SIP Servlets Server-Installing, Configuring and Running

Table of Contents

Getting Started with Restcomm for JBoss AS7

Features not yet available on Restcomm for JBoss AS7

- SIP Failover
- SNMP
- Jopr Monitoring

Some of the features mentioned above will likely be added in the future. As of the time of this writing, they are not available. Even though Jopr monitoring is not available, there is a Command Line Interface (CLI), which will be discussed further down. As the features become available, this guide will be updated to reflect the changes.

Downloading and Starting Restcomm SIP Servlets for JBoss AS7

If you have been working with JBoss for some time, you will quickly notice that the JBoss AS7 iteration has gone through a lot of changes. This guide will help you understand how you can quickly get started with JBoss AS7 within the Restcomm framework.

You can go to the link below to download the latest Restcomm SIP Servlets for JBoss AS7: link: Download Latest Version of Restcomm SIP Servlets for JBoss AS7

You will need to extract the content of the file into a directory on your local system. The root directory of the Restcomm SIP Servlets for JBoss AS7 that you downloaded will be referred to in this guide as $JBOSS_HOME.

If this is your first time working with Restcomm SIP Servlets for JBoss, you will need to make sure you have Java Run Time or JDK installed on your computer. You will also need to have the environment variables set. See the links below to learn how to get JRE or JDK setup on your system.

Installing and Configuring JDK

Setting Environment Variables

1. Starting Restcomm SIP Servlets for JBoss AS7 To start the server do the following:

```
$JBOSS_HOME/bin/standalone.sh
```

During the startup process, you will notice that the final part of the log output will be similar to the truncated output below. Notice that the Admin Console interface can be accessed at http://172.0.0.1:9990. This will be explained later.

```
14:28:43,972 INFO  [org.jboss.as] (Controller Boot Thread) JBAS015951: Admin console
listening on http://127.0.0.1:9990
14:28:43,974 INFO  [org.jboss.as] (Controller Boot Thread) JBAS015874: JBoss AS
7.1.2.Final "Steropes"
started in 22148ms - Started 222 of 306 services (83 services are passive or on-
demand)
```

You will notice that the startup is very fast. The reason for this is that JBoss was rewritten from the ground up for speed with services being started concurrently and non critical services remain passive until first use. This provides better system resource management. With the simple startup above, you will be able to enter the default web interface of the application server by going to this url http://127.0.0.1:8080. The result will show a screenshot similar to the one below.

Figure 1. JBoss Application Server 7 Welcome Page

With the standard startup script, you will not have access to any SIP functionalities. This is because of the modular approach implemented in JBoss AS7. There is a configuration file that needs to be used to activate additional functionalities like SIP and High Availability.

In order to start the Restcomm SIP Servlets for JBoss AS7 with SIP functionalities, you need to append the startup script with the SIP configuration file. The configuration files are located in the $JBOSS_HOME/standalone/configuration directory. You can see the content of the directory below

```
application-roles.properties  mgmt-users.properties     standalone-ha.xml
application-users.properties  mss-sip-stack.properties  standalone-sip.xml
dars                          standalone-full-ha.xml     standalone.xml
logging.properties            standalone-full.xml        standalone_xml_history
```

Starting Restcomm SIP Servlets for JBoss AS7 with SIP

If you want to start Restcomm SIP Servlets with SIP services activated, you need to go to the $JBOSS_HOME/bin directory. Type the following command:

```
./standalone.sh -c standalone-sip.xml
```

1. You will see a message similar to the one below once the server is successfully started

```
20:43:21,487 INFO  [org.jboss.as.server] (ServerService Thread Pool -- 37) JBAS018559:
Deployed "click2call.war"
20:43:21,489 INFO  [org.jboss.as.server] (ServerService Thread Pool -- 37) JBAS018559:
Deployed "sip-servlets-management.war"
20:43:21,647 INFO  [org.jboss.as] (Controller Boot Thread) JBAS015951: Admin console
listening on http://127.0.0.1:9990
20:43:21,648 INFO  [org.jboss.as] (Controller Boot Thread) JBAS015874: JBoss AS
7.1.2.Final "Steropes" started in 26560ms - Started 232 of 321 services (88 services
are passive or on-demand)
```

The click2call SIP sample application bundled with Restcomm SIP Servlets will become available at this url http://127.0.0.1:8080/click2call. You can configure multiple SIP softphones to use the sample application. See the section below for how to configure and test the SIP sample application.

Testing Click2Call with Restcomm SIP Servlets for JBOSS AS7

Once the server is started as stated in the previous section, you can configure multiple instances of any SIP softphone you prefer. In this example, Linphone will be used.

```
(configuring two instances of Linphone)

start Linphone
go to the Options menu

On the Network Settings tab,
    SIP (UDP) port to 5061. (leave the rest as default)
On the Manage SIP Accounts tab,
    click the add button
    Your SIP identity: = sip:linphone@127.0.0.1:5080
    SIP Proxy address: = sip 127.0.0.1:5080

Leave the rest of the settings as default.


Configuring Linphone (on the second shell)

go to the Options menu

On the Network Settings tab,
    SIP (UDP) port to 5062. (leave the rest as default)
On the Manage SIP Accounts tab,
    click the add button
    Your SIP identity: = sip:linphone2@127.0.0.1:5080
    SIP Proxy address: = sip 127.0.0.1:5080

Leave the rest of the settings as default.
```

A correctly configured Linphone will look like the screenshot below.

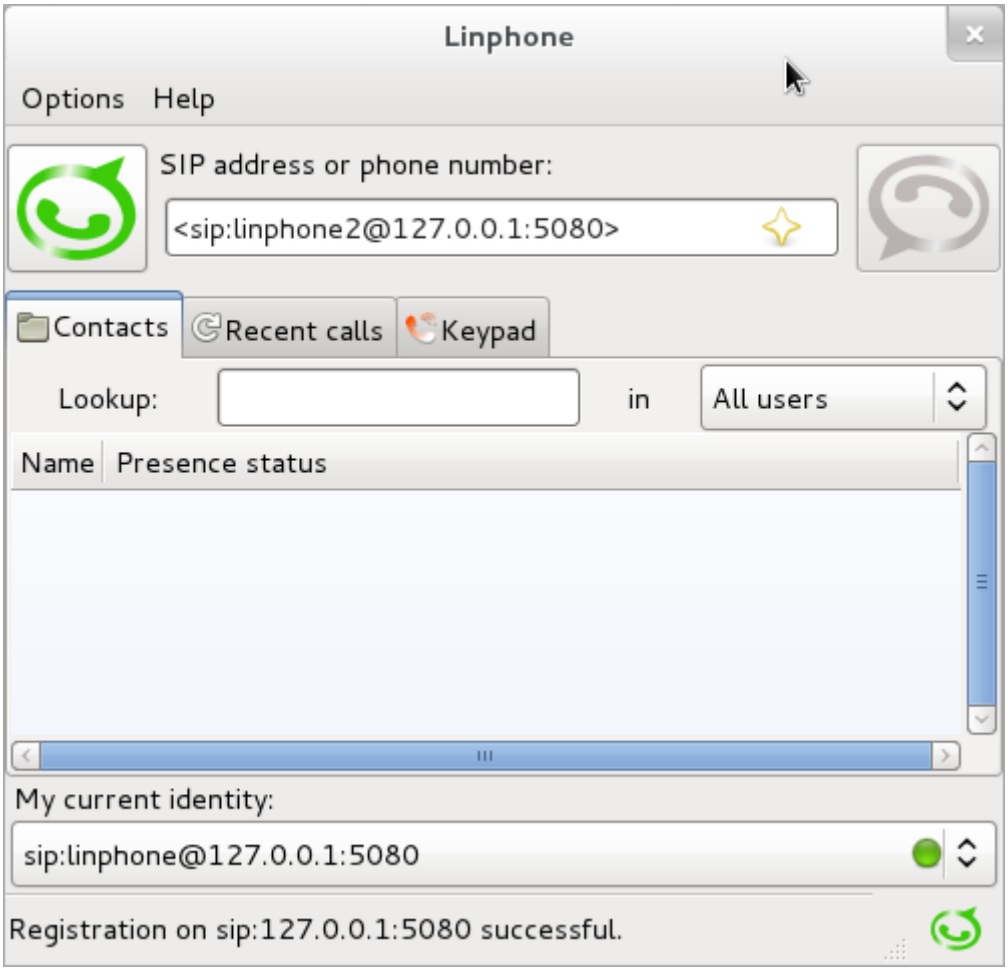

Figure 2. Successfully Configured Linphone

Once the phones are successfully registered with the Restcomm SIP Servlets for JBoss AS7 server, you can check the result in the sample SIP application at this url, http://127.0.0.1:8080/click2call

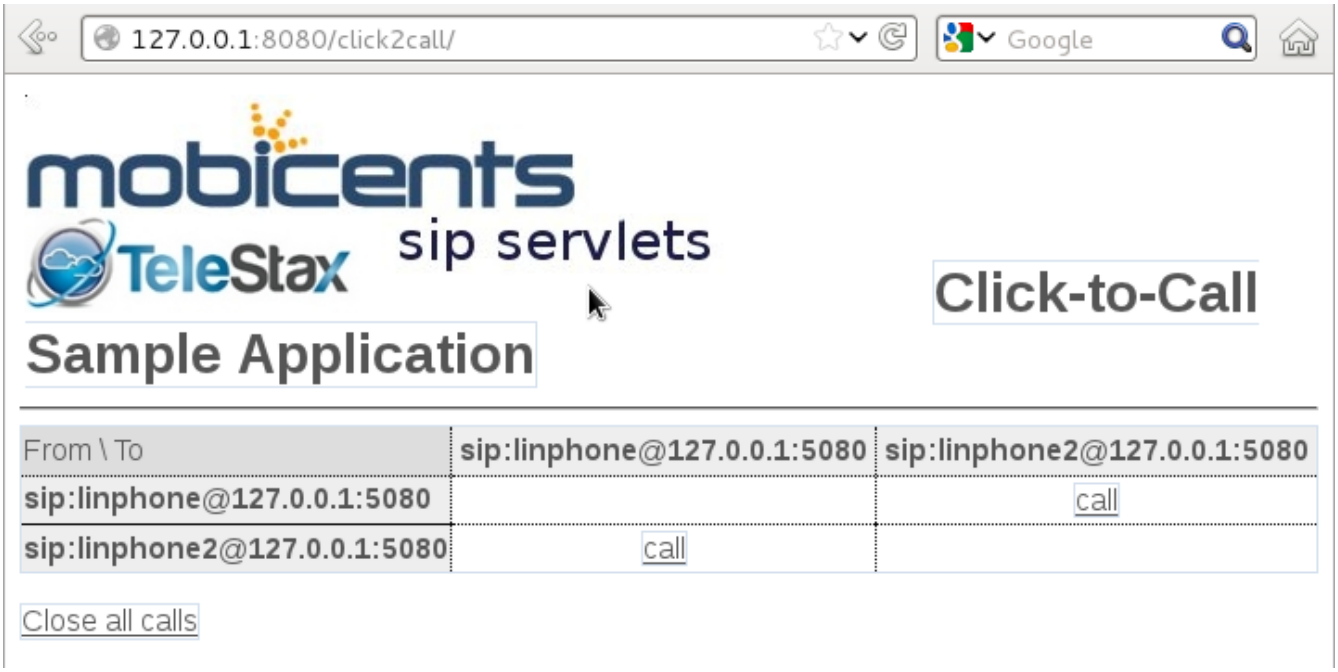

Figure 3. Click2call SIP Registered Softphones

You can make calls from the sample click2call application and see the logs in the shell terminal you used to start the Restcomm SIP Servlets for JBoss AS7 server.

Command Line Interface for Restcomm SIP Servlets JBoss AS7

Part of the task of any administrator who has to manage a JBoss server will be to monitor services offered to clients. There is a command line interface bundled with JBoss AS7 which can be accessed by going to the $JBOSS_HOME/bin directory.

You need to make sure that the JBoss server is running on your system and listening on port 9999. The section below will work you through steps to familiarize yourself with the CLI.

There are so many features available with the Restcomm SIP Servlets for JBoss AS7 CLI. The example below will concentrate on getting data from the SIP you started using the [path]_ ./standalone.sh -c standalone-sip.xml _ script.

In the $JBOSS_HOME/bin directory, type

```
./jboss-cli.sh
```

(This will show the message below)

```
You are disconnected at the moment.
Type 'connect' to connect to the server or
'help' for the list of supported commands.
```

At the [disconnected /] command prompt, type

```
connect
```

When you see the [standalone@localhost:9999 /] at the prompt, you are successfully connected to the server.

Navigating the CLI

Moving around the Restcomm SIP Servlets for JBoss AS7 CLI is similar to normal file system with a few exceptions. You can use commands like, (ls, cd, cd..) to navigate around the CLI

Follow the steps below to access SIP information from the CLI

```
At the prompt type (ls)

[standalone@localhost:9999 /] ls
core-service                  deployment                    extension
interface                     path                          socket-binding-group
subsystem                     system-property               launch-type=STANDALONE
management-major-version=1    management-minor-version=2    name=linux-fedora
namespaces=[]                 process-type=Server           product-name=undefined
product-version=undefined     profile-name=undefined        release-codename=Steropes
```

```
release-version=7.1.2.Final   running-mode=NORMAL           schema-locations=[]
server-state=running

[standalone@localhost:9999 /] cd deployment

[standalone@localhost:9999 deployment] ls
click2call.war                sip-servlets-management.war

[standalone@localhost:9999 deployment] cd click2call.war

[standalone@localhost:9999 deployment=click2call.war] ls
subdeployment
subsystem
content=[{"path" => "deployments/click2call.war","relative-to" =>
"jboss.server.base.dir","archive" => true}]
enabled=true
name=click2call.war
persistent=false
runtime-name=click2call.war
status=OK

[standalone@localhost:9999 deployment=click2call.war] cd subsystem

[standalone@localhost:9999 subsystem] ls
sip   web

[standalone@localhost:9999 subsystem] cd sip


[standalone@localhost:9999 subsystem=sip] ls
servlet
active-sip-application-sessions=7
active-sip-sessions=8
app-name=org.mobicents.servlet.sip.example.SimpleApplication
expired-sip-application-sessions=25
expired-sip-sessions=26
max-active-sip-sessions=-1
rejected-sip-application-sessions=0
rejected-sip-sessions=0
sip-application-session-avg-alive-time=180
sip-application-session-max-alive-time=230
sip-application-sessions-created=32
sip-application-sessions-per-sec=0.0
sip-session-avg-alive-time=162
sip-session-max-alive-time=180
sip-sessions-created=34
sip-sessions-per-sec=0.0
```

No SIP data on the CLI

The data from the SIP subsystem are only available if you have the click2call sample application running and your softphones are connected to the server.

1. SIP Servlets Management Console There is also a SIP servlets management console that is available at this url http://127.0.0.1:8080/sip-servlets-management. The resulting page will be similar to the screenshot below. More information will be provided about the SIP servlets management console in later chapters of this guide.

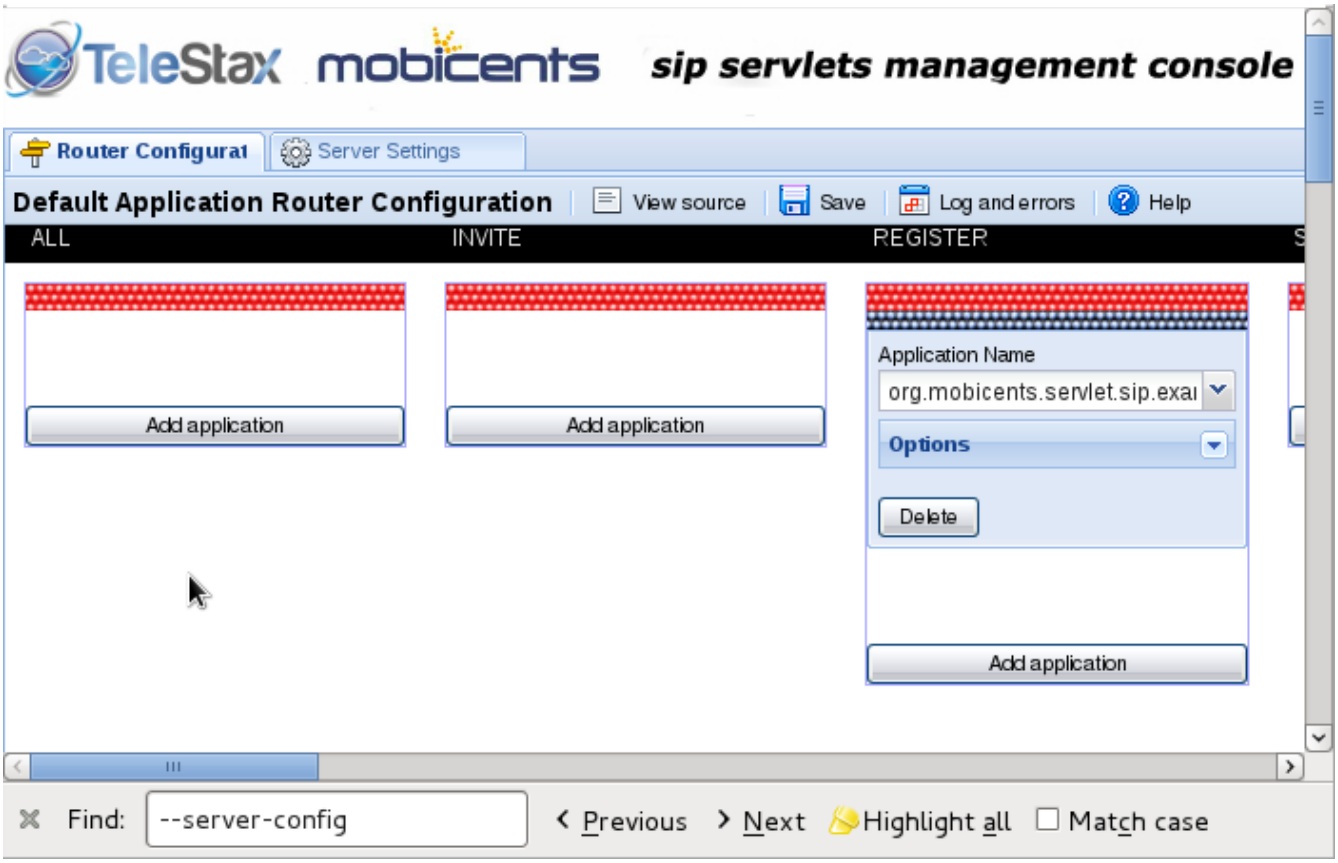

Figure 4. JBoss Application Server 7 Management Console

Accessing Management Console

Restcomm SIP Servlets for JBoss AS7 provides a management console that can be useful for accessing vital information about your server. In the welcome page that appears when you access http://127.0.0.1:8080, there is a link that points to the Administration Console.

If you don't have a user account for the management console, you will see a screenshot like the one below. It contains instructions about how to create a user account.

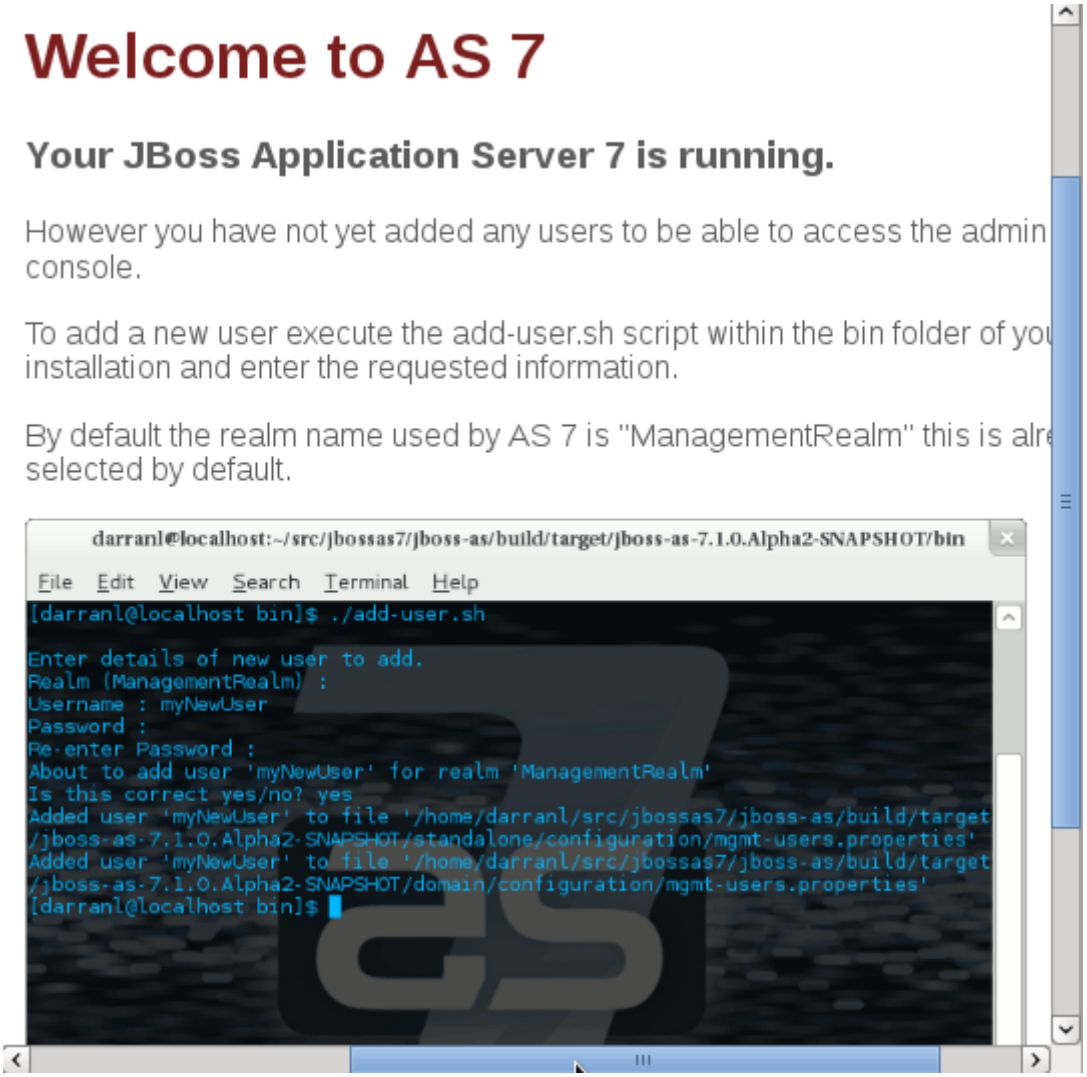

Figure 5. Administration Console Error Page

1. Creating a User Account Go to the $JBoss_HOME/bin directory and run the ./add-user.sh script. You can follow the interactive user mode to create an account for the Administration Console.

Once the user account has been created, you can access the Administration Console at this address http://127.0.0.1:9990/console/

The screenshot below shows you what the Administration Console looks like.

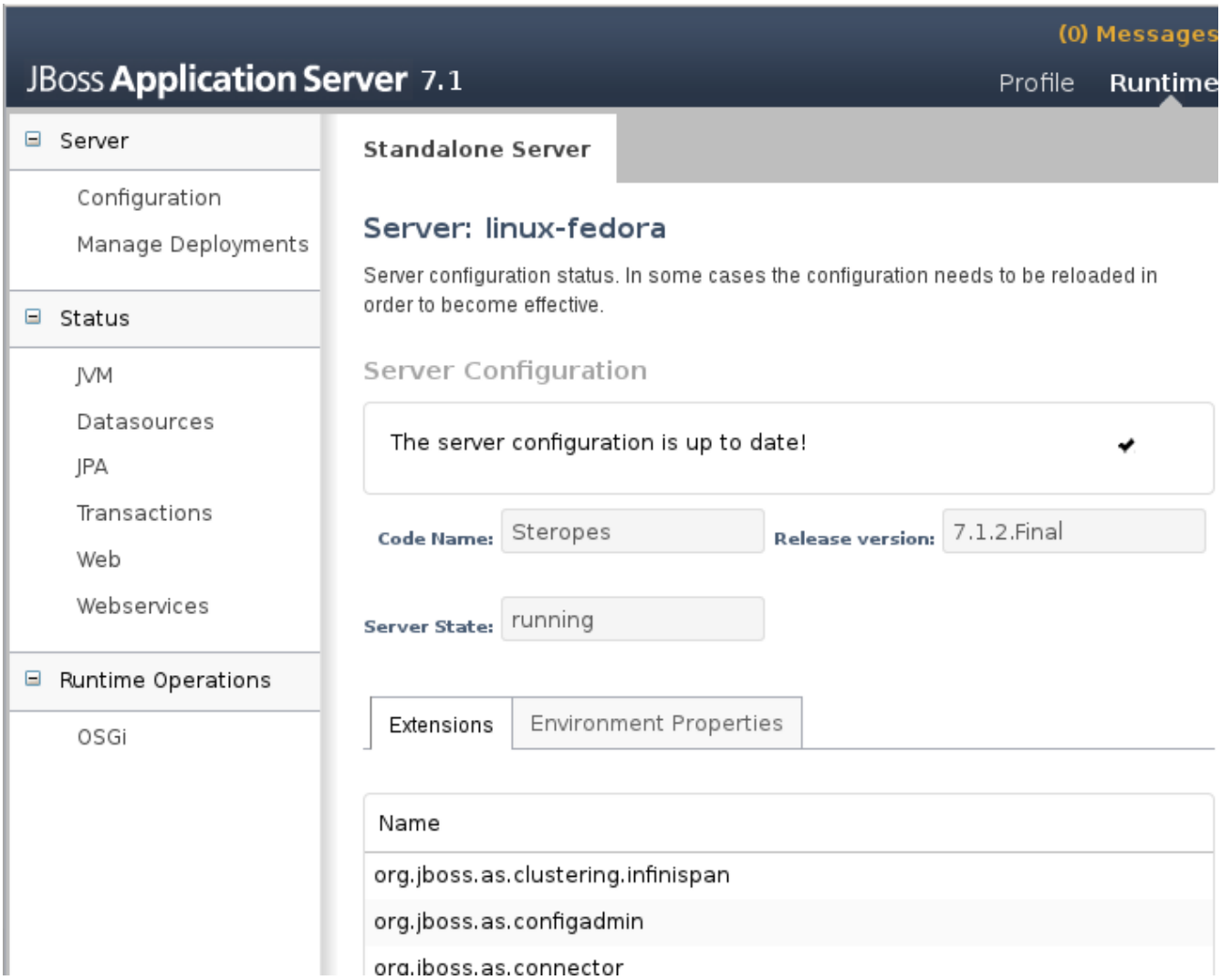

Figure 6. Administration Console

Deleting Administration Console User Account

Deleting the user account isn't very intuitive. In the event that you will need to remove an account and create another one, you can remove the account from the mgmt-users.properties file. It is located in the $Restcomm_JBoss_HOME/standalone/configuration directory. If you are running in the domain mode, you will need to check the corresponding configuration directory.

Installing the Restcomm for JBoss Binary Distribution on

For this procedure, it is assumed that the downloaded archive is saved in the *My Downloads* folder. . 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, *-jboss-<version>*. . Extract the contents of the archive, specifying the destination folder as the one created in the previous step. You can either use Winzip or the opensource tool called 7-Zip to extract the content of the donwloaded Restcomm SIP Servlets for JBoss AS7 file . It is recommended that the folder holding the Restcomm SIP Servlets for JBoss files (in this example, the folder named *-jboss-<version>*) is moved to a user-defined location for storing executable programs. For example, the *Program Files* folder.

Procedure: Running Restcomm SIP Servlets for JBoss on

There are several ways to start Restcomm 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 Restcomm 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 "-jboss-<version>"
   ```

3. Start the JBoss Application Server by executing one of the following files:
   - *run.bat* batch file:

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

   - *run.jar* executable Java archive:

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

Getting Started with Restcomm SIP Servlets for Tomcat 7

You can download the latest Restcomm SIP Servlets for Tomcat 7 link: Download Latest Version of for Tomcat 7

The content of the downloaded file can be extracted to any location you prefer on your computer. The root directory to which the content of the download is extracted will be referred to as $CATALINA_HOME.

The content of the $CATALINA_HOME/bin is similar to the output below.

```
bootstrap.jar                commons-daemon.jar           setclasspath.bat  tool-wrapper.bat
catalina.bat                 commons-daemon-native.tar.gz setclasspath.sh   tool-wrapper.sh
catalina-tasks.xml           configtest.bat               shutdown.bat      version.bat
cpappend.bat                 configtest.sh                shutdown.sh       version.sh
daemon.sh                    digest.bat                   startup.bat
digest.sh                    startup.sh                   tomcat-juli.jar
tomcat-native.tar.gz
```

You can start Restcomm SIP Servlets for Tomcat 7 by going to $CATALINA_HOME/bin directory and typing the following:

```
sudo ./catalina.sh run
```

The startup process is slightly different from Restcomm SIP Servlets for JBoss AS7. If you see an output like the one below, you know that Tomcat is correctly started. This is a truncated log from the startup process.

```
2012-08-21 22:23:41,025 INFO  [SipApplicationDispatcherImpl] (main)
SipApplicationDispatcher Started
2012-08-21 22:23:41,025 INFO  [SipStandardService] (main) SIP Standard Service
Started.
Aug 21, 2012 10:23:41 PM org.apache.catalina.startup.Catalina start
INFO: Server startup in 3608 ms
```

If you get an error message about environment variables or Java, make sure you have the CATALINA environment variables set.

Setting Environment Variables - JAVA and CATALINA

Testing Click2CallAsync with Restcomm for Tomcat 7

If Restcomm SIP Servlets for Tomcat 7 is started and running, you should be able to use your web browser to access the welcome page at this url http://127.0.0.1:8080/ This will show you a screenshot similar to the one below.

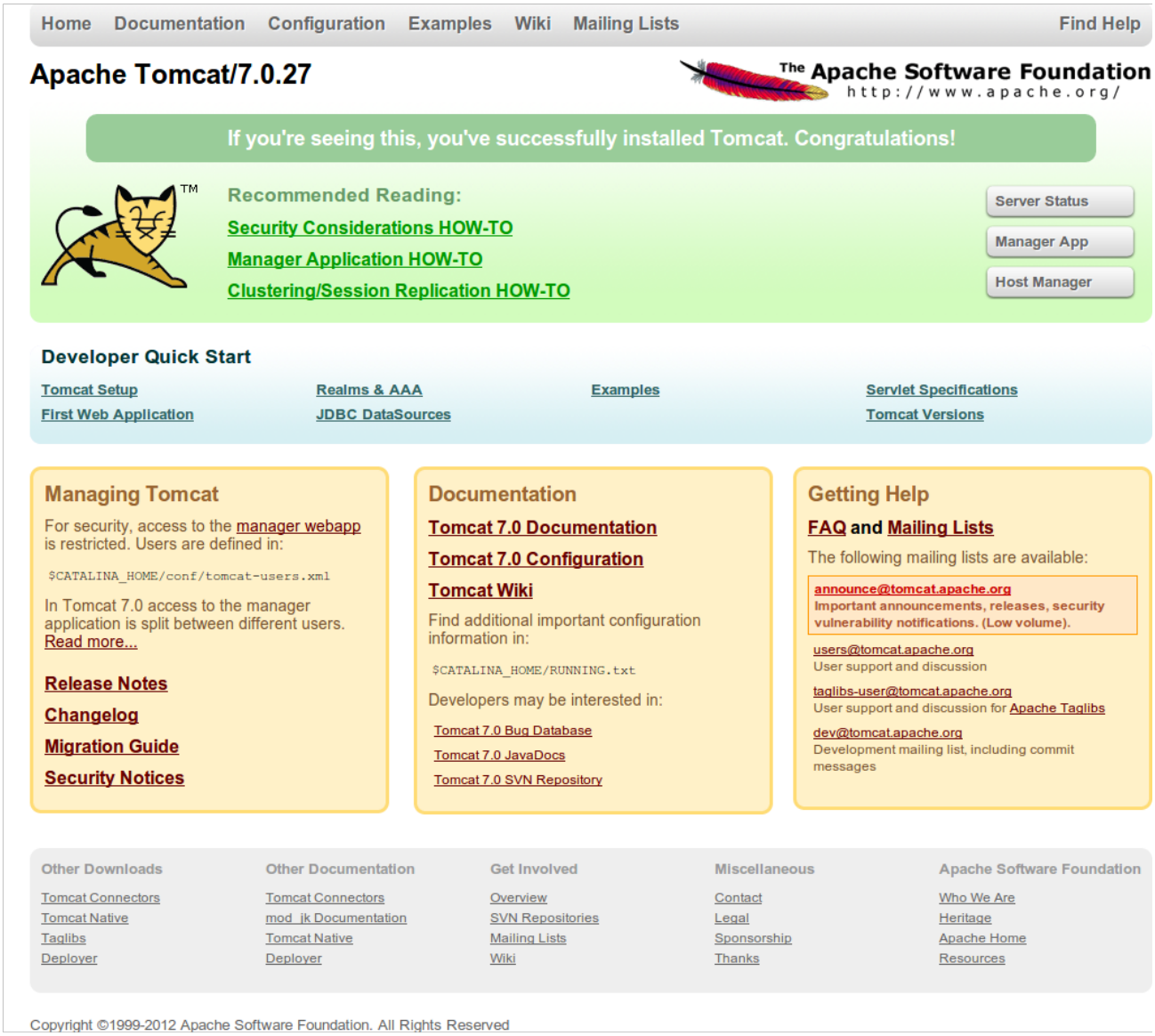

Figure 7. JBoss Application Server 7 Welcome Page

Deploying your application once the server is running is simple. You need to copy your .War files to the $CATALINA_HOME/webapps directory.

There is a pre-installed sample SIP application that you can use to test your Restcomm SIP Servlets Tomcat 7 configuration. The application is also located in the $CATALINA_HOME/webapps directory

Start your web browser and go to the link, http://127.0.0.1:8080/Click2CallAsync/

Sample Application Name

Note that the application name is case-sensitive and will not work if you try to access it as http://127.0.0.1:8080/click2callasync/

The sample SIP application page will be similar to the screenshot below.

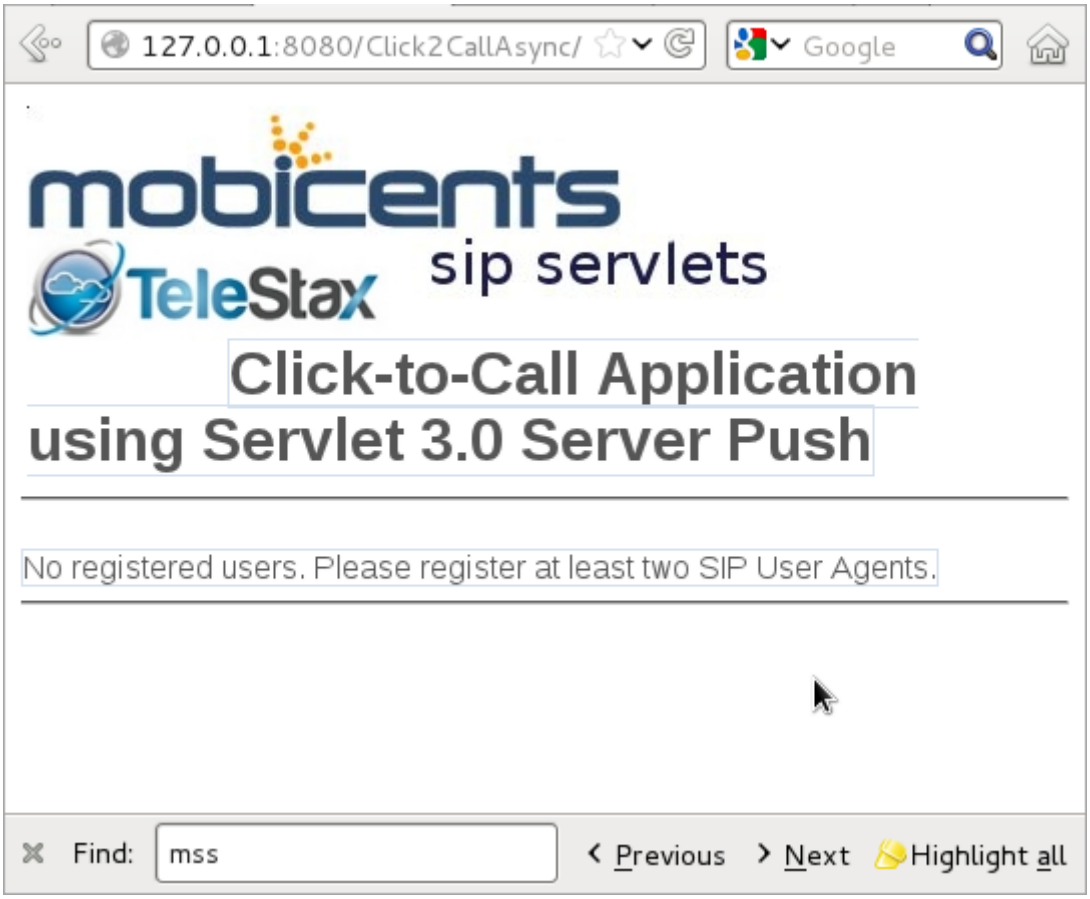

Figure 8. SIP Sample Click2CallAsync Application

In order to use the application, you can download a softphone and start multiple instances of the phone on a single server. In this guide, the softphone that will be used is Linphone. The configuration is as follows:

Multiple Instances of Linphone

On some Linux systems, you might need to use a different user profile in order to start a second instance of Linphone. Ex. sudo linphone

```
(configuring two instances of Linphone)

start Linphone
go to the Options menu

On the Network Settings tab,
        SIP (UDP) port to 5061. (leave the rest as default)
On the Manage SIP Accounts tab,
        click the add button
        Your SIP identity: = sip:linphone@127.0.0.1:5080
        SIP Proxy address: = sip 127.0.0.1:5080

Leave the rest of the settings as default.


Configuring Linphone (on the second shell)

go to the Options menu

On the Network Settings tab,
        SIP (UDP) port to 5062. (leave the rest as default)
On the Manage SIP Accounts tab,
        click the add button
        Your SIP identity: = sip:linphone2@127.0.0.1:5080
        SIP Proxy address: = sip 127.0.0.1:5080

Leave the rest of the settings as default.
```

Once the softphones are configured and are successfully registered with the Restcomm SIP Servlets for Tomcat 7 server, you will see a screenshot like the one below in the web browser at this url http://127.0.0.1:8080/Click2CallAsync/

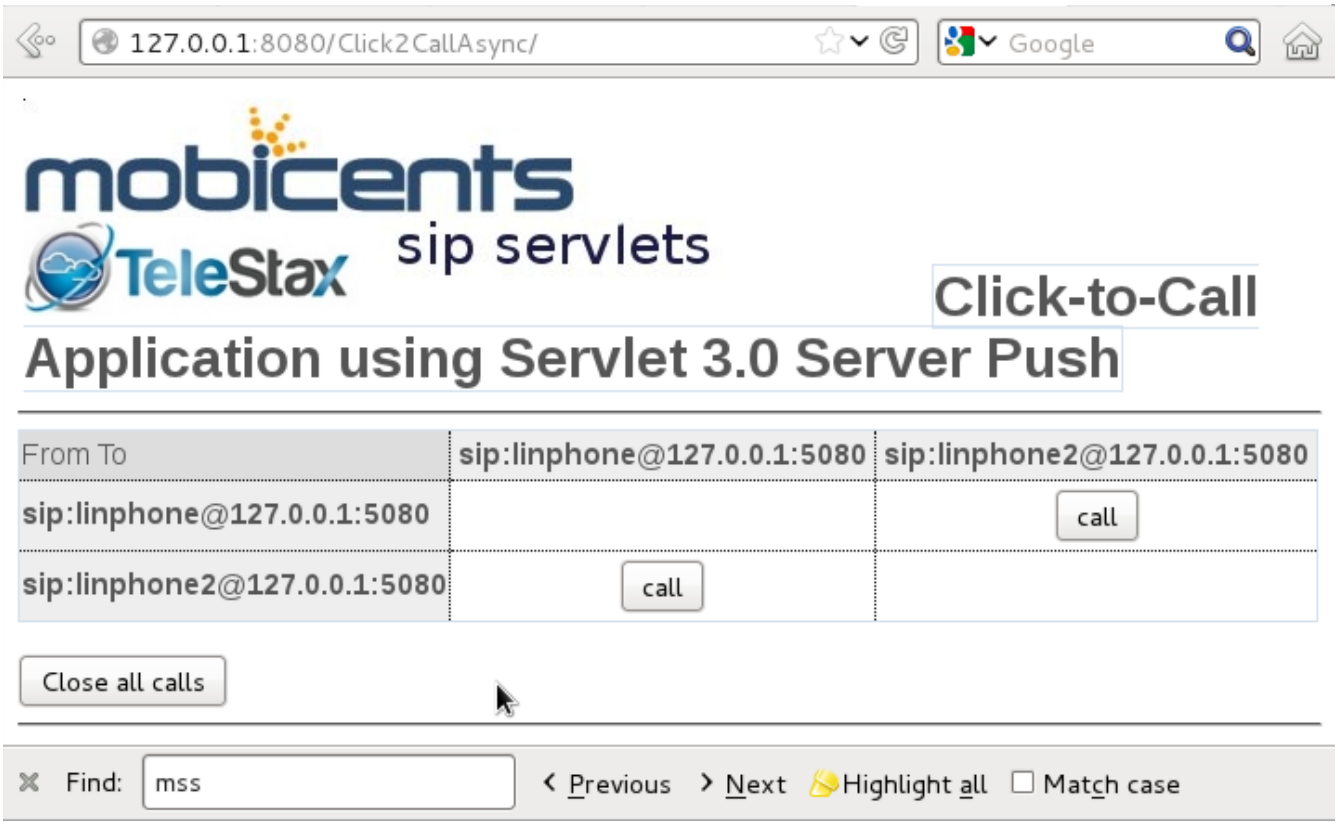

Figure 9. SIP Click2CallAsync with Registers Clients

You can make calls using the application and the softphones you configured will start ringing. It is important to start Restcomm SIP Servlets for Tomcat 7 in a terminal using the (./catalina.sh run) script. It will help with troubleshooting SIP calls. The logs you see on the terminal will let you know when a softphone registers with the Tomcat server and you will also be able to see the status of call setup and shutdown.

Stopping Restcomm SIP Servlets for Tomcat 7

The best way to stop a server is using the CTRL-D on the terminal in which the server was started. If you started the Restcomm SIP Servlets for Tomcat 7 server using the $CATALINA_HOME/bin/startup.sh, you can stop the server using $CATALINA_HOME/bin/shutdown.sh

Tomcat for Windows

Installing the Restcomm SIP Servlets for Tomcat 7 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 Restcomm SIP Servlets for Tomcat binary distribution you downloaded. In these instructions, we will refer to this folder as *-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.
    a. 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.

b. 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 kbd:[Shift] key and right-click on either a folder, the desktop, or inside a folder. This will cause an 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.

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

```
C:\Users\Me\My Downloads\-tomcat-<version>>jar -xvf ""
```

3. At this point, you may want to move the folder holding the Restcomm SIP Servlets for Tomcat binary files (in this example, the folder named *-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\-tomcat-<version>>delete ""
```

Configuring

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

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

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/user/temp/apps/sip_servlets_server/
Using CATALINA_HOME:   /home/user/temp/apps/sip_servlets_server/
Using CATALINA_TMPDIR: /home/user/temp/apps/sip_servlets_server/temp
Using JRE_HOME:       /etc/java-config-2/current-system-vm
```

Detailed instructions are given below, arranged by platform.

Procedure: Running Restcomm 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.

   a. 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 "-tomcat-<version>"
   ```

   b. Start the Tomcat Servlet Container by running the executable *startup.bat* batch file:

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

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/user/temp/apps/sip_servlets_server
Using CATALINA_HOME:   /home/user/temp/apps/sip_servlets_server
Using CATALINA_TMPDIR: /home/user/temp/apps/sip_servlets_server/temp
Using JRE_HOME:       /etc/java-config-2/current-system-vm
```

Procedure: Stopping Restcomm SIP Servlets for Tomcat on Windows

1. 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\-tomcat-<version>>bin\shutdown.bat
```

Sip Connectors

Restcomm SIP Servlets comes with default settings that are designed to get your system up and running without the need to know about all the detailed configurations. That said, there are situations in which you might like to fine-tune your setttings to adapt it to your needs. That is what the following section will help you achieve. You will get a better understand of SIP connectors and how to make them work for you.

Configuring SIP Connectors and Bindings

There are two important configuration files that you might need to modifying depending on your system needs. The standalone-sip.xml file in Restcomm SIP Servlets for JBoss AS7 and the server.xml file in Restcomm SIP Servlets for Tomcat. The extracts below will give you a snapshot of default configurations.

For JBoss

Changing the ports and other configuration for the SIP connector can be done in the standalone-sip.xml file. Below is an extract :

Example 1. Adding a SIP Connector to $JBOSS_HOME/standalone/configuration/standalone-sip.xml

```
 <socket-binding-group name="standard-sockets" default-interface="public" port-
offset="${jboss.socket.binding.port-offset:0}">
        <socket-binding name="management-native" interface="management"
port="${jboss.management.native.port:9999}"/>
        <socket-binding name="management-http" interface="management"
port="${jboss.management.http.port:9990}"/>
        <socket-binding name="management-https" interface="management"
port="${jboss.management.https.port:9443}"/>
        <socket-binding name="ajp" port="8009"/>
        <socket-binding name="http" port="8080"/>
        <socket-binding name="https" port="8443"/>
        <socket-binding name="sip-udp" port="5080"/>
        <socket-binding name="sip-tcp" port="5080"/>
        <socket-binding name="sip-tls" port="5081"/>
    <socket-binding name="sip-ws" port="5082"/>
        <socket-binding name="osgi-http" interface="management" port="8090"/>
        <socket-binding name="remoting" port="4447"/>
        <socket-binding name="txn-recovery-environment" port="4712"/>
        <socket-binding name="txn-status-manager" port="4713"/>
        <outbound-socket-binding name="mail-smtp">
            <remote-destination host="localhost" port="25"/>
        </outbound-socket-binding>
    </socket-binding-group>
```

If you need to add a connector for the same protocol, a new socket-binding should be created. A naming convention should be followed for the name attribute of the new socket-binding. The convention is <name>-sip-<protocol>. By example,

```
    <socket-binding name="second-sip-udp" port="5080"/>
```

SIP <connector> Attributes

port (defined at the socket-binding element)

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

protocol

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

signalingTransport (use socket-binding element)

Specifies the transport on which the container will be able to receive SIP messages. Supported Values are "udp", "tcp", "tls" and "ws".

use-load-balancer

Enable to specify if that particular connector will be using the Load Balancer for outbound traffic. The Load Balancer to be used is defined by the next `load-balancer` attributes below

load-balancer-address

Specifies the Load Balancer address used to route outbound traffic for this SIP Connector.

load-balancer-rmi-port

Specifies the RMI Port used to connect to the Load Balancer for this SIP Connector. This enables the connector to advertise to the Load Balancer the IP and Port it is listening on so that the Load Balancer can route traffic to that connector as well.

load-balancer-sip-port

Specifies the SIP Port of the Load Balancer to use for outbound traffic for this SIP Connector. This enables the connector to choose to which Load Balancer it sends outbound traffic. This is particularly useful for supporting different Load Balancers over different network interfaces

use-stun

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 Restcomm SIP Servlets [_mssstun__stun] for more information about STUN.

stun-server-address

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 Restcomm SIP Servlets [_mssstun__stun] for more information about STUN and public STUN servers.

stun-server-port

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 Restcomm SIP Servlets [_mssstun__stun] for more information about STUN.

use-static-address

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

static-server-address

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.

static-server-port

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.

http-follow-sip

Makes the application server aware of how the SIP Load Balancers assign request affinity, and

stores this information in the application session.

For Tomcat

Changing the ports and other configuration for the SIP connector can be done in the server.xml file. Below is an extract.

Example 2. Adding a SIP Connector to $CATALINA_HOME/conf/server.xml

```
  <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".

useLoadBalancer

Enable to specify if that particular connector will be using the Load Balancer for outbound traffic. The Load Balancer to be used is defined by the next `loadBalancer` attributes below

loadBalancerAddress

Specifies the Load Balancer address used to route outbound traffic for this SIP Connector.

loadBalancerRmiPort

Specifies the RMI Port used to connect to the Load Balancer for this SIP Connector. This enables the connector to advertise to the Load Balancer the IP and Port it is listening on so that the Load Balancer can route traffic to that connector as well.

loadBalancerSipPort

Specifies the SIP Port of the Load Balancer to use for outbound traffic for this SIP Connector. This enables the connector to choose to which Load Balancer it sends outbound traffic. This is particularly useful for supporting different Load Balancers over different network interfaces

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 Restcomm SIP Servlets [_mssstun__stun] 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 Restcomm SIP Servlets [_mssstun__stun] 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 Restcomm SIP Servlets [_mssstun__stun] 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.

A comprehensive list of implementing classes for the SIP Stack is available from the Class SipStackImpl page on nist.gov.

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: Application Router Packaging and Deployment, Application Selection Process, and Appendix C.

.

See the example chapters for more information about the Application Router Configuration for SIP Restcomm SIP Servlets for JBoss AS7

[_sfss_services_for_sip_servlets]

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

Example 3. 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.SipApplicationD
ispatcherImpl"
 usePrettyEncoding="false"
 additionalParameterableHeaders="Header1,Header2"
 bypassResponseExecutor="false"
 bypassRequestExecutor="false"
 baseTimerInterval="500"
 t2Interval="4000"
 t4Interval="5000"
 timerDInterval="32000"
 dispatcherThreadPoolSize="4"
 darConfigurationFileLocation="file:///home/user/workspaces/sip-servlets/
 sip-servlets-examples/reinvite-demo/reinvite-dar.properties"
 sipStackPropertiesFile="conf/mss-sip-stack.properties"
 dialogPendingRequestChecking="false"
 callIdMaxLength="32"
 tagHashMaxLength="10"
 canceledTimerTasksPurgePeriod="1">
```

For Restcomm SIP Servlets for JBoss AS7 this is located in standalone-sip.xml file :

Example 4. Configuring the Mobicents SubSystem Element in the Container's standalone.xml

```
  <subsystem xmlns="urn:org.mobicents:sip-servlets-as7:1.0" application-
router="dars/mobicents-dar.properties" stack-properties="mss-sip-stack.properties"
path-name="gov.nist" app-dispatcher-
class="org.mobicents.servlet.sip.core.SipApplicationDispatcherImpl" concurrency-
control-mode="SipApplicationSession" congestion-control-interval="-1">
            <connector name="sip-udp" protocol="SIP/2.0" scheme="sip" socket-
binding="sip-udp"/>
            <connector name="sip-tcp" protocol="SIP/2.0" scheme="sip" socket-
binding="sip-tcp"/>
            <connector name="sip-tls" protocol="SIP/2.0" scheme="sip" socket-
binding="sip-tls"/>
        </subsystem>
```

SIP Service element attributes

className

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

sipApplicationDispatcherClassName (Tomcat) - app-dispatcher-class (JBoss/EAP)

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 (Tomcat) - application-router (JBoss/EAP)

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 (Tomcat) - stack-properties (JBoss/EAP)

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:

usePrettyEncoding (Tomcat) - use-pretty-encoding (JBoss/EAP)

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 (Tomcat) - additional-parameterable-headers (JBoss/EAP)

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.

baseTimerInterval (Tomcat) - base-timer-interval (JBoss/EAP)

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).

t2Interval (Tomcat) - t2-interval (JBoss/EAP)

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

t4Interval (Tomcat) - t4-interval (JBoss/EAP)

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

timerDInterval (Tomcat) - timerD-interval (JBoss/EAP)

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).

dialogPendingRequestChecking (Tomcat) - dialog-pending-request-checking (JBoss/EAP)

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.

callIdMaxLength (Tomcat) - call-id-max-length (JBoss/EAP)

This property allows to shorten the size of Call-ID Header. This is useful when integrating with Lync (which has a limit of 32 in size) or older SIP Servers

tagHashMaxLength (Tomcat) - tag-hash-max-length (JBoss/EAP)

This property allows to shorten the size of tags in From and To Header. This is useful when integrating with Lync (which has a limit of 10 in size) or older SIP Servers

dnsServerLocatorClass (Tomcat) - dns-server-locator-class (JBoss/EAP)

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.

dnsResolverClass (Tomcat) - dns-resolver-class (JBoss/EAP)

Specifies the `org.mobicents.javax.servlet.sip.dns.DNSResolver` 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.servlet.sip.dns.MobicentsDNSResolver`, but any class implementing the `org.mobicents.servlet.sip.dns.DNSResolver` interface. To disable DNS lookups, this attribute should be left empty.

addressResolverClass (Tomcat) - address-resolver-class (JBoss/EAP)

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.

canceledTimerTasksPurgePeriod (Tomcat) - canceled-timer-tasks-purge-period (JBoss/EAP)

Defines a period to due a purge in the container timer schedulers. The purge may prevent excessive memory usage for apps that cancel most of the timers it sets.

```
== SIP Servlets Server Logging
```

```
Logging is an important part of working with Restcomm SIP Servlets.
There are a few files that you need to be familiar with in order to successfully
troubleshoot and adapt Restcomm SIP Servlets server monitoring and logging to your
environment.
```

```
.Logging Files for Restcomm SIP Servlets for JBoss AS7
$JBOSS/standalone/configuration/logging.properties
```

```
$JBOSS/standalone/configuration/mss-sip-stack.properties
```

```
$JBOSS/standalone/configuration/standalone-sip.xml
```

```
.Setting the log file name in $JBOSS/standalone/configuration/standalone-sip.xml
====
[source,xml]
----
```

```
 </formatter>
<file relative-to="jboss.server.log.dir" path="server.log"/>
<suffix value=".yyyy-MM-dd"/>
<append value="true"/>
----
====
```

```
The configuration above produces SIP logs that can be found in the
$JBOSS_HOME/standalone/log directory.
Below is an extract of the log files.
```

```
----
```

```
                      server.log.2012-08-14  server.log.2012-08-24
server.log            server.log.2012-08-16  server.log.2012-08-25
server.log.2012-08-07  server.log.2012-08-21  server.log.2012-08-26
server.log.2012-08-13  server.log.2012-08-22
----
```

```
.Logging Files for Restcomm SIP Servlets for Tomcat
If you are working with Tomcat, the log configuration files are located in the
$CATALINA_HOME/conf/ directory.
The log4j configuration file is located in $CATALINA_HOME/lib/ directory
```

```
$CATALINA_HOME/conf/logging.properties
```

```
$CATALINA_HOME/conf/mss-sip-stack.properties
```

```
$CATALINA_HOME/conf/server.xml
```

```
$CATALINA_HOME/lib/log4j.xml
```

```
.Truncated Sample Configuration from Server.xml
.Setting the log file name $CATALINA_HOME/conf/server.xml
====
[source,xml]
----
```

```
<Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"
               prefix="localhost_access_log." suffix=".txt"
               pattern="%h %l %u %t &amp;quot;%r&amp;quot; %s %b" resolveHosts="false"/>
----
====
```

```
.Truncated Sample Configuration from log4j.xml
.Configuring the log file name $CATALINA_HOME/lib/log4j.xml
====
[source,xml]
----
```

```
<log4j:configuration xmlns:log4j="http://jakarta.apache.org/log4j/">
  <appender name="rolling-file" class="org.apache.log4j.RollingFileAppender">
    <param name="file" value="${catalina.home}/logs/sip-server.log"/>
    <param name="MaxFileSize" value="1000KB"/>
----
====
```

```
The result of the extracted configuration above that is taken from the log4j.xml file
and can be found in the $CATALINA_HOME/logs directory.
```

```
JAIN-SIP Stack Logging
```

```
There are two separate levels of logging:
```

- Logging at the container level, which can be configured using the *log4j.xml* or *standalone-sip.xml* configuration file seen above
- Logging of the JAIN SIP stack, which is configured through the container logging and the SIP stack properties themselves

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

```
To use LOG4J in JAIN SIP Stack in Tomcat, you need to define a category in
[path]_CATALINE_HOME/lib/jboss-log4j.xml_ and set it to `DEBUG`.
```

```
.Configuring the JAIN SIP Stack to log into the Tomcat Container's logs
====
[source,xml]
----
```

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

```
To use LOG4J in JAIN SIP Stack in JBoss, you need to define a logger in
[path]_JBOSS_HOME/standalone/configuration/standalone-sip.xml_ and set it to
`DEBUG`.
```

```
.Configuring the JAIN SIP Stack to log into the JBoss Container's logs
====
[source,xml]
----
```

```
                        <logger category="gov.nist">
                        <level name="DEBUG"/>
                    </logger>
----
====
```

```
For this category to be used in Restcomm SIP Servlets, you need to specify it in
[path]_JBOSS_HOME/standalone/configuration/mss-sip-stack.properties_ or
[path]_CATALINE_HOME/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.
```

```
====
**NOTE**
When using ``loadbalanceraddress`` algorithm for LB it is necessary to add the
following at ``mss-sip-stack.properties`` file for the keepalive mechanism:
```

```
----
org.mobicents.ha.javax.sip.LoadBalancerHeartBeatingServiceClassName=org.mobicents.h
a.javax.sip.MultiNetworkLoadBalancerHeartBeatingServiceImpl
----
```