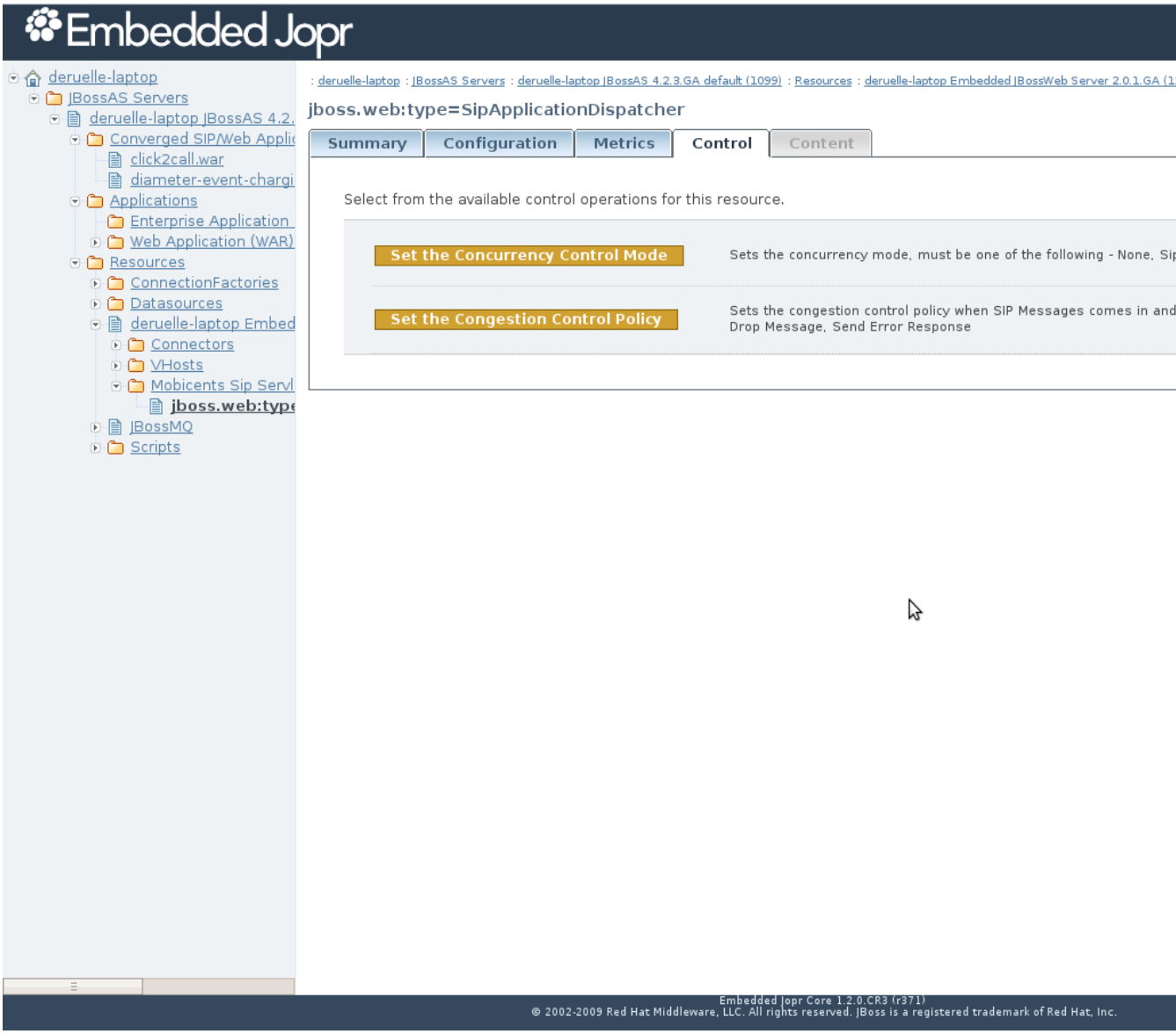


Figure 6.3. SIP Application Dispatcher Congestion Control Parameters Configuration

- To begin metrics collection, and see them on the monitoring application, you must use an example application (such as the location service) so that the SIP Servlets Server processes SIP Messages.

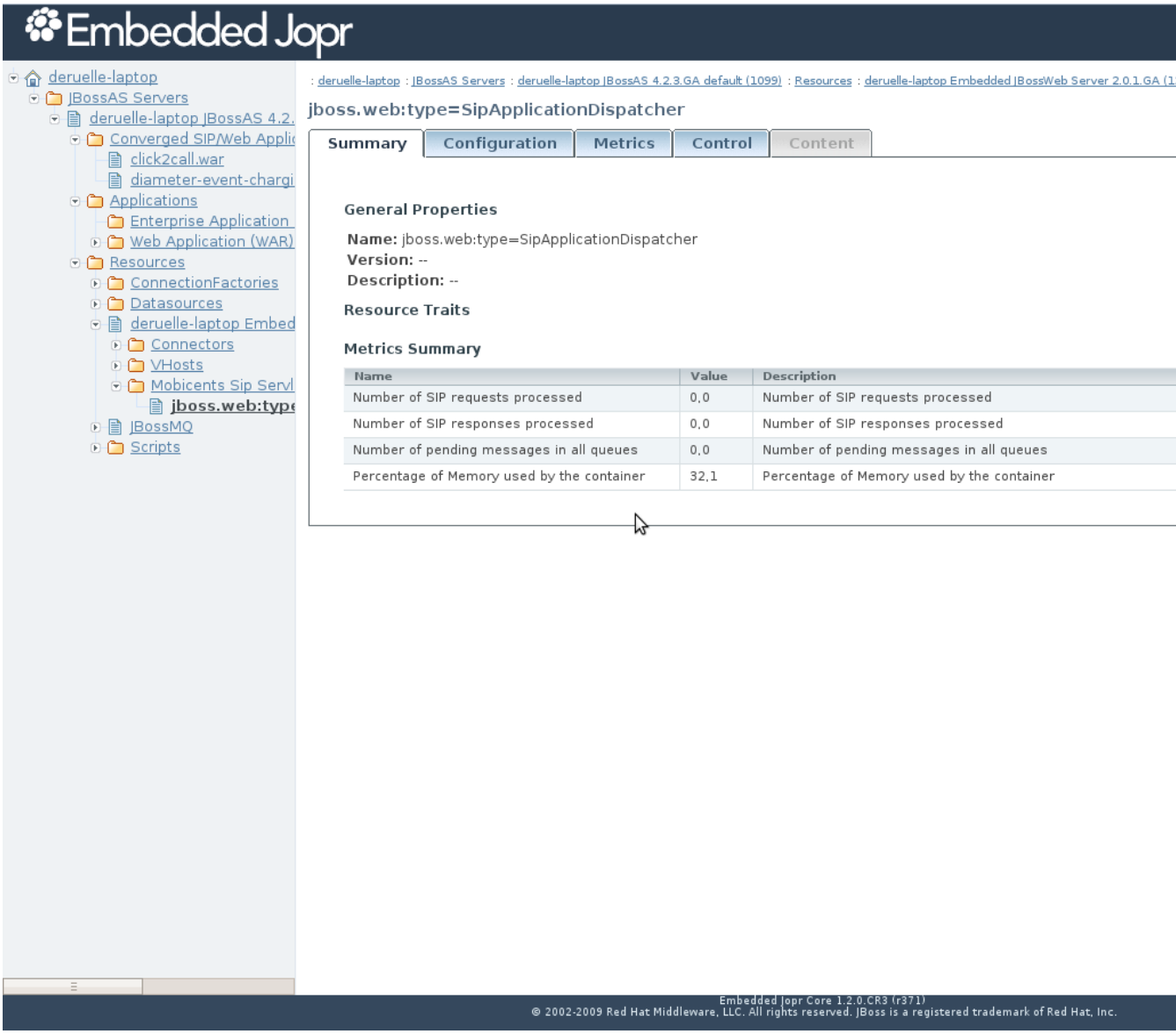


Figure 6.4. Selected Application Metrics

- To see metrics for the application, click on the application under the Converged SIP/Web Application (SAR/WAR) link.

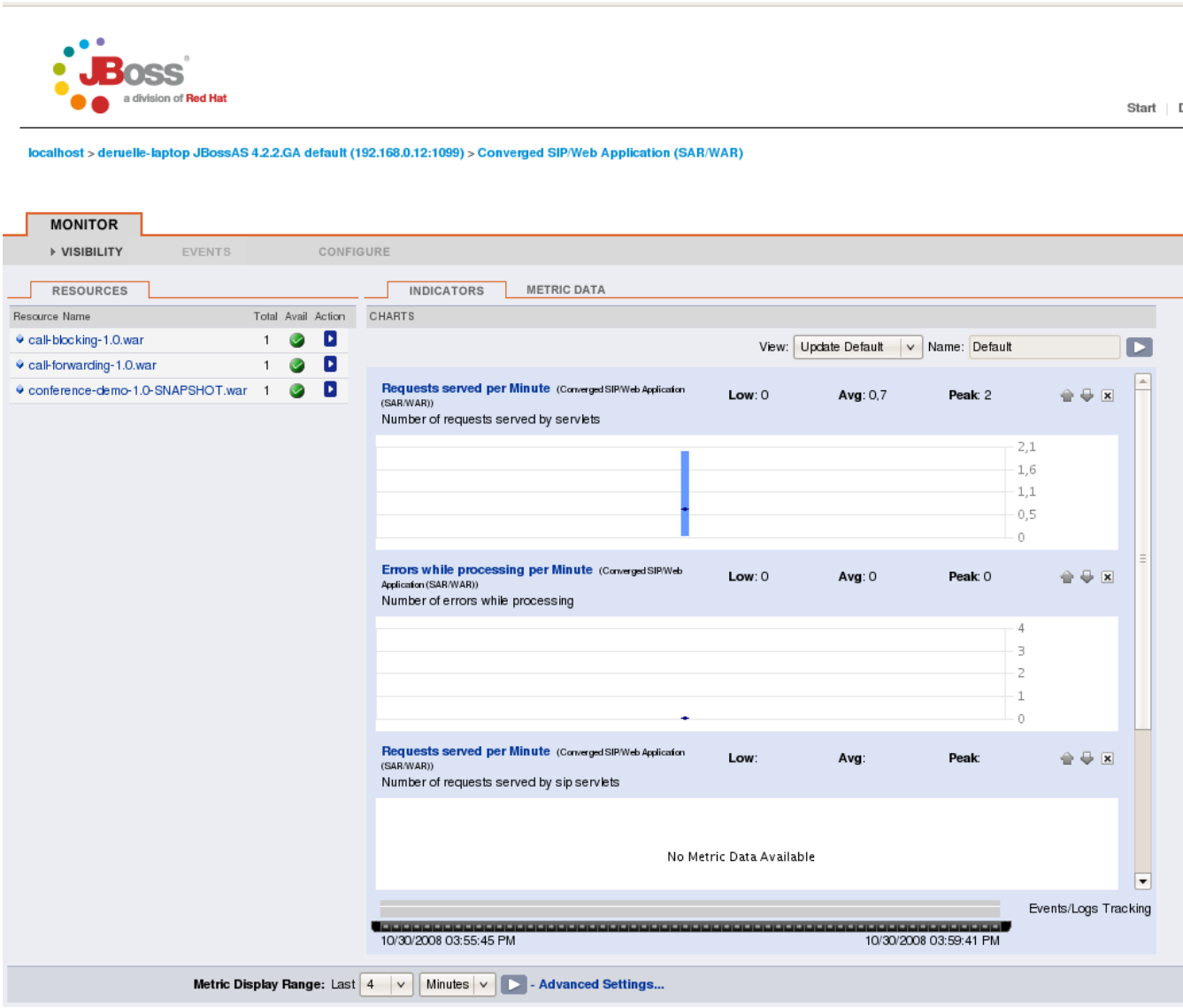


Figure 6.5. Specific Application Metrics

- Select the **Metrics** tab to actually see the metrics for the application.

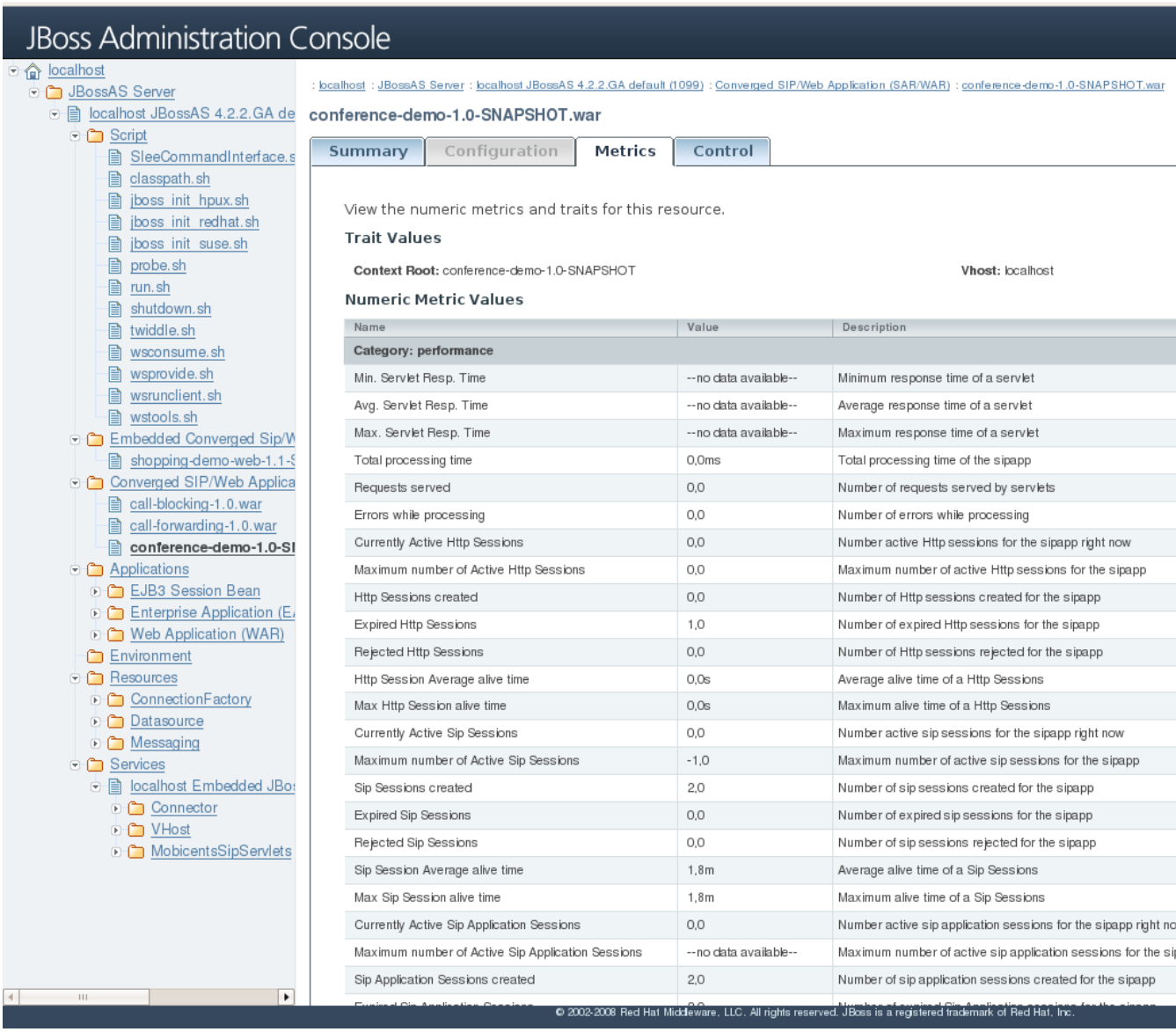


Figure 6.6. Specific Application Metrics Detail

6.2.2.2. Jopr For Production

- Log in to the Jopr console on *http://localhost:7080* [http://localhost:7080/].

  From the **Dashboard** tab, in the Auto Discovery portlet, import your server (for example, deruelle-laptop JBossAS 4.2.3.GA default (192.168.0.12:1099)) from under localhost.

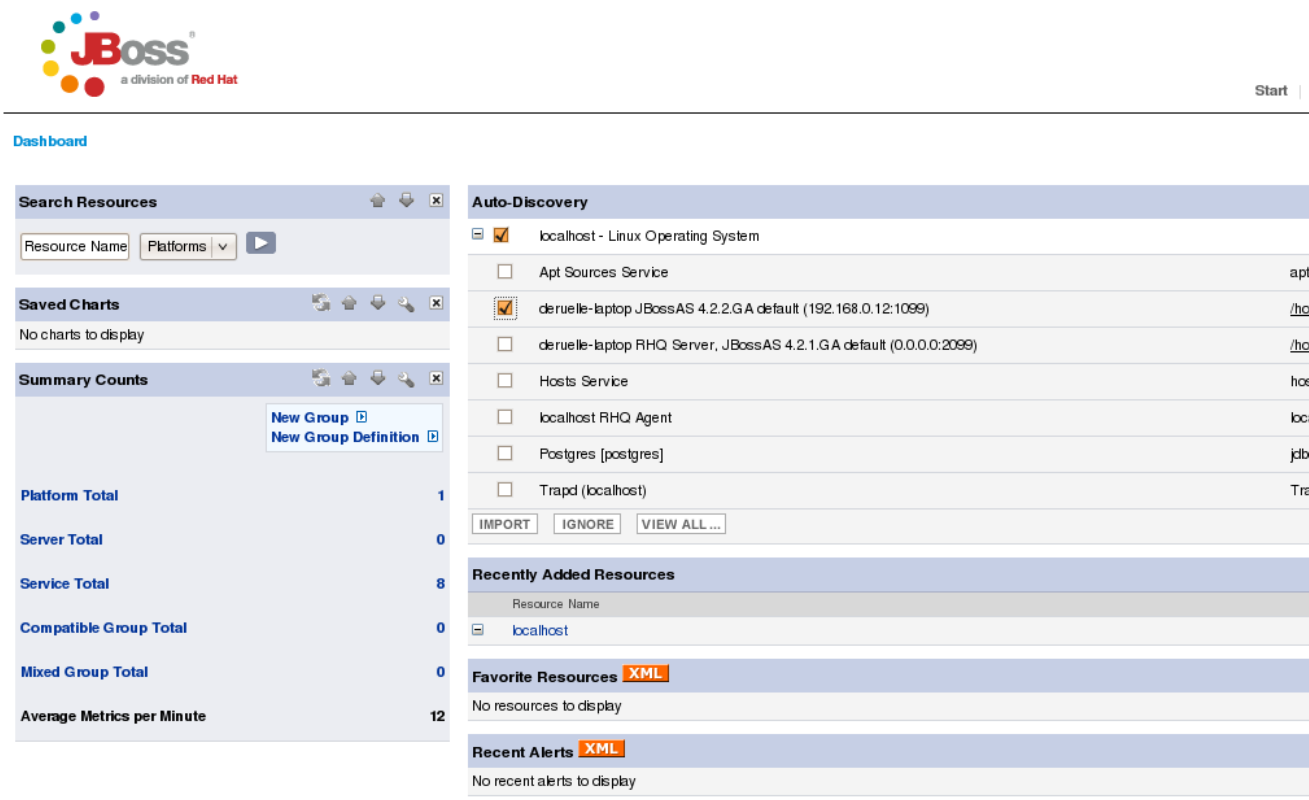


Figure 6.7. Server Import for Monitoring

- From the Dashboard, in the Recently Added Resources portlet, click on the server.

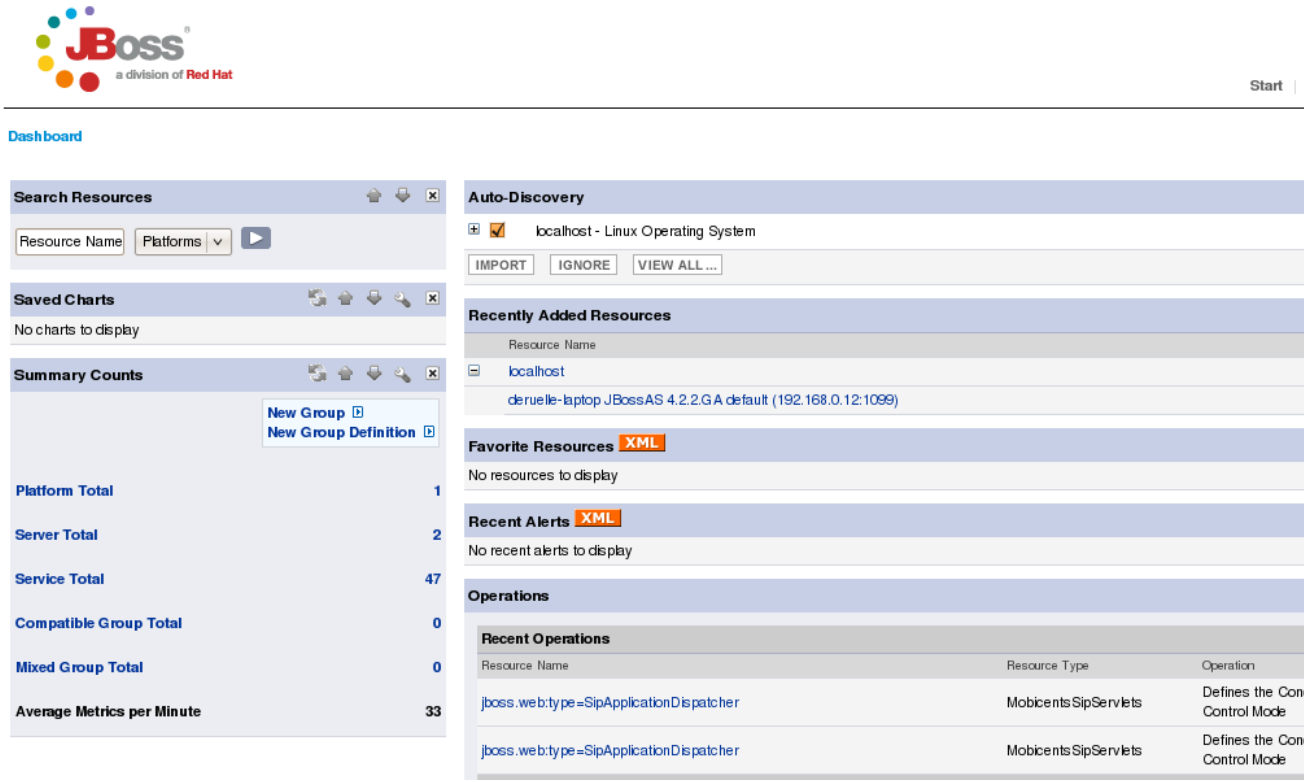


Figure 6.8. Selecting the Server for Monitoring

- On the new Monitor view click on the JBossWeb Server link.

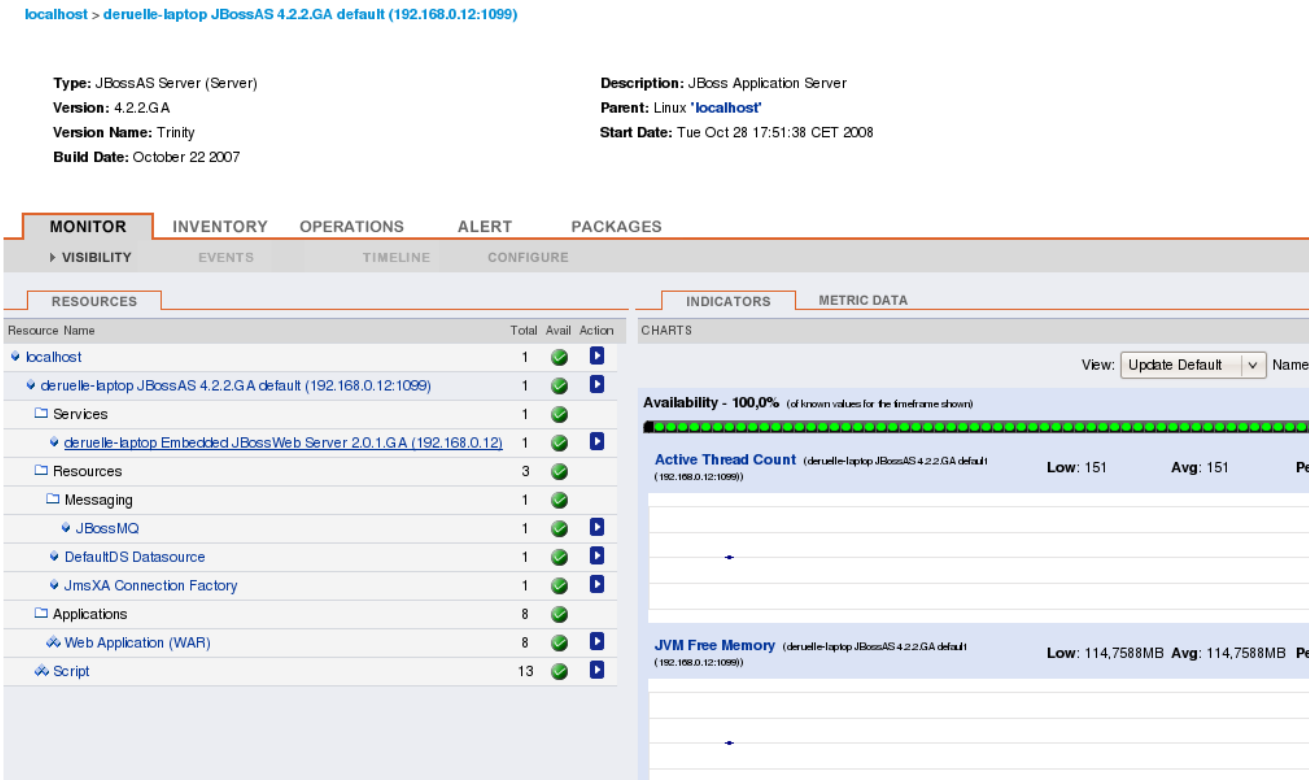


Figure 6.9. Servlet Container Monitoring

- Click on the **jboss.web:type=SipApplicationDispatcher** link to view the incoming metrics.

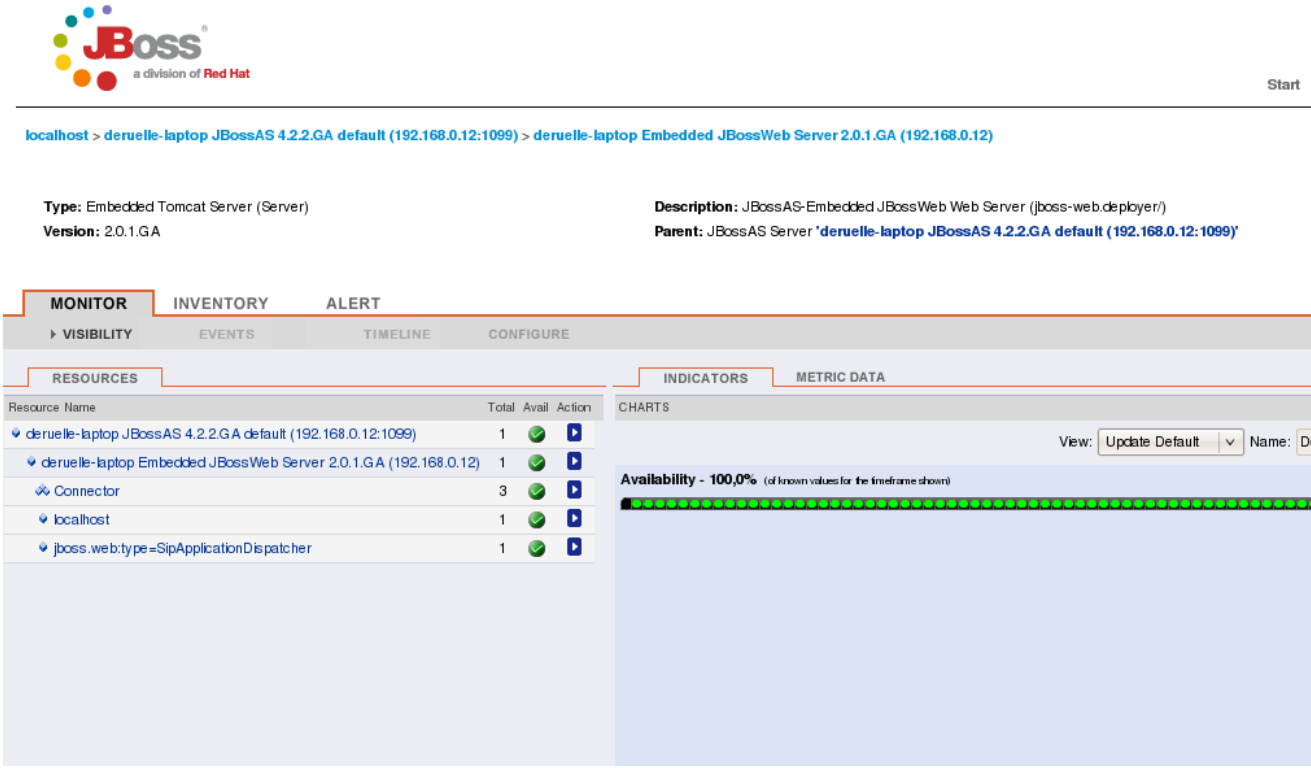


Figure 6.10. Mobicents SIP Servlets Server Metrics

- Click on the **Configuration** tab to tune the Container configuration parameters such as Concurrency and Congestion as defined in *Section 7.2, "Concurrency and Congestion Control"*

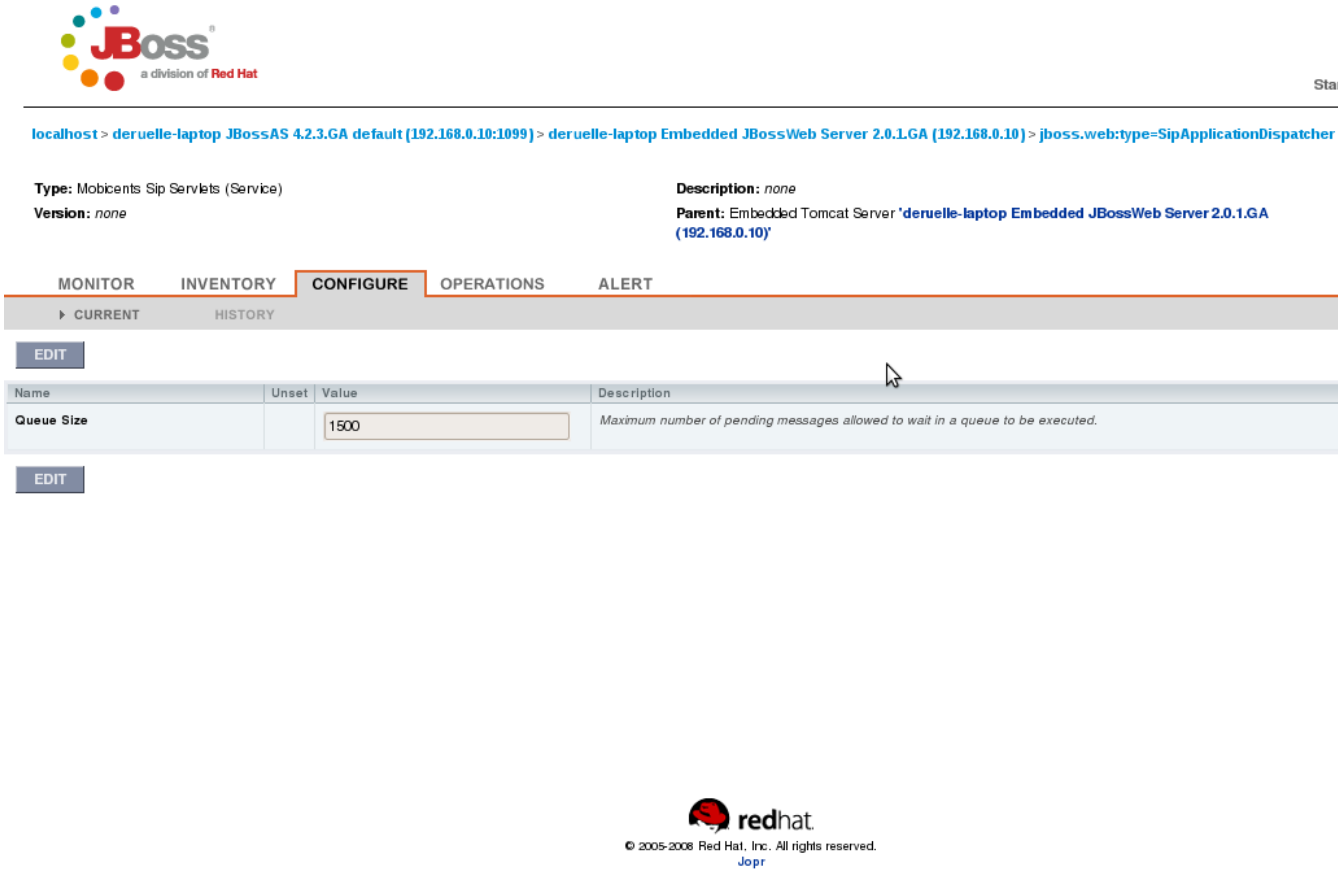


Figure 6.11. SIP Application Dispatcher Configuration

- Click on the Control Tab to be able to set the Concurrency Control Mode and Congestion Control Policy as defined in *Section 7.2, "Concurrency and Congestion Control"*.

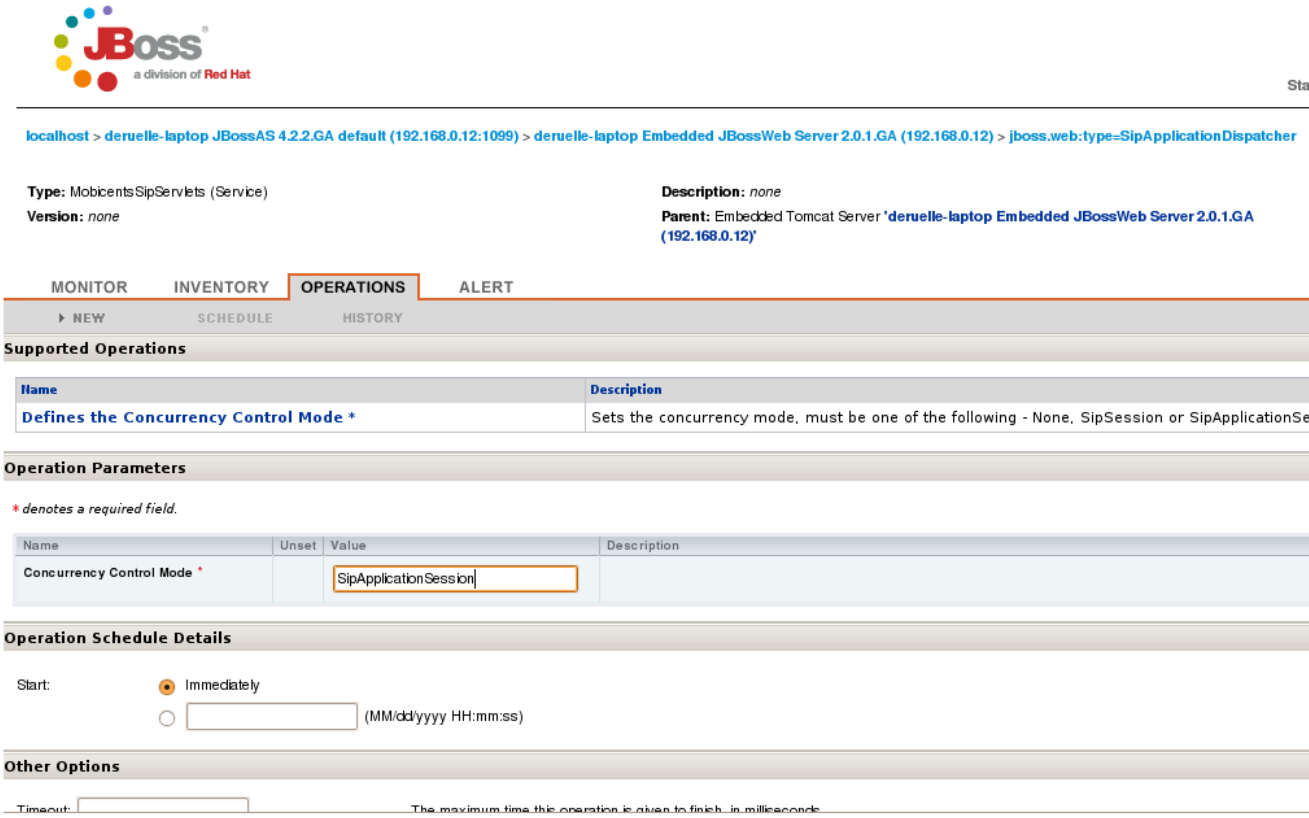


Figure 6.12. SIP Application Dispatcher Congestion Control Parameters Configuration

- To begin metrics collection, and see them on the monitoring application, you must use an example application (such as location service) so that the SIP Servlets Server processes SIP Messages.

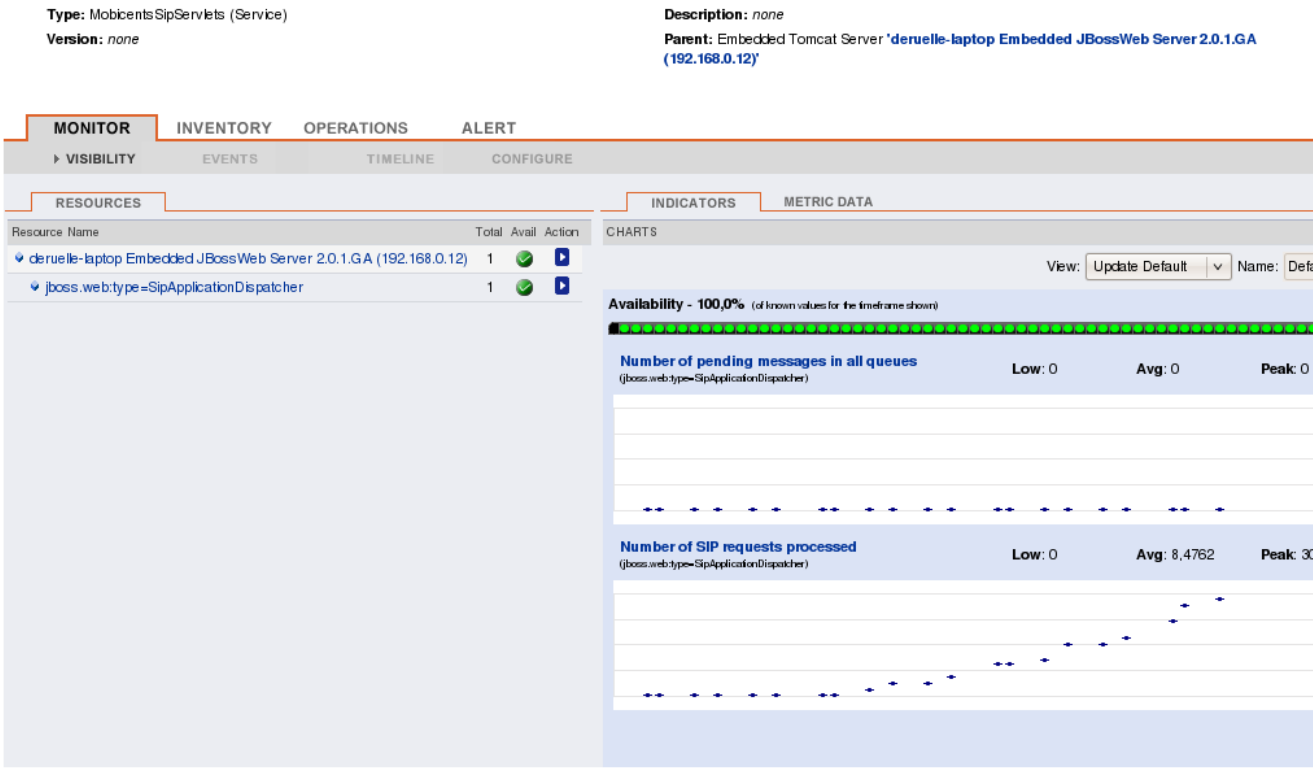


Figure 6.13. Selected Application Metrics

- To see Metrics about your application, click on the Converged SIP/Web Application (SAR/WAR) link.

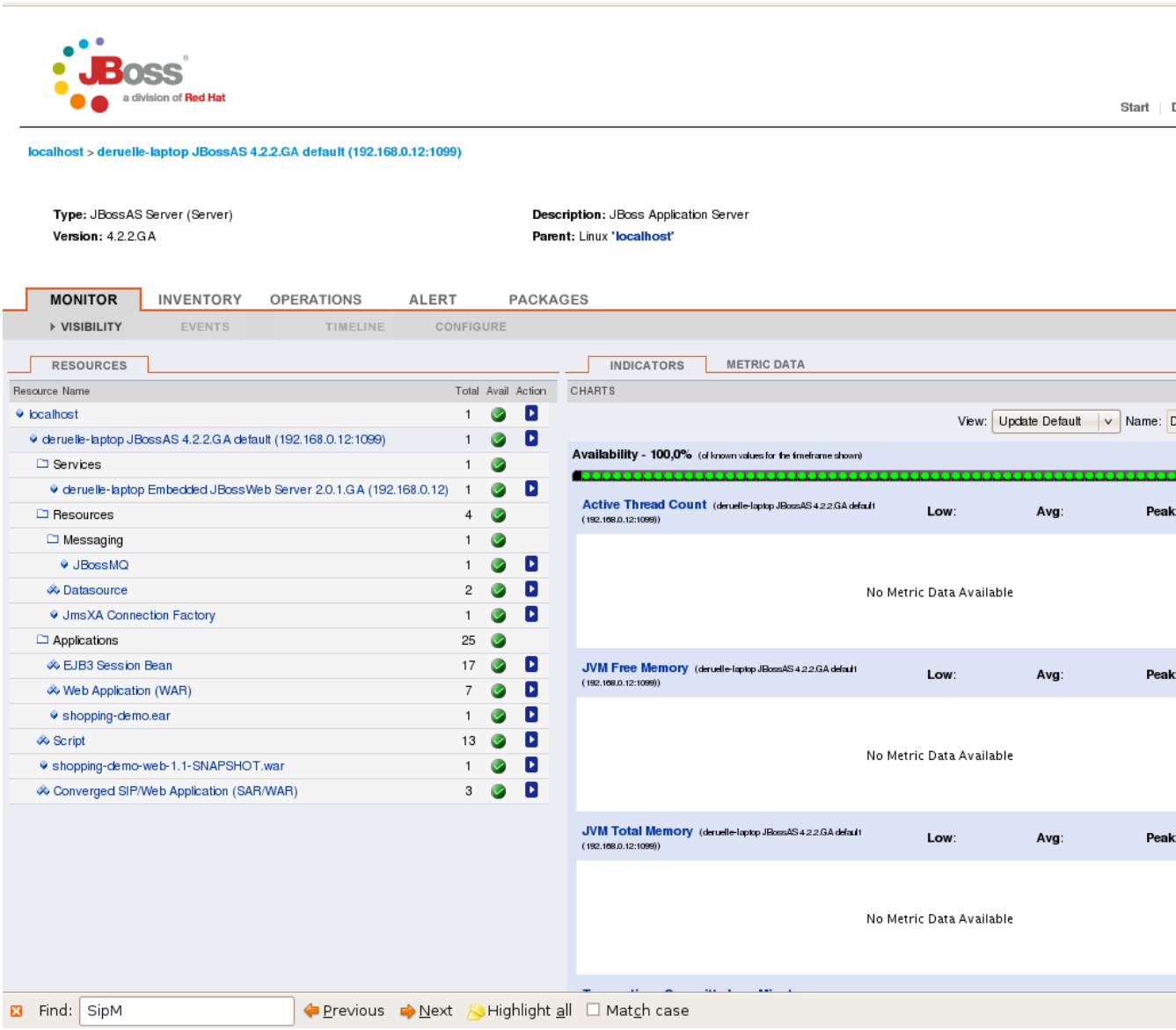


Figure 6.14. Specific Application Metrics

- Click on the application to actually see the metrics for the application.

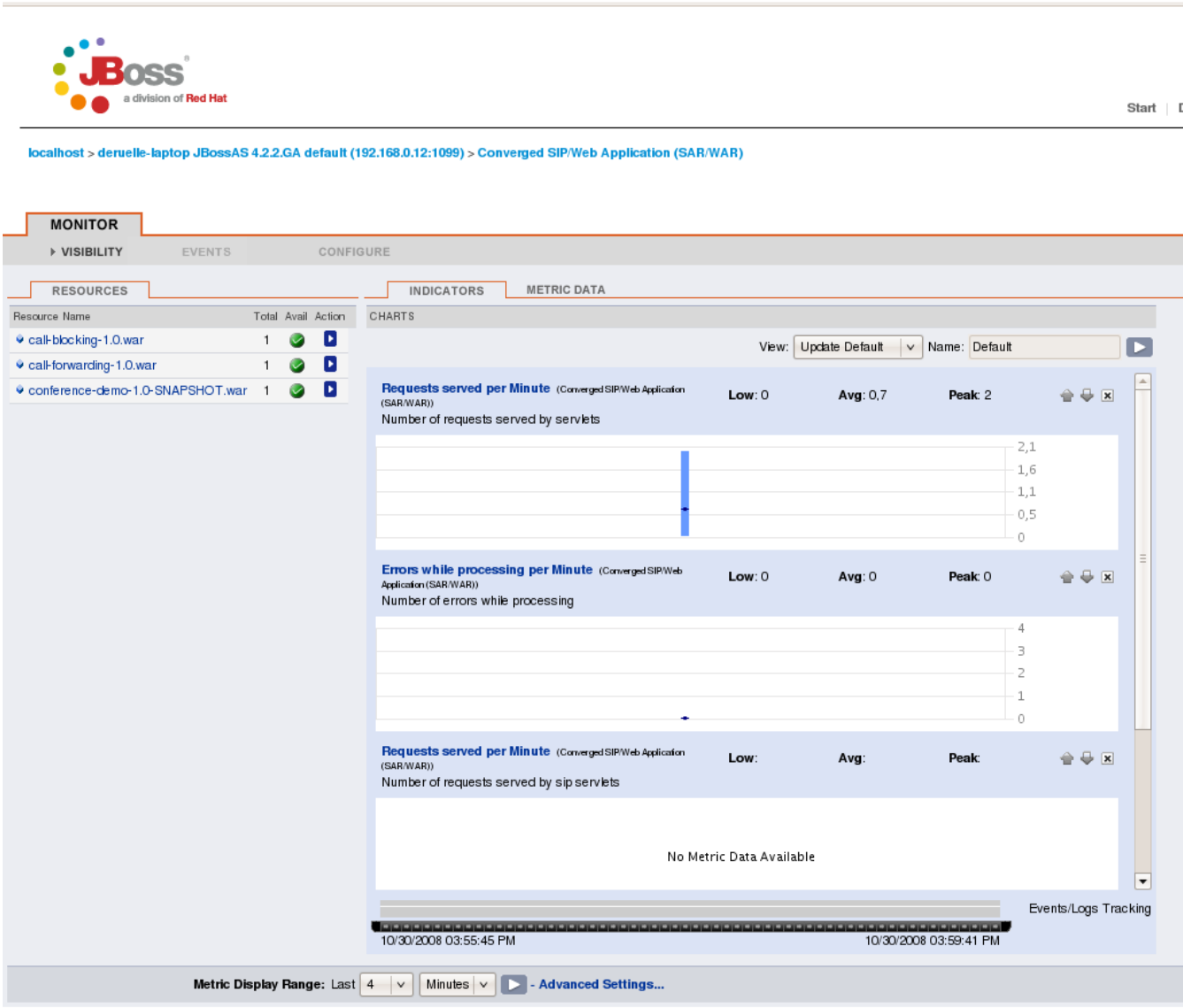


Figure 6.15. Specific Application Metrics Detail

- Click on the **Configure** link for the **Monitor** tab to select the metric data to view.

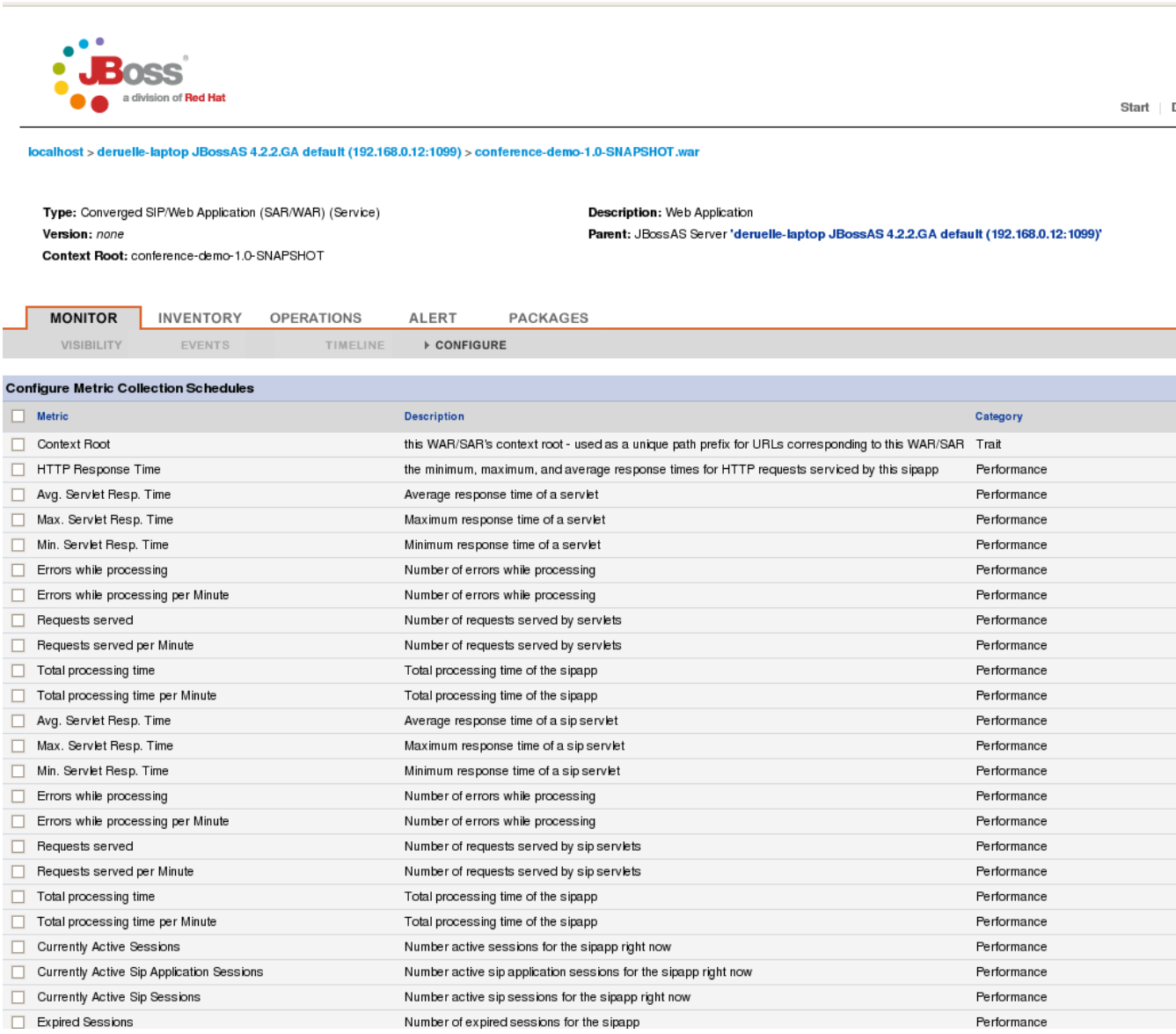


Figure 6.16. Selecting Application Metrics

- Select the **MetaData** tab to see the metrics of your application.

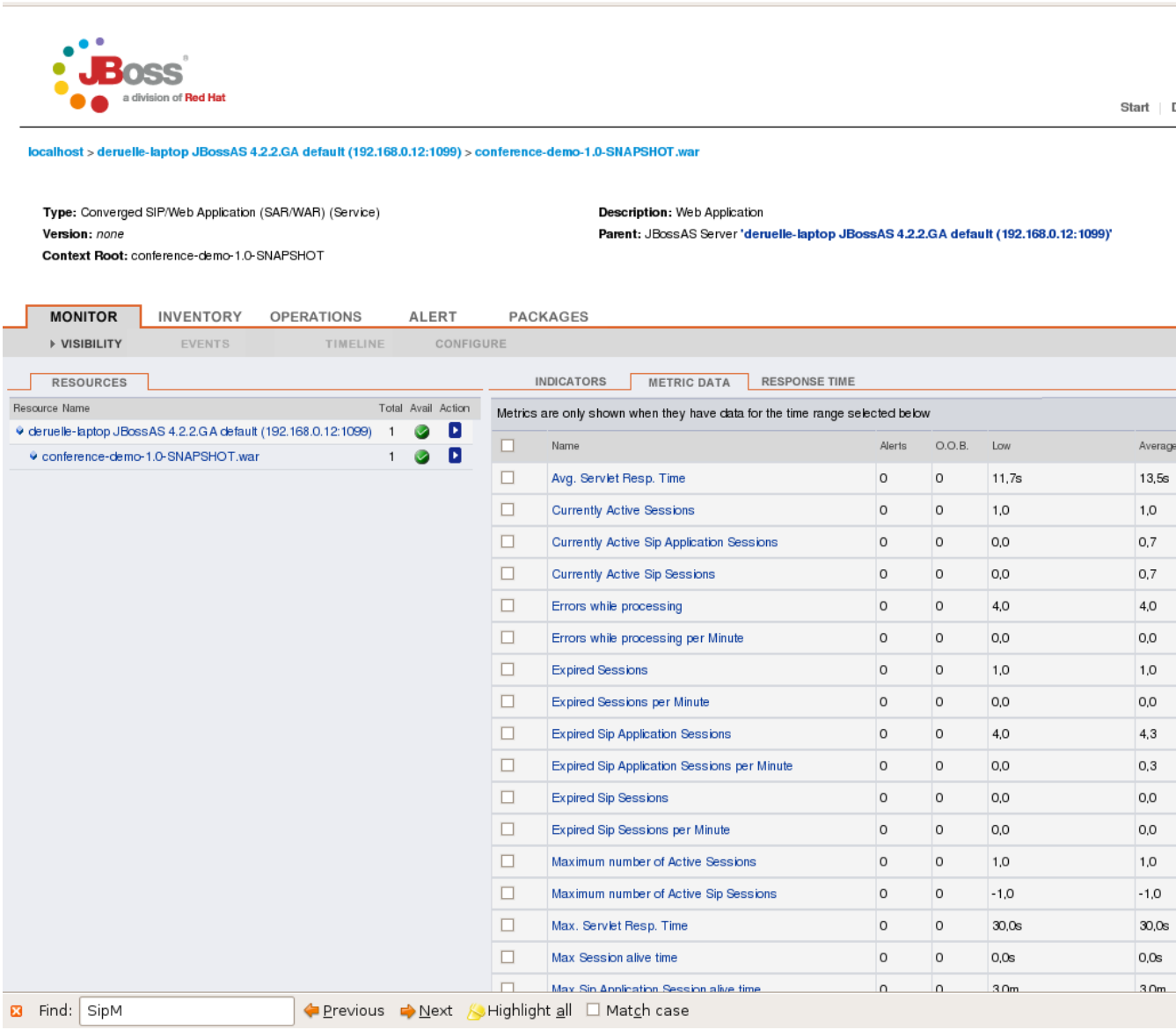


Figure 6.17. Selected Application Metrics Detail

6.3. SIP Load Balancer Jopr Monitoring and Management

This documentation provides information on how to enable the management of JBoss Communications SIP Load Balancer through Jopr with our custom SIP Load Balancer Jopr plug in.

With the JBoss Communications SIP Load Balancer Jopr plug in, you can currently see metrics, configure and manage the JBoss Communications SIP Load Balancer.

6.3.1. Installation of the Enterprise Monitoring and Management Console

- Follow the *Jopr installation instructions* [http://jboss.org/community/docs/DOC-12828] to install the latest version of Jopr.
- Stop the Jopr server and agent if they are running.
- Get the JBoss Communications SIP Load Balancer Jopr plugin from the JBoss maven repository at *http://repository.jboss.org/maven2/org/mobicents/tools/sip-balancer-jopr-plugin/1.0/sip-balancer-jopr-plugin-1.0.jar*
- Copy the JBoss Communications SIP Load Balancer Jopr plugin jar to the following directory:

  ```
  jopr-server/jbossas/server/default/deploy/rhq.ear/rhq-downloads/rhq-plugins/
  ```

- Start the Jopr server then the agent.
- Start your JBoss Communications SIP Load Balancer as explained here (and one node that connect to it to see it appear in the list of nodes).

6.3.2. Usage Instructions

- Log in to the Jopr console from the **Dashboard** tab, in the Auto Discovery portlet and import your JBoss Communications SIP Load Balancer (for example, JBoss Communications SIP Load Balancer 1.0-SNAPSHOT) from under localhost.

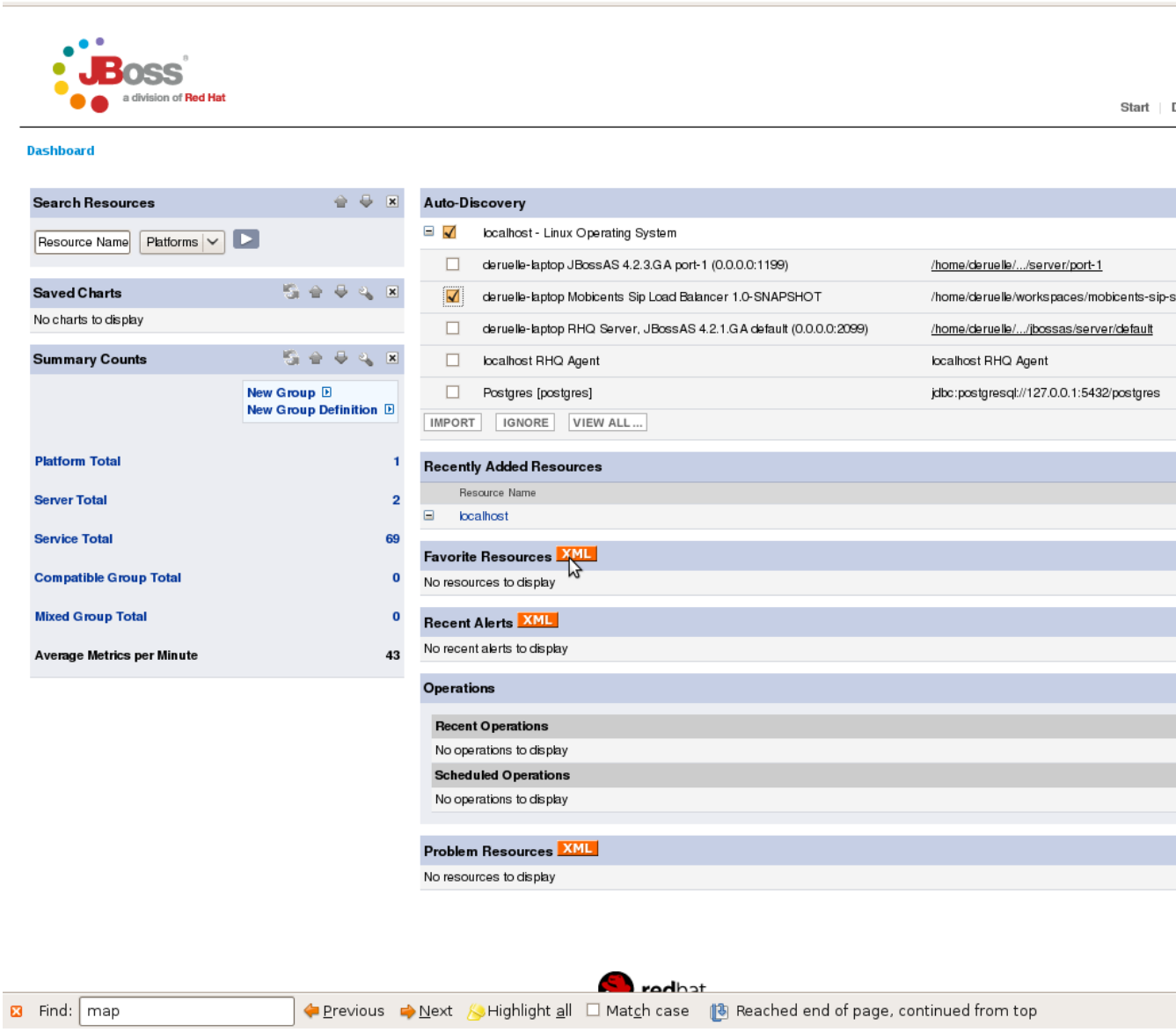


Figure 6.18.

- From the Dashboard, in the Recently Added Resources portlet, click on the Mobicents SIP Load Balancer.

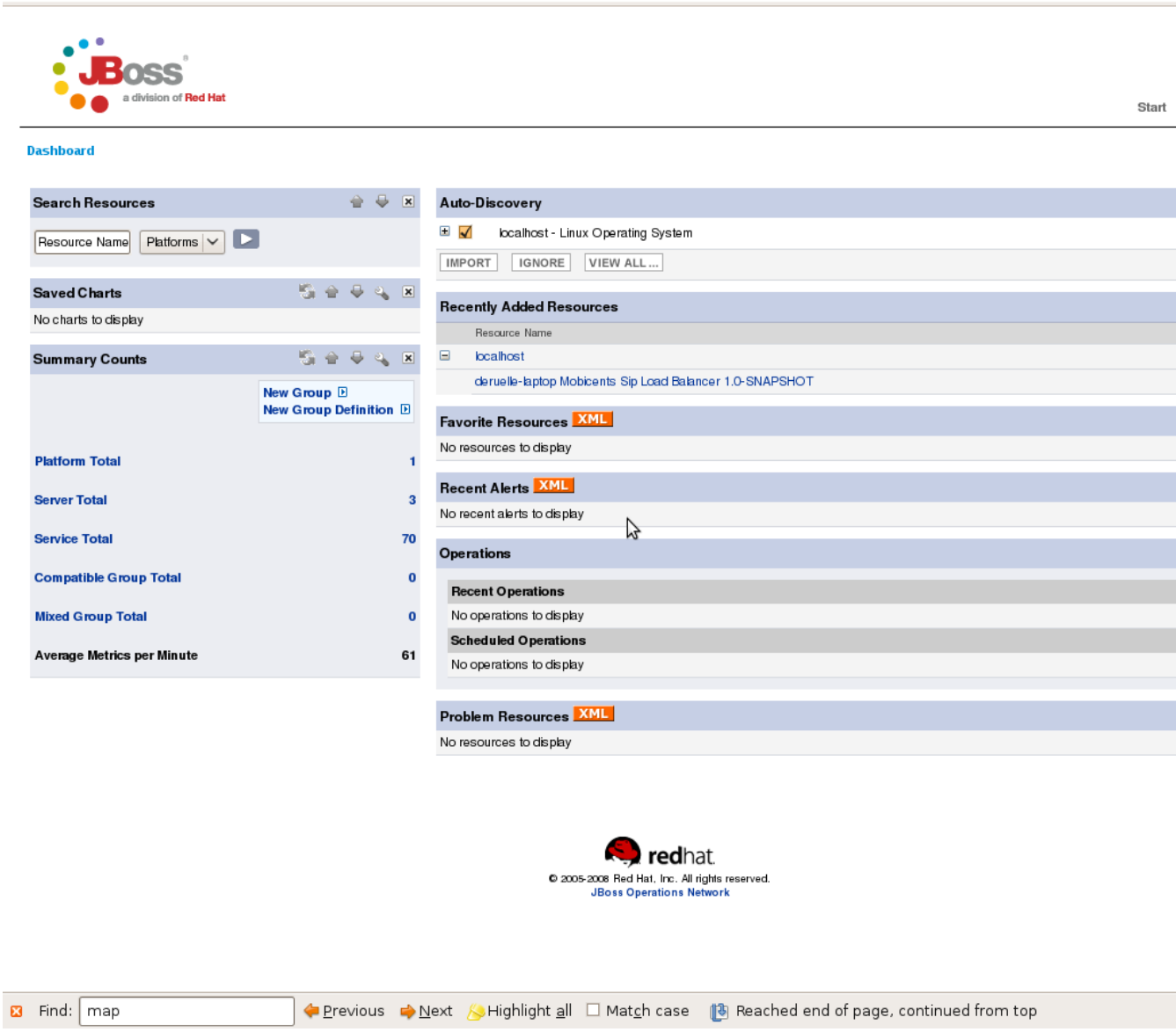
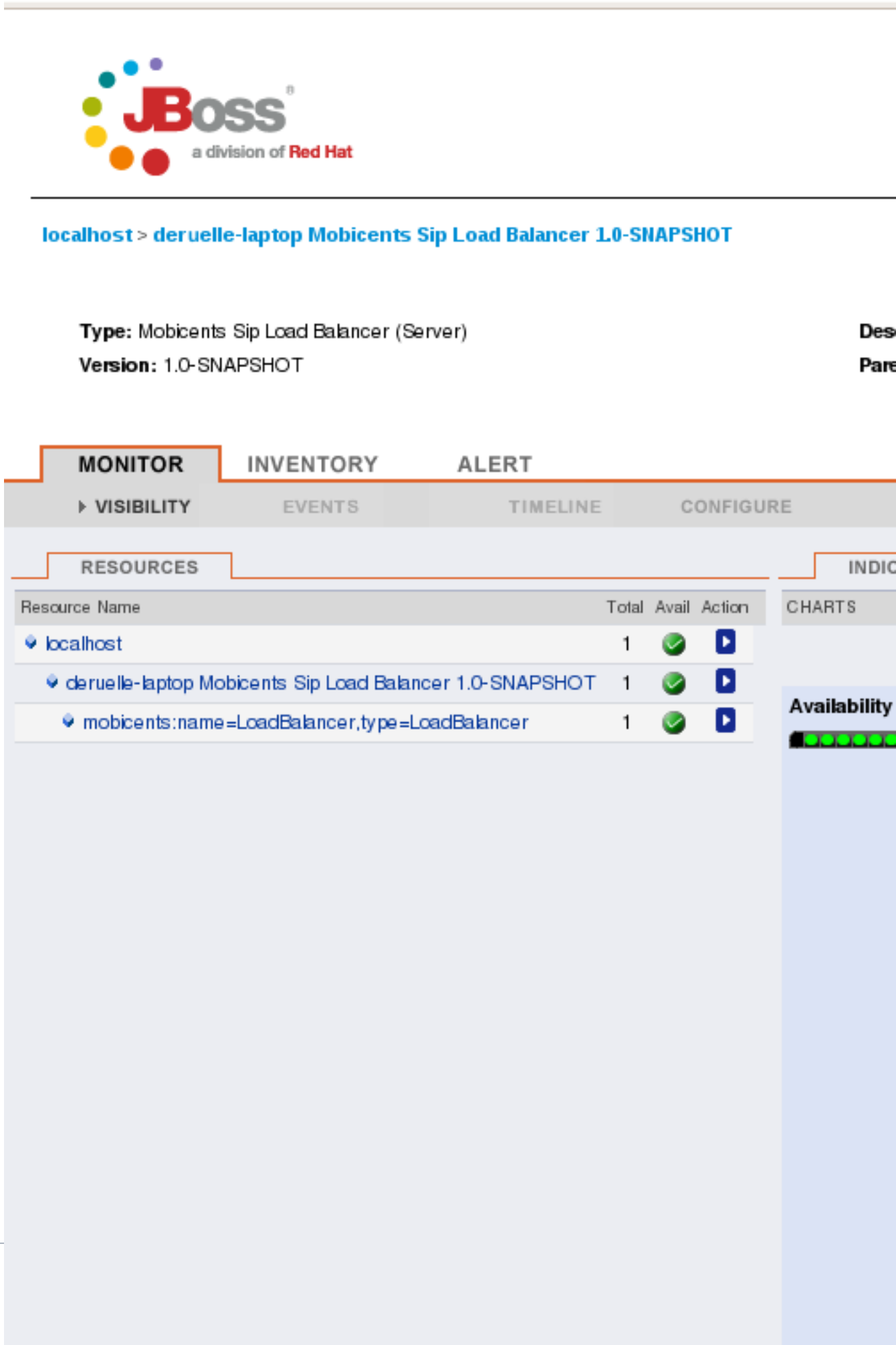
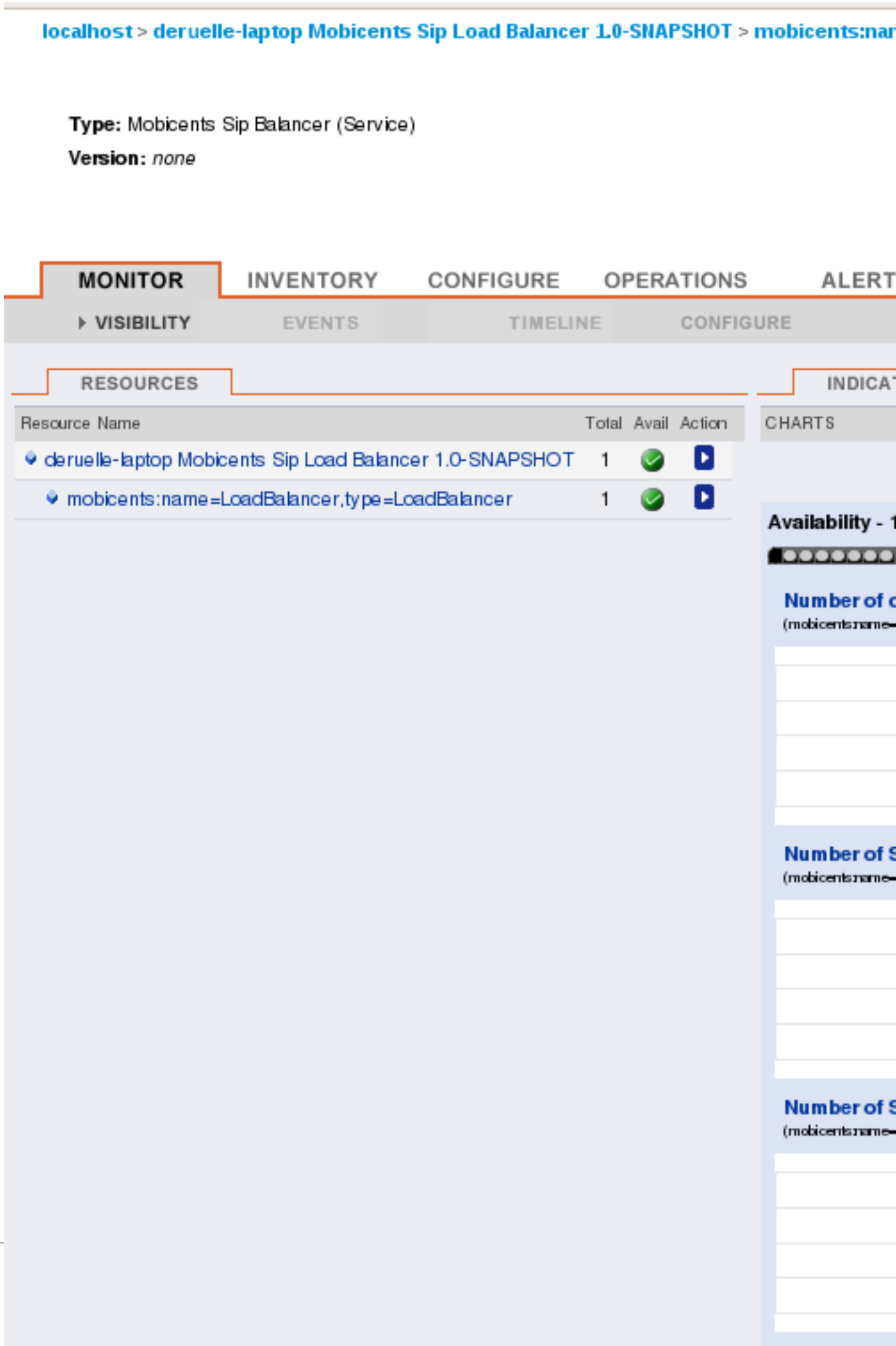


Figure 6.19.

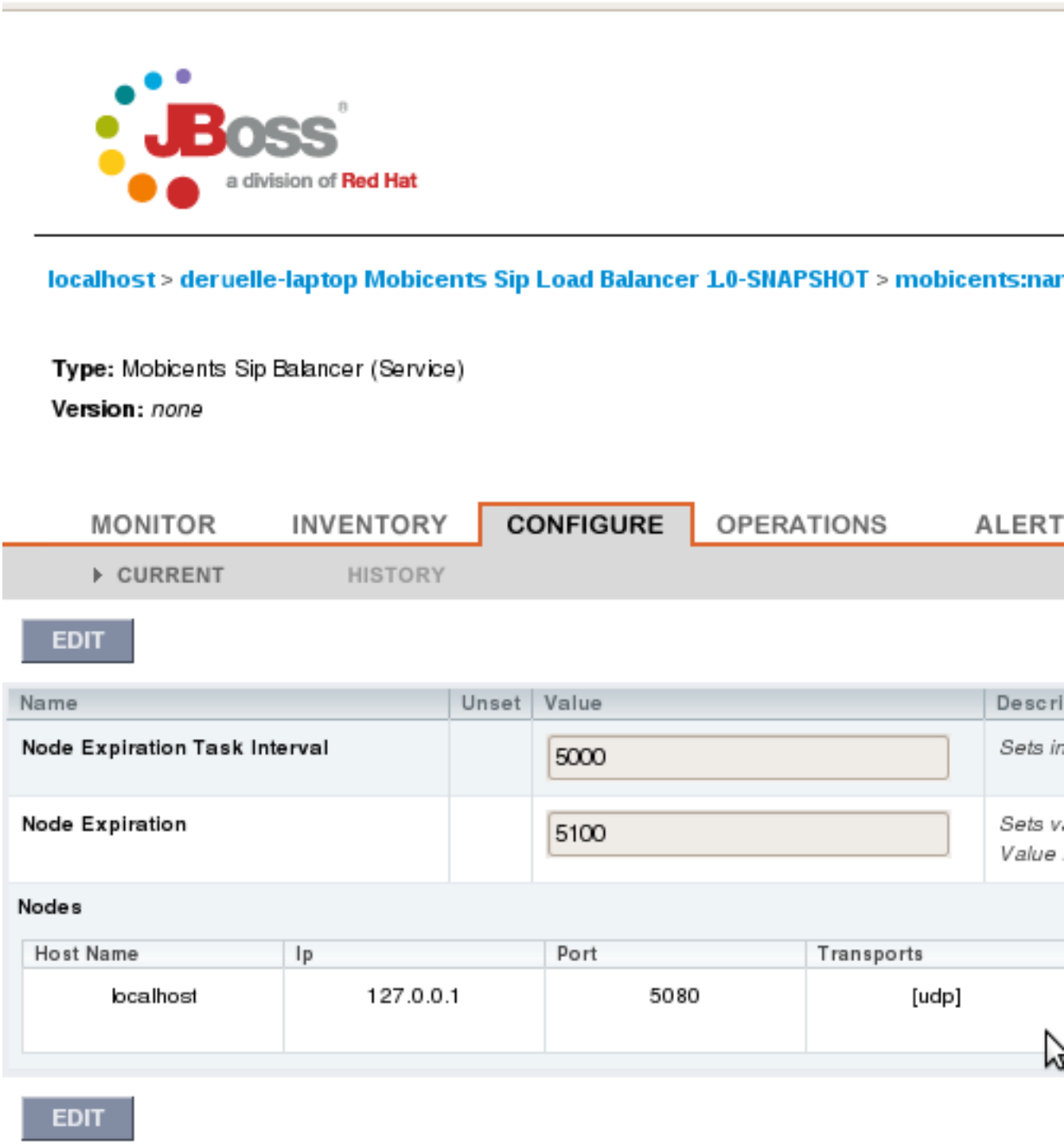
- In the new Monitor view, click on the **mobicents:name=LoadBalancer,type=LoadBalancer** link.


JBoss
a division of Red Hat
localhost > deruelle-laptop Mobicents Sip Load Balancer 1.0-SNAPSHOT
Type: Mobicents Sip Load Balancer (Server)
Version: 1.0-SNAPSHOT
MONITOR
INVENTORY
ALERT
VISIBILITY
EVENTS
TIMELINE
CONFIGURE
RESOURCES
INDIC
Resource Name
Total
Avail
Action
CHARTS
localhost
1
deruelle-laptop Mobicents Sip Load Balancer 1.0-SNAPSHOT
1
mobicents:name=LoadBalancer,type=LoadBalancer
1
Availability

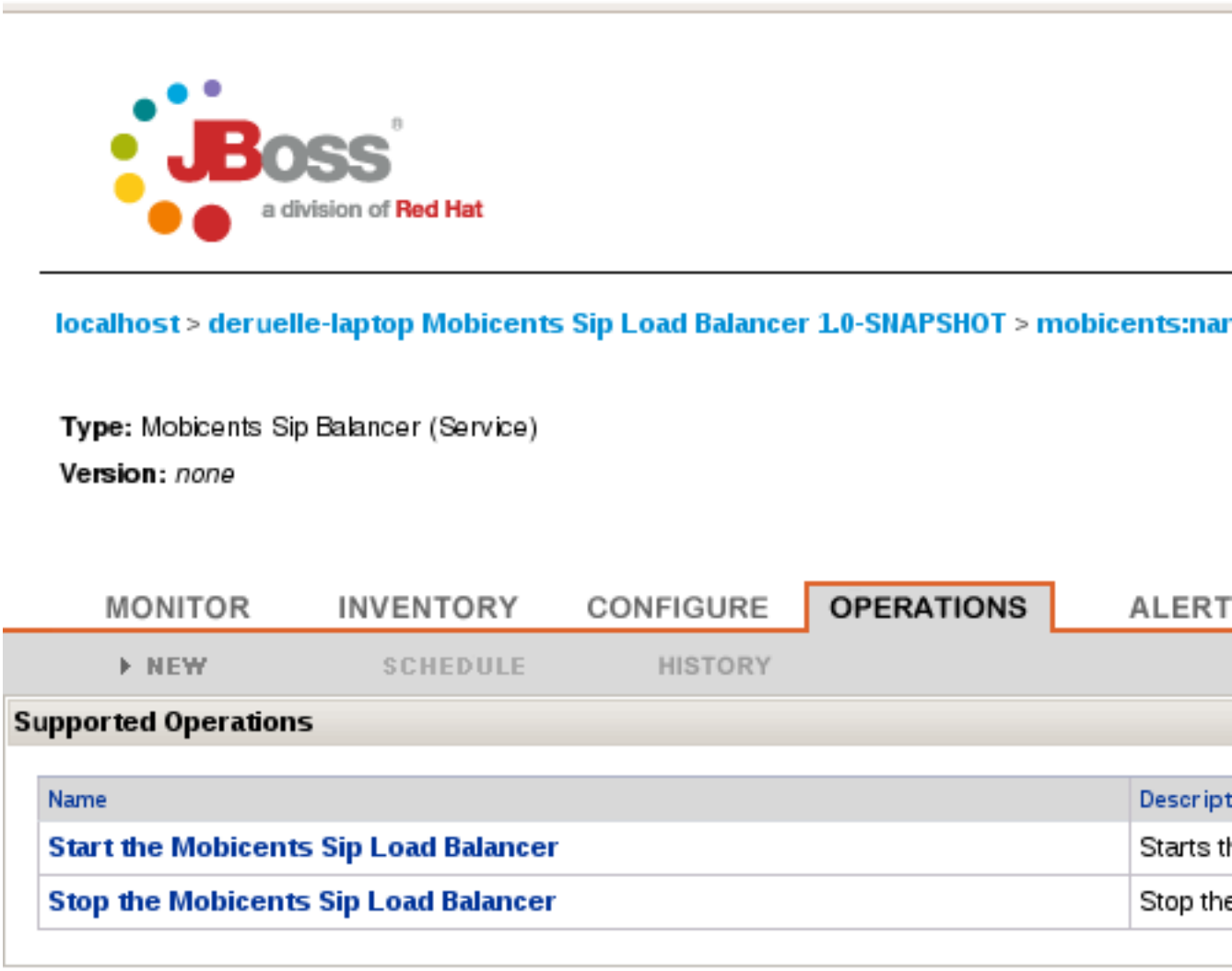
- You can now see the metrics coming in.


localhost > deruelle-laptop Mobicents Sip Load Balancer 1.0-SNAPSHOT > mobicents:nar
Type: Mobicents Sip Balancer (Service)
Version: none
MONITOR
INVENTORY
CONFIGURE
OPERATIONS
ALERT
VISIBILITY
EVENTS
TIMELINE
CONFIGURE
RESOURCES
INDICA
Resource Name
Total
Avail
Action
CHARTS
deruelle-laptop Mobicents Sip Load Balancer 1.0-SNAPSHOT
1
mobicents:name=LoadBalancer,type=LoadBalancer
1
Availability -
Number of
Number of
Number of

- To configure the Load Balancer and see the list of nodes in the cluster it is connected to, click on the **Configure** tab.


JBoss
a division of Red Hat
localhost > deruelle-laptop Mobicents Sip Load Balancer 1.0-SNAPSHOT > mobicents:nar
Type: Mobicents Sip Balancer (Service)
Version: none
MONITOR
INVENTORY
CONFIGURE
OPERATIONS
ALERT
CURRENT
HISTORY
EDIT
Name
Unset
Value
Descri
Node Expiration Task Interval
5000
Sets in
Node Expiration
5100
Sets v
Value
Nodes
Host Name
Ip
Port
Transports
localhost
127.0.0.1
5080
[udp]
EDIT

- To start and stop the Load Balancer, click on the **Operations** tab.


JBoss
a division of Red Hat
localhost > deruelle-laptop Mobicents Sip Load Balancer 1.0-SNAPSHOT > mobicents:nar
Type: Mobicents Sip Balancer (Service)
Version: none
MONITOR
INVENTORY
CONFIGURE
OPERATIONS
ALERT
NEW
SCHEDULE
HISTORY
Supported Operations
Name
Descript
Start the Mobicents Sip Load Balancer
Starts th
Stop the Mobicents Sip Load Balancer
Stop the

Advanced Features of the SIP Servlets Server

The advanced features of SIP Servlets include Concurrency and Congestion Control, load balancing with the JBoss Communications Load Balancer, and, exclusively for JBCP SIP Servlets for JBoss, clustering and failover support.

7.1. Media Support

JBoss Communications SIP Servlets provides support for applications to set up calls through SIP by implementing the SIP Servlets 1.1 Specification.

As most Telco services have the need for managing and controlling media (for example, to play announcements, mix calls and recognize DTMF), JBoss Communications SIP Servlets allows applications to control media through JSR 309.

7.1.1. JSR 309: Media Server Control API

This specification is a protocol agnostic API for Media Server Control. It provides a portable interface to create media rich applications with IVR, Conferencing, Speech Recognition, and similar features.

JBoss Communications Media Server provides an implementation of the *JSR 309 specification* [http://jcp.org/en/jsr/detail?id=309] using the MGCP protocol, to allow any Media Server (located in the same Virtual Machine or on a remote server) supporting MGCP to be controlled.

The following examples demonstrate its usage:

- *Media Example* [http://www.mobicents.org/mss-jsr309-demo.html] : a SIP Servlet application showing how to use media capabilities (Media playback, Recording, Text to Speech with FreeTTS and DTMF detection).
- *Conference Demo* [http://www.mobicents.org/conference-demo-jsr309.html] : a Conference Media Server demo application built on GWT with server-push updates.
- *Shopping Example* [http://www.mobicents.org/shopping-demo-jsr309.html] : a Converged JEE Application showing SEAM integration, JEE, Media integration with TTS and DTMF support.

7.2. Concurrency and Congestion Control

Concurrency and Congestion control refer to settings that define the way in which messages are processed under heavy load. The way JBoss Communications SIP Servlets Server processes messages in a production environment is crucial to ensure quality of service for customers.

Concurrency control mode tuning affects the way in which the SIP Servlets Server processes messages, whereas Congestion Control tuning affects the point at which the server begins rejecting new requests. Both of these parameters can be set using the following methods:

- Through the SIP Servlets Management Console.
- Editing the server's `server.xml` configuration file.
- From the `dispatcher` MBean.
- From the Embedded Jopr integrated management platform.

Concurrency Control. The JSR 289 expert group does not specify how concurrency control should be implemented.

JBoss Communications SIP Servlets for JBoss and JBoss Communications SIP Servlets for Tomcat have concurrency control implemented as a configurable mode, which defines the way in which the SIP Servlets Server processes messages.

It has to be noted that this concurrency control mechanism is not cluster aware and will work per node only, it is not a cluster wide lock.

The following modes are provided, and cater for the particular setup required in an implementation:

None
: All SIP messages are processed as soon as possible in a thread from the global thread pool.

  When two messages belong to the same `SipSession`, they can be concurrently processed. Ensure that SIP Messages that access shared resources (such as the session attribute) concurrently are synchronized in a thread-safe manner.

Transaction
: Bypass the SIP Servlets request/response executors, and utilize the JAIN SIP built-in Transaction serialization to manage race conditions on the same transaction.

  By default, the SIP Servlets server uses a ThreadPoolExecutor linked to a LinkedBlockingQueue to dispatch the request/response threads. The container can thus handles two different response (for example a 180 Ringing and a 200 OK) concurrently, a race condition can occur where the second response overtakes the first one (200 OK dispatched to the application before the 180 Ringing).

SipSession
: SIP messages are processed as soon as possible except for messages originating from the same `SipSession`. These messages are excluded from any simultaneous processing.

  Messages from the same `SipSession` are processed sequentially, in the order they originally arrived.

Two (or more) messages from different `SipSession` instances in the same `SipApplicationSession` may be processed simultaneously. For this reason, ensure that SIP Messages that access shared resources (such as the session attribute) concurrently are synchronized in a thread-safe manner.

Thread-safety is particularly important in Back-to-Back User Agent (B2BUA) cases, where each communication leg of the B2BUA consists of a different `SipSession` in the same `SipApplicationSession`.

SipApplicationSession
: SIP messages are processed as soon as possible, with the guarantee that no two messages from the same `SipSession` *or* from the same `SipApplicationSession` will ever be processed simultaneously. Of all the available methods, this mode is the best choice for guaranteed thread-safety.

  If applications access shared resources in an unmanaged way (for example, by accessing a `SipSession` attribute from an unmanaged thread, or from an Enterprise JavaBean) access will not be synchronized.

Congestion Control. JBoss Communications Sip Servlets currently provides the following congestion control mechanisms:

- Congestion control is largely application-specific and it is implemented in a pipeline. First messages arrive in the JAIN SIP message queue where they will wait on the locks needed before they can be processed. To avoid keeping too many messages in the queue and potentially running out of memory, older messages are discarded without any error indication. This prevents spam and flood DoS attacks to accumulate large backlog and render the server unresponsive. It also guranatees flood recovery time of 20 seconds or less, in the mean time retransmissions are already queuing so that normal SIP calls can continue without dropping them. After the request has passed the first queue it enters the SIP transaction layer where there is a customizable optional congestion control logic. There is one packaged congestion control algorithm which can be enabled by setting the following property `gov.nist.javax.sip.SIP_MESSAGE_VALVE=gov.nist.javax.sip.stack.CongestionControlMessageValve`. For this algorithm you can set the limit value by the following property `gov.nist.javax.sip.MAX_SERVER_TRANSACTIONS=2000`. You can also implement your own algorithm and change the class name in `gov.nist.javax.sip.SIP_MESSAGE_VALVE` to activate it.

  There is also another optional legacy congestion control stage with another queue where messages can be discarded based on dynamic parameters such as available JVM heap memory or number of messages in the queue. This method will be deprecated and is not recommended. All SIP messages which cannot be processed immediately are put into a queue, and wait for either a free thread or for the lock on their session to be released. The size of the SIP message queue is a tunable parameter, which defaults to `1500`.

- If the SIP Message queue becomes full, the container immediately begins rejecting new SIP requests until the queue clears. This is achieved by using one of the following methods:

  - Sending a `503` SIP error code to the originating application.
  - Dropping incoming messages (according to the specified congestion control policy).
- If the container exceeds the configurable memory threshold (90% by default), new SIP requests are rejected until the memory usage falls below the specified memory threshold. This is achieved by using one of the following methods:
  - Sending a `503` SIP error code to the originating application.
  - Dropping incoming messages (according to the specified congestion control policy).

A background task gathers information about the current server congestion. The data collection interval can be adjusted, and congestion control deactivated, by setting the interval to 0 or a negative value.

The congestion control policy defines how an incoming message is handled when the server is overloaded. The following parameters are configurable:

- DropMessage - drop any incoming message
- ErrorResponse - send a 503 - Service Unavailable response to any incoming request (Default).

Configuring the Concurrency and Congestion Control Settings. The concurrency and congestion control settings can be configured through the SIP Servlets Management Console, using the following methods:

- Through the SIP Servlets Management Console.
- Editing the server's `server.xml` configuration file.
- From the `dispatcher` MBean.
- From the Embedded Jopr integrated management platform.

Tuning Parameters with the SIP Servlets Management Console

The easiest way to configure the **SIP Message Queue Size** and **Concurrency Control Mode** tunable parameters is to open the `SIP Servlets Management Console` in your browser (by going to *http://localhost:8080/sip-servlets-management*), making your changes, and then clicking **Apply**.

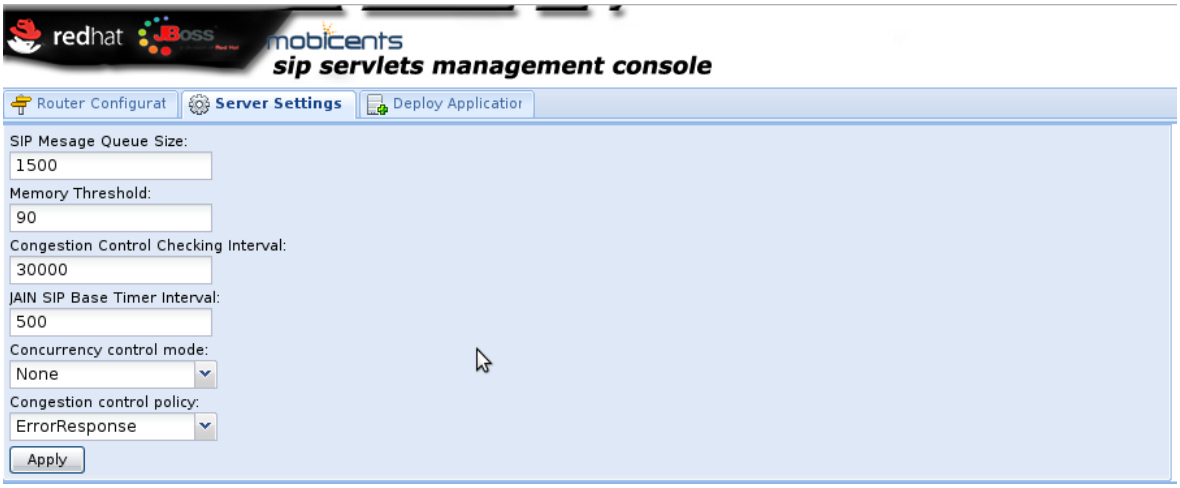


Figure 7.1. SIP Servlets Management Console Concurrency and Congestion Control Tuning Parameters

Persistent Settings

Concurrency and congestion control settings altered through the SIP Servlets Management Console are not saved to the `server.xml` configuration file.

To make settings persistent, append the settings to the `server.xml` file directly.

Tuning Parameters Permanently by Editing `server.xml`

Alternatively, you can edit your server's `server.xml` configuration file, which has the benefit of making your chosen settings changes permanent. Instructions follow, grouped by the SIP Servlets Server you are running:

Procedure 7.1. Tuning JBoss Communications SIP Servlets for JBoss Server Settings for Concurrency and Congestion Control

1. **Open server.xml File**

   Open the `$JBOSS_HOME/server/default/deploy/jboss-web.deployer/server.xml` configuration file in a text editor.

2. **Add Parameters to <service> Element**

   Locate the <service> element, and add the `concurrencyControlMode` and/or `sipMessageQueueSize` attributes.

Example 7.1. server.xml Concurrency and Congestion Control Attributes

```
<Service
 name="jboss.web"
 className="org.mobicents.servlet.sip.startup.SipStandardService"
 sipApplicationDispatcherClassName="org.mobicents.servlet.sip.core.SipApplicationDispatcherImpl"
 usePrettyEncoding="false"
 sipStackPropertiesFile="conf/mss-sip-stack.properties"
 darConfigurationFileLocation="conf/dars/mobicents-dar.properties"
 concurrencyControlMode="SipApplicationSession"
 sipMessageQueueSize="1600"
 backToNormalSipMessageQueueSize="1300"
 congestionControlCheckingInterval="2000"
 congestionControlPolicy="DropMessage"
 memoryThreshold="95"
 backToNormalMemoryThreshold="90">
```

Possible values for the `concurrencyControlMode` attribute include: *`None`*, *`SipSession`* or *`SipApplicationSession`*. *`SipSession`* is the value of this attribute when it is not present —and overridden—in `server.xml`.

3. **Define the Correct Attribute Values**

   The following default values for the concurrency and congestion control parameters are used regardless of whether the attributes are defined in the `server.xml` file:

   - sipMessageQueueSize="1500"
   - backToNormalSipMessageQueueSize="1300"
   - congestionControlCheckingInterval="30000" (30 seconds, in milliseconds)
   - memoryThreshold="95" (in percentage)
   - backToNormalMemoryThreshold="90" (in percentage)
   - congestionControlPolicy="ErrorResponse"

   Experimentation is required for these tuning parameters depending on the operating system and server.

Procedure 7.2. Tuning JBoss Communications SIP Servlets for Tomcat Server Settings for Concurrency and Congestion Control

1. **Open server.xml File**

   Open the `$CATALINA_HOME/conf/server.xml` configuration file in your text editor.

2. **Add Parameters to <service> Element**

   Locate the <service> element, and add the `concurrencyControlMode` and/or `sipMessageQueueSize` attributes.

   Possible values for the `concurrencyControlMode` attribute include: *`None`*, *`SipSession`* or *`SipApplicationSession`*. *`SipSession`* is the value of this attribute when it is not present —and overridden—in `server.xml`.

3. **Define the Correct Attribute Values**

   The following default values for the concurrency and congestion control parameters are used regardless of whether the attributes are defined in the `server.xml` file:

   - sipMessageQueueSize="1500"
   - backToNormalSipMessageQueueSize="1300"
   - congestionControlCheckingInterval="30000" (30 seconds, in milliseconds)
   - memoryThreshold="95" (in percentage)
   - backToNormalMemoryThreshold="90" (in percentage)
   - congestionControlPolicy="ErrorResponse"

   Experimentation is required for these tuning parameters depending on the operating system and server.

Tuning Parameters from the dispatcher MBean

Navigate to the `dispatcher` MBean from JBoss Communications SIP Servlets for JBoss's JMX console.

All changes performed at run time are effective immediately, but do not persist across reboots. As with JBoss and Tomcat, the `server.xml` must be appended with the settings in order to make the configuration persistent.

When editing the `dispatcher` MBean from JBoss Communications SIP Servlets for JBoss's JMX console, values allowed for the concurrency control mode are **`None`**, **`SipSession`** or **`SipApplicationSession`**.

Tuning Parameters from Enterprise Monitoring and Managent Console

If the Enterprise Monitoring and Managenemt console is installed as described in *Chapter 6, Enterprise Monitoring and Management*, the tunable parameters can be altered by following

the instructions in *Section 6.2.2.1, "Jopr for Development"* or *Section 6.2.2.2, "Jopr For Production"*

7.3. SIP Servlets Application Security

The information present in SIP requests often contains sensitive user information. To protect user information, SIP Security can be enabled on the server, and within the SIP application to mitigate the risk of unauthorised access to the information.

Application security varies depending on the server type used. The following procedures describe how to configure the JBoss Application Server, and the Tomcat server.

Procedure 7.3. Enable SIP Application Security in JBoss Application Server

1. **Add Security Policy to Server**

   1. Open a terminal and navigate to the conf directory:

```
home]$ cd server/default/conf/
```

   2. Open `login-config.xml` (using your preferred editor) and append the security policy to the file:

```
<application-policy name="sip-servlets">
<authentication>
  <login-module code="org.jboss.security.auth.spi.UsersRolesLoginModule"
    flag = "required">
  <module-option name="usersProperties">props/sip-servlets-
    users.properties</module-option>
  <module-option name="rolesProperties">props/sip-servlets-
    roles.properties</module-option>
  <module-option name="hashAlgorithm">MD5</module-option>
  <module-option name="hashEncoding">rfc2617</module-option>
  <module-option name="hashUserPassword">false</module-option>
  <module-option name="hashStorePassword">true</module-option>
  <module-option name="passwordIsA1Hash">true</module-option>
  <module-option name="storeDigestCallback">
    org.jboss.security.auth.spi.RFC2617Digest</module-option>
  </login-module>
</authentication>
```

```
</application-policy>
```

2. **Update SIP Server User Properties File**

   1. Open a terminal and navigate to the /props directory:

   ```
   home]$ cd server/default/props/
   ```

   2. Open `sip-servlets-users.properties` and append the user lines to the file:

   ```
   # A sample users.properties file, this line creates user "admin" with
   # password "admin" for "sip-servlets-realm"
   admin=<A1_cryptographic_string>
   ```

   3. To create <A1_cryptographic_string>, execute the following command in a terminal:

   ```
   home]$ java -cp ../server/default/lib/jbosssx.jar
   org.jboss.security.auth.spi.RFC2617Digest admin sip-servlets-realm <password>
   ```

   4. Copy the A1 hash, and paste it into the admin parameter in the previous step.

   5. Save and close `sip-servlets-users.properties`.

3. **Update the SIP Server Roles File**

   1. Open a terminal and navigate to the /props directory:

   ```
   home]$ cd server/default/props/
   ```

   2. Open `sip-servlets-roles.properties` (using your preferred editor) and append the following information to the file:

   ```
   # A sample roles.properties file for use with some roles
   # Each line in this file assigns roles to the users defined in
   # sip-servlets-users.properties
   ```

```
admin=caller,role1,role2,..
```

4. **Add the Security Domain to the SIP Application**

   1. Open the `jboss-web.xml` file for the SIP application to which security is required.

   2. Add the <security-domain> element as a child of the <jboss-web> element:

```
<jboss-web >
<!--Uncomment the security-domain to enable security. You will need to edit the
 htmladaptor-->
<!--login configuration to setup the login modules used to authentication users.-->
  <security-domain>java:/jaas/sip-servlets</security-domain>
</jboss-web >
```

5. **Add Security Constraints to the SIP Application**

   1. Open the `sip.xml` file for the SIP application.

   2. Add the <security-domain> element as a child of the <jboss-web> element:

```
<security-constraint>
  <display-name>REGISTER Method Security Constraint</display-name>
  <resource-collection>
    <resource-name>SimpleSipServlet</resource-name>
    <description>Require authenticated REGISTER requests</description>
    <servlet-name>SimpleSipServlet</servlet-name>
    <sip-method>REGISTER</sip-method>
  </resource-collection>
  <auth-constraint>
    <role-name>caller</role-name>
  </auth-constraint>
</security-constraint>
<login-config>
  <auth-method>DIGEST</auth-method>
  <realm-name>sip-servlets-realm</realm-name>
</login-config>
```

Procedure 7.4. Enable SIP Application Security in Tomcat Server

1. **Activate the Memory Realm in Catalina:**

    1. Open a terminal and navigate to the `/conf` directory:

    ```
    home]$ cd server/default/<tomcat_home>/conf/
    ```

    2. Open `server.xml` and uncomment the following line:

    ```
    <!--<Realm className="org.apache.catalina.realm.MemoryRealm"/>-->
    ```

2. **Update SIP Server User Properties File**

    1. In the `/conf` directory, open `tomcat-users.xml` (using your preferred editor) and append the following <user> child element:

    ```
    <user name="user" password="password" roles="caller"/>
    ```

3. **Add Security Constraints to the SIP Application**

    1. Open the `sip.xml` file for the SIP application to which security is required.

    2. Add the <security-domain> child element to the <jboss-web> element:

    ```
    <security-constraint>
      <display-name>REGISTER Method Security Constraint</display-name>
      <resource-collection>
        <resource-name>SimpleSipServlet</resource-name>
        <description>Require authenticated REGISTER requests</description>
        <servlet-name>SimpleSipServlet</servlet-name>
        <sip-method>REGISTER</sip-method>
      </resource-collection>
      <auth-constraint>
        <role-name>caller</role-name>
      </auth-constraint>
    </security-constraint>
    <login-config>
    ```

```
  <auth-method>DIGEST</auth-method>
  <realm-name>sip-servlets-realm</realm-name>
</login-config>
```

7.4. STUN Support

The Session Traversal Utilities for NAT (STUN) prococol is used in Network Address Translation (NAT) traversal for real-time voice, video, messaging, and related interactive IP application communications. This light-weight, client-server protocol allows applications passing through a NAT to obtain the public IP address for the UDP connections the application uses to connect to remote hosts.

STUN support is provided at the SIP connector level, using the *STUN for Java* [https://stun4j.dev.java.net/] project. The STUN for Java project provides a Java implementation of the STUN Protocol (RFC 3489), which allows each SIP connector to select whether it should use STUN to discover a public IP address, and then use this address in the SIP messages sent through the connector.

To make a SIP connector STUN-enabled, three attributes must be appended to the <connector> child element in the `server.xml` file. The properties are:

- useStun="true"

  Enables STUN support for this connector. Ensure that the `ipAddress` attribute is not set to `127.0.0.1`.

- stunServerAddress="<Public_STUN_Server>"

  STUN server address used to discover the public IP address of this SIP Connector. See *Table 7.1, "Public STUN Servers"* for a suggested list of public STUN servers.

- stunServerPort="3478"

  STUN server port of the STUN server used in the `stunServerAddress` attribute. Both TCP and UDP protocols communicate with STUN servers using this port only.

Note

A complete list of available SIP connector attributes and their descriptions is located in the *Section 2.3.1, "Configuring SIP Connectors"* section of this guide.

A number of public STUN servers are available, and can be specified in the `stunServerAddress`. Depending on the router firmware used, the STUN reply packets' MAPPED_ADDRESS may be changed to the router's WAN port. To alleviate this problem, certain public STUN servers provide

XOR_MAPPED_ADDRESS support. *Table 7.1, "Public STUN Servers"* provides a selection of public STUN servers.

Table 7.1. Public STUN Servers

Server Address	XOR Support	DNS SRV Record
stun.ekiga.net	Yes	Yes
stun.fwdnet.net	No	Yes
stun.ideasip.com	No	Yes
stun01.sipphone.com	Yes	No
stun.softjoys.com	No	No
stun.voipbuster.com	No	No
stun.voxgratia.org	No	No
stun.xten.com	Yes	Yes
stunserver.org	Yes	Yes

Note

For more information about NAT traversal best practices, refer to *Section 8.2, "NAT Traversal"*.

7.5. Mobicents vendor-specific Extensions to JSR 289

Mobicents provide Extensions for applications or external systems to interact with the Mobicents SIP Servlets container as well as Extensions not defined in the specification in the JSR 289 specification that can prove useful and might be proposed for inclusion in a next release of the SIP Servlets specification

Javadoc for Mobicents JSR 289 Extensions [http://ci.jboss.org/jenkins/view/Mobicents/job/MobicentsBooks/lastSuccessfulBuild/artifact/api-docs/jsr289/extensions/index.html]

7.6. CDI Telco Framework

CDI is the Java standard for dependency injection and contextual lifecycle management, led by Gavin King for Red Hat, Inc. and is a Java Community Process(JCP) specification that integrates cleanly with the Java EE platform. Any Java EE 6-compliant application server provides support for JSR-299 (even the web profile). It seemed a natural fit create a new framework based on CDI for the Telco world.

CDI-Telco-Framework (CTF) from Mobicents brings the power and productivity benefits of CDI into the Mobicents Sip Servlets platform providing dependency injection and contextual lifecycle management for converged HTTP/SIP applications. This new framework is intended to become a replacement for our previous Seam Telco Framework.

CTF mission statement is to simplify SipServlets development by introducing a component based programming model, ease of development by making available SIP utilities out of the box, and finally providing dependency injection and contextual lifecycle management to the SipServlets.

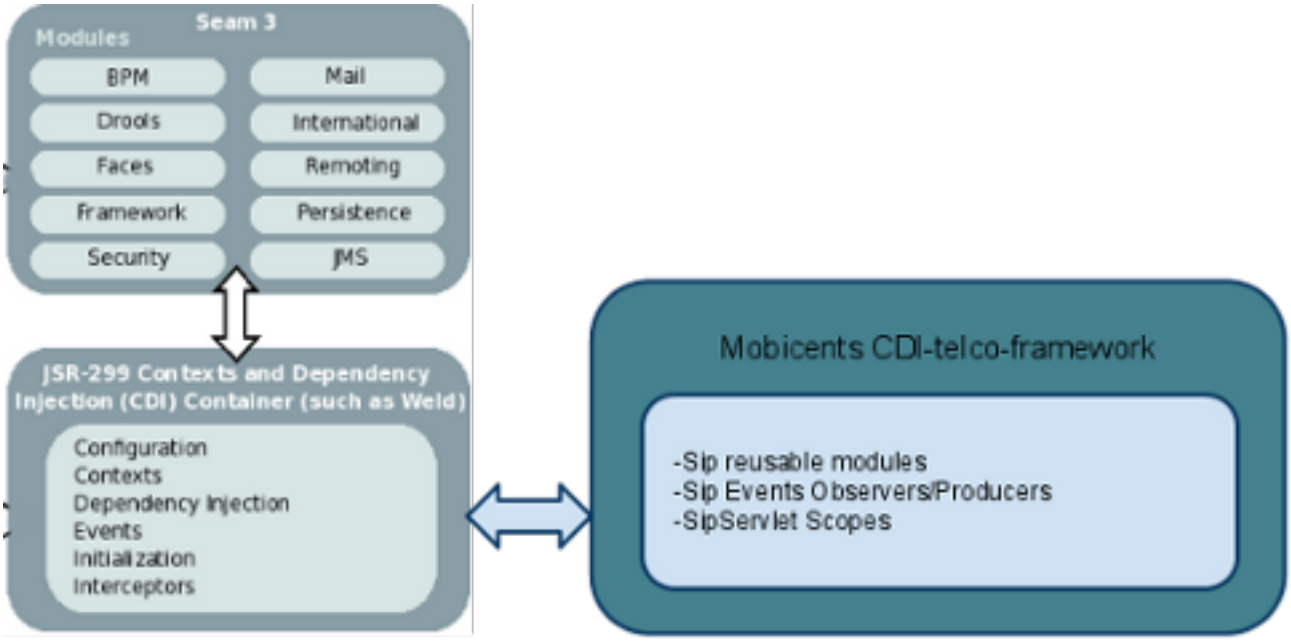


Figure 7.2. CDI Telco Framework Extension

More information about the CTF can be found on the *CDI Telco Framework Documentation* [http://docs.jboss.org/mobicents/frameworks/ctf/1.0.0.ALPHA1/user_guide/en-US/html_single/].

7.7. Diameter Support

The Diameter Protocol (*RFC 3588* [http://www.ietf.org/rfc/rfc3588.txt]) is a computer networking protocol for Authentication, Authorization, and Accounting (AAA). The Diameter version included in JBoss Communications SIP Servlets currently support Base, Sh, Ro and Rf.

For more information regarding Diameter support, refer to the *Diameter Home Page* [http://groups.google.com/group/mobicents-public/web/mobicents-diameter]. For a list of Diameter examples, refer to *Chapter 4, SIP Servlet Example Applications*.

7.8. SIP and IMS Extensions

SIP Extensions in the SIP Servlets Server are based on the Internet Engineering Task Force's (IETF) Request for Comments (RFC) protocol recommendations. *Table 7.2, "Supported SIP Extensions"* lists the supported RFCs for the SIP Servlets Server.

Table 7.2. Supported SIP Extensions

Extension	RFC Number	Description
DNS	*RFC 3263* [http://www.ietf.org/rfc/rfc3263.txt]	SIP: Locating SIP Servers

Extension	RFC Number	Description
ENUM	*RFC 2916* [http://www.ietf.org/rfc/rfc2916.txt]	E.164 number and DNS
INFO	*RFC 2976* [http://www.ietf.org/rfc/rfc2976.txt]	The SIP INFO Method
IPv6	*RFC 2460* [http://www.ietf.org/rfc/rfc2460.txt]	Internet Protocol, Version 6 (IPv6) Specification
JOIN	*RFC 3911* [http://www.ietf.org/rfc/rfc3911.txt]	The SIP "Join" Header
MESSAGE	*RFC 3428* [http://www.ietf.org/rfc/rfc3428.txt]	SIP Extension for Instant Messaging
PATH	*RFC 3327* [http://www.ietf.org/rfc/rfc3327.txt]	SIP Extension Header Field for Registering Non-Adjacent Contacts
PRACK	*RFC 3262* [http://www.ietf.org/rfc/rfc3262.txt]	Reliability of Provisional Responses in the SIP
PUBLISH	*RFC 3903* [http://www.ietf.org/rfc/rfc3903.txt]	SIP Extension for Event State Publication
REASON	*RFC 3326* [http://www.ietf.org/rfc/rfc3326.txt]	The Reason Header Field for the Session Initiation Protocol (SIP)
REFER	*RFC 3515* [http://www.ietf.org/rfc/rfc3515.txt]	The SIP Refer Method
REPLACES	*RFC 3891* [http://www.ietf.org/rfc/rfc3891.txt]	The SIP "Replaces" Header
STUN	*RFC 3489* [http://www.ietf.org/rfc/rfc3489.txt]	STUN - Simple Traversal of User Datagram Protocol (UDP) through Network Address Translators (NATs)
SUBSCRIBE/NOTIFY	*RFC 3265* [http://www.ietf.org/rfc/rfc3265.txt]	SIP-specific Event Notification
Symmetric Response Routing	*RFC 3581* [http://www.ietf.org/rfc/rfc3581.txt]	An Extension to the Session Initiation Protocol (SIP) for Symmetric Response Routing
Multipart type	*RFC 4662* [http://www.ietf.org/rfc/rfc4662.txt]	A Session Initiation Protocol (SIP) Event Notification

IMS Private Header (P-Header) Extensions are provided according to the recommendations of the *3rd Generation Partnering Project (3GPP)* [http://www.3gpp.org/] and the IETF. P-Header

extensions are primarily used to store information about the networks a call traverses, including security or call charging details.

Table 7.3, "IMS P-Header Extensions" describes the list of supported P-Headers, including links to the relevant ITEF memorandum where available.

Table 7.3. IMS P-Header Extensions

Extension	Description
AuthorizationHeaderIMS	Defines a new auth-param for the Authorization header used in REGISTER requests.
PAccessNetworkInfoHeader	Contains information regarding the access network the User Agent (UA) uses to connect to the SIP Proxy. The information contained in this header may be sensitive, such as the cell ID, so it is important to secure all SIP application that interface with this header.
PAssertedIdentityHeader [http://www.ietf.org/rfc/rfc3324.txt]	Contains an identity resulting from an authentication process, derived from a SIP network intermediary. The identity may be based on SIP Digest authentication.
PAssertedServiceHeader	Contains information used by "trust domains", according to Spec(T) specifications detailed in *RFC 3324* [http://www.ietf.org/rfc/rfc3324.txt].
PAssociatedURIHeader [http://www.ietf.org/rfc/rfc3455.txt]	Contains a list of URIs that are allocated to the user. The header is defined in the 200 OK response to a REGISTER request. It allows the User Agent Client (UAC) to determine the URIs the service provider has associated to the user's address-of-record URI.
PathHeader [http://www.ietf.org/rfc/rfc3327.txt]	SIP Extension header, with syntax similar to the RecordRoute header. Used in conjunction with SIP REGISTER requests and 200 class messages in response to REGISTER responses.
PCalledPartyIDHeader	Typically inserted en-route into an INVITE request by the proxy, the header is populated with the Request_URI received by the proxy in the request. The header allows the User Agent Server (UAS) to identify which address-of-record the invitation was sent to, and can

Extension	Description
	be used to render distinctive audio-visual alert notes based on the URI.
PChargingFunctionAddressesHeader	Contains a list of one or more of the Charging Collection Function (CCF) and the Event Charging Function (ECF) addresses. The CCF and ECF addresses may be passed during the establishment of a dialog, or in a standalone transaction.
PChargingVectorHeader	Contains a unique charging identifier and correlation information, which is used by network operators to correctly charge for routing events through their networks.
PMediaAuthorizationHeader [http://www.ietf.org/rfc/rfc3313.txt]	Contains one or more session-specific media authorization tokens, which are used for QoS of the media streams.
PPreferredIdentityHeader [http://www.ietf.org/rfc/rfc3325.txt]	Contains a SIP URI and an optional display-name. For example, "James May" <sip:james@domain.com>. This header is used by trusted proxy servers to identify the user to other trusted proxies, and can be used to select the correct SIP URI in the case of multiple user identities.
PPreferredServiceHeader	Used by the PAssertedService Header to determine the preferred user service. Multiple PPreferreedService headers may be present in a single request.
PProfileKeyHeader [http://www.ietf.org/rfc/rfc5002.txt]	Contains a key used by a proxy to query the user database for a given profile. The key may contain wildcards that are used as part of the query into the database.
PrivacyHeader [http://www.ietf.org/rfc/rfc3323.txt]	Contains values that determine whether particular header information is deemed as private by the UA for requests and responses.
PServedUserHeader [http://www.ietf.org/rfc/rfc5502.txt]	Contains an identity of the user that represents the served user. The header is added to the initial requests for a dialog or standalone request, which are then routed between nodes in a trusted domain.
PUserDatabaseHeader [http://www.ietf.org/rfc/rfc4457.txt]	Contains the address of the HSS handling the user that generated the request. The header field is added to request routed from an

Extension	Description
	Interrogating Call Session Control Function (I-CSCF) to a Serving Call Session Control Function (S-CSCF).
PVisitedNetworkIDHeader [http://www.ietf.org/rfc/rfc3455.txt]	Contains the identifier of a visited network. The identifier is a text string or token than it known by both the registrar or the home proxy at the home network, and the proxies in the visited network.
SecurityClientHeader, SecurityServerHeader, SecurityVerifyHeader [http://www.ietf.org/rfc/rfc3329.txt]	Contains information used to negotiate the security mechanisms between a UAC, and other SIP entities including UAS, proxy and registrar.
ServiceRouteHeader [http://www.ietf.org/rfc/rfc3608.txt]	Contains a route vector that will direct requests through a specified sequence of proxies. The header may be included by a registrar in response to a REGISTER request.
WWWAuthenticateHeaderIms [http://snad.ncsl.nist.gov/proj/iptel/jain-sip-1.2/javadoc/gov/nist/javax/sip/header/ims/WWWAuthenticateHeaderIms.html]	Extends the WWWAuthenticateResponse header functionality by defining an additional authorization parameter (auth-param).

7.9. JRuby/Rails Integration with Torquebox Telco Framework

JBoss Communications Sip Servlets is compatible with the *Torquebox* [http://torquebox.org/] Telco Framework JRuby on Rails integration. The framework allows you to create powerful, pure or converged VoIP JRuby on Rails applications.

JRuby features a powerful and well deployed scripting language that allows you to modify your application at runtime (this is true even for the SIP and Media part that JBoss Communications SIP Servlets offer) without restarting the server. In addition, TorqueBox is a new kind of Ruby application platform that integrates popular technologies such as Ruby-on-Rails, while extending the footprint of Ruby applications to include support for Job Scheduling, Task Queues, SOAP Handling, and other capabilities.

To obtain more information about building pure JRuby-Rails applications that leverage JBoss Communications SIP Servlets SIP and media capabilities, refer to the *Torquebox User Documentation* [http://torquebox.org/documentation/browse/LATEST/telecom.html].

Check this *blog post* [http://jeanderuelle.blogspot.com/2009/03/my-jruby-rails-app-on-jboss-can-make.html] to help you create your first pure Torquebox JRuby Telco application and our *pure JRuby on Rails TorqueBox Telco example* [http://www.mobicents.org/mss-pure-jruby-telco.html] showcasing this integration.

7.10. SIP Servlets - JAIN SLEE Interoperability

JAIN SLEE is a more complex specification than SIP Servlets, and it has been know as heavyweight and with a steep learning curve. However JAIN SLEE has standardized a high performing event driven application server, an execution environment with a good concurrency model and powerful protocol agnostic capabilities thus covering a variety of Telco protocols.

SIP Servlets on the other hand is much simpler and easier to get started with. Its focus is on extending the HTTP Servlets and Java EE hosting environments with SIP capabilities. SIP Servlets is more of a SIP programming framework, while JSLEE is a complete, self sufficient application platform. The fact that SIP Servlets is focused on SIP and Java EE makes it a natural fit to build JEE converged applications.

Table 7.4. SIP Servlets / JAIN SLEE Comparison Table

SIP Servlets	JAIN SLEE
Application Architecture	
Based on HTTP Servlets. Unit of logic is the SIP Servlets	Component based, Object Orientated architecture. Unit of logic is the Service Building Block
Composition through Application Router	Composition through parent-child relationship
Application State	
Servlets are stateless	SBBs may be stateful
Shared state stored in a session and visible to all Servlets with access to the session	SBB state is transacted and a property of the SBB itself. Shared state may be stored in a separate ActivityContext via a type safe interface
Concurrency Control	
Application managed: use of Java monitors	System Managed: isolation of concurrent transactions
Facilities (Utilities for Applications)	
Timer, Listeners	Timer, Trace, Alarm, Statistics, Profiles.
Protocol Support	
SIP, HTTP and Media (JSR 309)	Protocol agnostic. Consistent event model, regardless of protocol/resource
Availability Mechanisms	
Container managed state (session object) that can be replicated	Container managed state (SBB CMP, Facility, ActivityContext) that can be replicated
No transaction context for SIP message processing	Transaction context for event delivery

SIP Servlets	JAIN SLEE
Non transacted state operations	Container managed state operations are transacted
Facilities are non transacted	Facilities, timers, are transacted
No defined failure model	Well defined and understood failure model via transactions
Management	
No standard management mechanisms defined	JMX Interface for managing applications, life cycle, upgrades, ...

JSLEE and SIP Servlets target different audiences with different needs, but they can be complementary in a number of real world cases.

SIP Servlets focuses on SIP and its integration with Java EE. It is also more of a SIP framework within Java EE. JSLEE is an event driven application server with protocol agnostic architecture, spanning any legacy or potential future protocols. SIP Servlets applications are generally simpler to implement and accelerate time to market for Web and SIP deployment scenarios. JSLEE has a steeper learning curve and covers a wider set of target deployment environments.

As JBoss is the only vendor to implement both specifications through JBoss Communications, this makes it a natural fit to build converged and interoperable JSLEE/SIP Servlets applications that are able to comply with standards in a portable manner. We built an application that could leverage standards all the way without resorting to vendor proprietary extensions by making SIP Servlets and JSLEE work together. *Our "JSLEE and SIP-Servlets Interoperability with Mobicents Communication Platform" paper* [http://mobicents.googlecode.com/files/deruelle-JSleeSipServletsInteroperability-final.pdf] describes our approach and the possible different approaches we have identified to achieve the goal of interoperability between SIP Servlets and JSLEE.

You can also use our *JSLEE/SIP Servlets interoperability example* [http://www.mobicents.org/jslee-sips-interop-demo.html], showcasing our approach.

7.11. Eclipse IDE Tools

The SIP Servlets Eclipse tools assist developers in creating JSR-289 applications with JBoss Communications. You can use the Dynamic Web Project wizard for converged applications to get started with an empty project, and then test your application with a real SIP Phone right from the IDE.

Figure 7.3. SIP Servlets Eclipse IDE Tools

7.11.1. Pre-Install requirements

Eclipse 3.4 is required.

7.11.2. Installation

The standard Eclipse update site installation mechanism is leveraged. The JBoss Communications Update Site is at the following location: *http://mobicents.googlecode.com/svn/downloads/sip-servlets-eclipse-update-site*. After adding this update site to Eclipse you can proceed with the regular Eclipse Plug-in Installation. If you need help, the process is demonstrated in *this video* [http://www.youtube.com/watch?v=LZOmLEC2IeQ].

7.11.3. SIP Servlets Core Plug-in

This plug-in allows you to create Dynamic Web Projects with the SIP Facet. There are a number of new Dynamic Web Project configurations for Converged applications. It is best to use the ones marked as "recommended". After you complete the wizard, a complete converged project skeleton will be generated. Working with this type of project is similar to working with normal Web projects. You can see a demo *here* [http://people.redhat.com/vralev/mss-eclipse-plugin-demo/mss-eclipse.htm].

7.11.4. SIP Phone Plug-in

The SIP Phone plug-in integrates a SIP phone inside your Eclipse IDE. You can use the phone to test your SIP or Media applications. The phone uses the microphone and speakers on your computer and allows you to simulate real-world scenarios.

Best Practices

This chapter discusses Best Practices related to JBoss Communications SIP Servlets usage in real world deployments.

8.1. JBoss Communications SIP Servlets Performance Tips

Because the default profile of JBoss Communications SIP Servlets is targeted at a development environment, some tuning is required to make the server performance suitable for a production environment.

A useful presentation from OKI Japan

8.1.1. Tuning JBoss

To ensure the server is finely tuned for a production envirionment, certain configuration must be changed. Visit the *JBoss Application Server Tuning* [http://wiki.jboss.org/wiki/Wiki.jsp?page=JBossASTuningSliming] wiki page to learn about optimization techniques.

While it is preferable to have a fast Application Server, most of the information doesn't apply to JBoss Communications. In summary, the most important optimization technique is to remove logs, leaving only what is required.

Check the log configuration file in the following location and review the information.

```
$JBOSS_HOME/server/default/conf/jboss-log4j.xml
```

8.1.2. Tuning JBoss Communications SIP Servlets

- *Congestion Control*: It is recommended that this feature is enabled to avoid overload of the server and that the *sipMessageQueueSize* and *memoryThreshold* parameters are tuned according to *Section 7.2, "Concurrency and Congestion Control"*.
- *Concurrency* : *Default Value: None.* For better performance, it is recommended to leave this value set to **`None`**.

8.1.3. Tuning The JAIN SIP Stack

The stack can be fine-tuned by altering the SIP stack properties, defined in the external properties file specified by the *sipStackPropertiesFile* attribute as described in *Section 2.3.1, "Configuring SIP Connectors"*.

- *gov.nist.javax.sip.THREAD_POOL_SIZE*

  Default value: 64

This thread pool is responsible for parsing SIP messages received from socket messages into objects.

A smaller value will make the stack less responsive, since new messages have to wait in a queue for free threads. In UDP, this can lead to more retransmissions.

Large thread pool sizes result in allocating resources that are otherwise not required.

- *gov.nist.javax.sip.REENTRANT_LISTENER*

  Default value: true

  This flag indicates whether the SIP stack listener is executed by a single thread, or concurrently by the threads that parse the messages.

  JBoss Communications SIP Servlets expects this flag to be set to **`true`**, therefore do not change the value.

- *gov.nist.javax.sip.LOG_MESSAGE_CONTENT*

  Default value: true

  Set the parameter to **`false`** to disable message logging.

- *gov.nist.javax.sip.TRACE_LEVEL=0*

  Default value: 32.

  Set the parameter to **`0`** to disable JAIN SIP stack logging.

- *gov.nist.javax.sip.RECEIVE_UDP_BUFFER_SIZE=65536* *and* *gov.nist.javax.sip.SEND_UDP_BUFFER_SIZE=65536*

  Default value: 65536.

  Those properties control the size of the UDP buffer used for SIP messages. Under load, if the buffer capacity is overflown the messages are dropped causing retransmissions, further increasing the load and causing even more retransmissions.

- *gov.nist.javax.sip.MAX_MESSAGE_SIZE=10000*

  Default value: 10000.

  This property controls the maximum size of content that can be read for a SIP Message on UDP. The default is 65536. The average UDP message size is quite lower than this so reducing this property will benefit memory usage since a byte buffer of this size is created for every message received.

  It also defines the maximum size of content that a TCP connection can read. Must be at least 4K. Default is "infinity" -- ie. no limit. This is to prevent DOS attacks launched by writing to a TCP connection until the server chokes.

- *gov.nist.javax.sip.TCP_POST_PARSING_THREAD_POOL_SIZE=30*

  Default value: 30.

  Use 0 or do not set this option to disable it. When using TCP, your phones/clients usually connect independently, creating their own TCP sockets. Sometimes however SIP devices are allowed to tunnel multiple calls over a single socket. This can also be simulated with SIPP by running "sipp -t t1".

  In the stack, each TCP socket has it's own thread. When all calls are using the same socket they all use a single thread, which leads to severe performance penalty, especially on multi-core machines. This option instructs the SIP stack to use a thread pool and split the CPU load between many threads. The number of the threads is specified in this parameter.

  The processing is split immediately after the parsing of the message. It cannot be split before the parsing because in TCP the SIP message size is in the Content-Length header of the message and the access to the TCP network stream has to be synchronized.

  Additionally, in TCP the message size can be larger. This causes most of the parsing for all calls to occur in a single thread, which may have impact on the performance in trivial applications using a single socket for all calls. In most applications it doesn't have performance impact. If the phones/clients use separate TCP sockets for each call, this option doesn't have much impact, except the slightly increased memory footprint caused by the thread pool. It is recommended to disable this option in this case by setting it 0 or not setting it at all. You can simulate multi-socket mode with "sipp -t t0". With this option also we avoid closing the TCP socket when something fails, because we must keep processing other messages for other calls. Note: This option relies on accurate Content-Length headers in the SIP messages. It cannot recover once a malformed message is processed, because the stream iterator will not be aligned any more. Eventually the connection will be closed.

The full list of JAIN SIP stack properties is available from *the SIP Stack Properties Home Page* [http://snad.ncsl.nist.gov/proj/iptel/jain-sip-1.2/javadoc/javax/sip/SipStack.html] and the full list of implementation specific properties are available from the *SIP Stack Implementation Home Page* [http://snad.ncsl.nist.gov/proj/iptel/jain-sip-1.2/javadoc/gov/nist/javax/sip/SipStackImpl.html].

8.1.4. Tuning The JVM

The following tuning information applies to Sun JDK 1.6, however the information should also apply to Sun JDK 1.5.

Note

For more information on tuning JBoss Communications SIP Servlets performance, refer to the *OKI Japan Presentation* [http://www.slideshare.net/jean.deruelle/tuning-and-development-with-sip-servlets-on-mobicents].

> For more information on performance, refer to the *Performance White Paper* [http://java.sun.com/performance/reference/whitepapers/6_performance.html].

To pass arguments to the JVM, change `$JBOSS_HOME/bin/run.conf` (Linux) or `$JBOSS_HOME/bin/run.bat` (Windows).

- **Garbage Collection**

  JVM ergonomics automatically attempt to select the best garbage collector. The default behaviour is to select the throughput collector, however a disadvantage of the throughput collector is that it can have long pauses times, which ultimately blocks JVM processing.

  For low-load implementations, consider using the incremental, low-pause, garbage collector (activated by specifying ***`-XX:+UseConcMarkSweepGC -XX:+CMSIncrementalMode`***). Many SIP applications can benefit from this garbage collector type because it reduces the retransmission amount.

  For more information please read: *Garbage Collector Tuning* [http://java.sun.com/javase/technologies/hotspot/gc/gc_tuning_6.html]

- **Heap Size**

  Heap size is an important consideration for garbage collection. Having an unnecessarily large heap can stop the JVM for seconds, to perform garbage collection.

  Small heap sizes are not recommended either, because they put unnecessary pressure on the garbage collection system.

8.1.5. Tuning The Operating System

The following tuning information is provided for Red Hat Enterprise Linux (RHEL) servers that are running high-load configurations. The tuning information should also apply to other Linux distributions.

After you have configured RHEL with the tuning information, you must restart the operating system. You should see improvements in I/O response times. With SIP, the performance improvement can be as high as 20%.

- **Large Memory Pages**

  Setting large memory pages can reduce CPU utilization by up to 5%.

  Ensure that the option ***`-XX:+UseLargePages`*** is passed and ensure that the following Java HotSpot(TM) Server VM warning does not occur:

  `Failed to reserve shared memory (errno = 22)" when starting JBoss.` It means that the number of pages at OS level is still not enough.

To learn more about large memory pages, and how to configure them, refer to *Java Support for Large Memory Pages* [http://java.sun.com/javase/technologies/hotspot/largememory.jsp] and *Andrig's Miller blog post* [http://andrigoss.blogspot.com/2008/02/jvm-performance-tuning.html].

- *Network buffers*

  You can increase the network buffers size by adding the following lines to your `/etc/sysctl.conf` file:

  - **`net.core.rmem_max = 16777216`**
  - **`net.core.wmem_max = 16777216`**
  - **`net.ipv4.tcp_rmem = 4096 87380 16777216`**
  - **`net.ipv4.tcp_wmem = 4096 65536 16777216`**
  - **`net.core.netdev_max_backlog = 300000`**

- Execute the following command to set the network interface address:

  ```
  sudo ifconfig [eth0] txqueuelen 1000 #
  ```

  Replace [eth0] with the correct name of the actual network interface you are setting up.

8.2. NAT Traversal

In a production environment, it is common to see SIP and Media data passing through different kinds of Network Address Translation (NAT) to reach the required endpoints. Because NAT Traversal is a complex topic, refer to the following information to help determine the most effective method to handle NAT issues.

8.2.1. STUN

STUN (Session Traversal Utilities for NAT) is not generally considered a viable solution for enterprises because STUN cannot be used with symmetric NATs.

Most enterprise-grade firewalls are symmetric, therefore STUN support must be provided in the SIP Clients themselves.

Most of the proxy and media gateways installed by VoIP providers recognize the public IP address the packets have originated from. When both SIP end points are behind a NAT, they can act as gateways to clients behind NAT.

8.2.2. TURN

TURN (Traversal Using Relay NAT) is an IETF standard, which implements media relays for SIP end-points. The standard overcomes the problems of clients behind symmetric NATs which cannot rely on STUN to solve NAT traversal.

TURN connects clients behind a NAT to a single peer, providing the same protection offered by symmetric NATs and firewalls. The TURN server acts as a relay; any data received is forwarded.

This type of implementation is not ideal. It assumes the clients have a trust relationship with a TURN server, and a request session allocation based on shared credentials.

This can result in scalability issues, and requires changes in the SIP clients. It is also impossible to determine when a direct, or TURN, connection is appropriate.

8.2.3. ICE

ICE (Interactive Connection Establishment) leverages both STUN and TURN to solve the NAT traversal issues.

It allows devices to probe for multiple paths of communication, by attempting to use different port numbers and STUN techniques. If ICE support is present in both devices, it is quite possible that the devices can initiate and maintain communication end-to-end, without any intermediary media relay.

Additionally, ICE can detect cases where direct communication is impossible and automatically initiate fall-back to a media relay.

ICE is not currently in widespread use in SIP devices, because ICE capability must be embedded within the SIP devices.

Depending on the negotiated connection, a reINVITE may be required during a session, which adds more load to the SIP network and more latency to the call.

If the initiating ICE client attempts to call a non-ICE client, then the call setup-process will revert to a conventional SIP call requiring NAT traversal to be solved by other means.

8.2.4. Other Approaches

Other approaches include using proxy and media that can act as gateways, Session Border Controllers, enhanced Firewall with Application Layer Gateway (ALG) and Tunnelling.

Appendix A. Revision History

Revision History

Revision 4.0 Mon Jan 24 2011 TomWells<twells@redhat.com>

Publican valid edition of community documentation.

Revision 3.0 Thu Jun 11 2009 JaredMorgan<jmorgan@redhat.com>

Second release of the "parameterized" documentation.

Revision 2.0 Fri Mar 06 2009 DouglasSilas<dhensley@redhat.com>

First release of the "parameterized", and much-improved JBCP documentation.