Mobicents

1.20

User Guide

Douglas Silas

ISBN:

Publication date:

Mobicents is a highly scalable event-driven application server with a robust component model and fault-tolerant exec ution environment. Mobicents is the first and only Open Source Platform certified for JSLEE 1.0 compliance. It complements J2EE to enable the convergence of voice, video and data in next-generation intelligent applications. Web and SIP can be combined together to achieve more sophisticated and natural user experience.

Mobicents: User Guide

by Red Hat Documentation Group and Douglas Silas

1. Document Conventions

Certain words in this manual are represented in different fonts, styles, and weights. This highlighting indicates that the word is part of a specific category. The categories include the following:

`Courier font`
Courier font represents `commands`, `file names and paths`, and `prompts` .

When shown as below, it indicates computer output:

```
Desktop       about.html       logs      paulwesterberg.png
Mail          backupfiles      mail      reports
```

`bold Courier font`
Bold Courier font represents text that you are to type, such as: **`service jonas start`**

If you have to run a command as root, the root prompt (#) precedes the command:

```
# gconftool-2
```

`italic Courier font`
Italic Courier font represents a variable, such as an installation directory: *`install_dir`*`/bin/`

bold font
Bold font represents **application programs** and **text found on a graphical interface**.

When shown like this: **OK** , it indicates a button on a graphical application interface.

Additionally, the manual uses different strategies to draw your attention to pieces of information. In order of how critical the information is to you, these items are marked as follows:

Note

A note is typically information that you need to understand the behavior of the system.

Tip

A tip is typically an alternative way of performing a task.

Important

Important information is necessary, but possibly unexpected, such as a configuration change that will not persist after a reboot.

Caution

A caution indicates an act that would violate your support agreement, such as recompiling the kernel.

Warning

A warning indicates potential data loss, as may happen when tuning hardware for maximum performance.

2. We Need Feedback!

You should over ride this by creating your own local Feedback.xml file.

Introduction

1. The Mobicents Converged Application Server

Mobicents is a highly scalable event-driven application server with a robust component model and a fault-tolerant execution environment. Mobicents is the first and only Voice-over-Internet Protocol (VoIP) platform certified for Java Service Logic Execution Environment (JSLEE) 1.0 compliance. It complements the Java Platform Enterprise Edition (J2EE) to enable the convergence of voice, video and data in next-generation intelligent applications. With the Mobicents Converged Application Server, Web and Session Initiation Protocol (SIP) applications can be combined to achieve a more sophisticated and natural user experience.

In addition to telecommunications applications, Mobicents is suitable for a wide variety of problem domains demanding an Event Driven Architecture (EDA) for high-volume, low-latency signaling. Examples include financial trading, online gaming, Radio-Frequency Identification (RFID) sensor network integration) and distributed control.

Figure 1.1. The Mobicents execution environment shown together with various extenal resources and management interfaces

2. Service Building Blocks and Next-Generation Intelligent Netoworks

In the scope of telecommunications' Next Generation Intelligent Networks (NGINs), Mobicents fits in as a high-performance core engine for Service Delivery Platforms (SDPs) and IP Multimedia Subsystems (IMSes).

Mobicents enables the composition of Service Building Blocks (SBBs) such as call control, billing, user-provisioning, administration, and presence-sensing features. The JAIN SLEE [1] specification allows popular protocol stacks such as SIP to be plugged in as resource adapters. The SLEE Service Building Blocks have many similarities to Enterprise Java Beans (EJBs), and naturally accommodate integration with enterprise applications, the Web, Customer Relationship Management (CRM) and Service-Oriented Architecture (SOA) end points.

Out-of-the-box monitoring and management of Mobicents components is achieved via SLEE standard-based Java Management Extensions (JMXes) and Simple Network Management Profile (SNMP) interfaces.

[1] JAIN SLEE stands for Java API for Intelligent Network Service Logic and Execution Environment architecture. JSLEE (Java Service Logic Execution Environment) is a short synonym for JAIN SLEE.

Beyond telecommunications, Mobicents is applicable to a wider variety of problems demanding high-volume, low-latency signaling. Examples include financial trading, online gaming, RFID sensor network integration, and distributed control.

3. Mobicents Use Cases

The Mobicents Converged Application Server has a wide variety of use cases, including all of the following domains:

- *VoIP Call Control Services*
- *Instant Messaging Services*
- *Online Multi-Player Games*
- *Financial Day Trading*
- *First Responder Communication Networks*
- *Sensor Network Integration and RFID*

VoIP Call Control Services. Voice-over-Internet Protocol (VoIP) services are some of the primary use cases for Mobicents. Examples of VoIP services include call routing, forwarding, termination, voice mailbox, conferencing and user provisioning.

Instant Messaging Services.

Online Multi-Player Games. SLEE is a great platform for Massively Multi-Player (MMP games. The following are some of the principle requirements for MMP servers, provided by the leader of *Java.net Games community* [https://games.dev.java.net/nonav/index.html]:

1. *Worst-case* latencies from end-to-end in the back-end system, including all database operations, of no more than 100 ms.
2. A programming model that allows the game programmer to program simple persistent objects, with no more work than any other standard Java object, and have them execute on data events coming into the system. These objects must be "real objects" in the simulation sense of the term.
3. A programming model that is optimistically parallel while *appearing* to the programmer as a single-threaded event-driven model. Each event has to be ACID-transactional (Atomicity, Consistency, Isolation, Durability), and atomic unto itself. The programmer cannot be required to be aware in any way of multiple threads, database access, or locking. It must be inherently and transparently race-condition-proof and deadlock-proof.
4. It must scale to massive numbers of simultaneous users (5-to-6 figures) online simultaneously accessing the same database of objects.

5. It needs to provide failover, fault-tolerance and efficient load-balancing. The last is critical. Game applications are cost-sensitive. Without the ability to load-balance the potentially heavy loads of multiple apps over banks of low-cost computers used to maximal efficiency, the economic model falls apart.

If you are familiar with SLEE, then you have already noticed that these requirements closely track the fundamental SLEE design principles.

Here are some pointers to books, blogs and papers on the subject of Massive Multi-Player games:

- *Internet-Based Games by R. Young* [http://liquidnarrative.csc.ncsu.edu/pubs/phic.pdf]
- *Massively Multiplayer Game Development* [http://www.amazon.com/exec/obidos/tg/detail/-/1584502436/002-3357427-0961607?v=glance] by Thor Alexander
- *MMP Blog* [http://hardcodedgames.com/mmpgamedev/]
- *A Mobile Gaming Platform for the IMS* [http://www.ibr.cs.tu-bs.de/users/wellnitz/papers/misc/ims-netgames2004.pdf]
- *SIP based Game Server* [http://forum.nokia.com.cn/doc/SIPpresentation(24workshopHK)-new.pdf]

Financial Day Trading. Financial trading is a well-understood problem domain demanding highly-available, fast-response technology to serve billions of trading calls per day. Each trading call is a concise message identifying the trader, ticker and price. Day trading applications fit nicely on top of event-driven containers.

First Responder Communication Networks. We live in an unsafe world. Project P25 is focused on public safety communications digital radio interoperability. It is spurred on by growing concern which has driven many countries' governmentsincluding the US Federal Governmentto reorganize in order to create focused positions to address homeland security. The National Institute of Standards and Technology (NIST) is building a test system for Radio-Frequency subsystem interoperability Standard, which is based on these three protocols: the Session Initiation Protocol (SIP), the Service Discovery Protocol (SDP), and the Real-time Transport Protocol (RTP). The reference implementation and test system will be built as SLEE services.

Sensor Network Integration and RFID. Gartner predicts that sensor technologies will be part of our everyday life by the year 2015. Sensors will be everywhere: as RFID tags on consumer products, devices monitoring tire pressure, location-tracking tags carried by workers in sensitive or hazardous environments, etc. In addition, all enterprise activities will be monitored using enterprise tools connected to the network; by 2010 these, as embedded Internet devices, will represent 95% of all Internet-connected systems.

Pervasive sensor technologies will transform the objective of IT from providing support for fundamentally manual processes to automating the execution of tasks in response to a continuously-changing environment monitored by sensors. This requires a new kind of architecture capable of delivering *extreme* transaction processing.

Installation

1. Installing the Mobicents Binary Distribution

The Mobicents Converged Application Server runs on top of the JBoss Application Server. The Mobicents 1.20 binary distribution comes bundled with the JBoss Application Server version 4.2.2 GA.

Hardware Requirements for the Binary Distribution. Both the Mobicents Converged Application Server and the JBoss Application Server are 100% Java. Mobicents will run on the same hardware that the JBoss Application Server runs on.

Pre-Install Requirements for the Binary Distribution. You should ensure a couple of things before downloading and installing the Mobicents binary release:

- You must have sufficient disk space in order to install the Mobicents binary release. Once unzipped, version 1.20 of the Mobicents binary release requires about 230 MB of free disk space. Keep in mind that disk space requirements may change from release to release.
- A working installation of Java 1.5 or higher is required in order to run Mobicents and the JBoss Application Server.

Installing the Mobicents Binary. Once the preconditions and requirements have been met, you are ready to install the Mobicents binary distribution.

1. First, Download the latest version of the Mobicents binary distribution. The latest version can be found by going to the *Mobicents project page on SourceForge.net* [https://sourceforge.net/projects/mobicents/], clicking on *Download Mobicents*, and then looking for *Mobicents Server* and *Latest Version*.
2. Then verify the integrity of the zip file. Along with the zip file there is a sha1 file[1] which contains a checksum that can be used to verify the integrity of the zip file, and can alert you if the file has been changed since it was uploaded, or if it was corrupted upon download.

   On a Linux system you can use the `sha1sum` command to verify the integrity of the binary distribution zip file.

   On a Windows system you will need to download a sha1sum-generating program such as the `sha1sum.exe` program. This will generate a checksum that you can compare with the sum in the sha1 file.

   Note that if the two checksums are not identical, it means that the downloaded zip file was corrupted upon download, or has been changed since it was last uploaded to the server.

3. Finally, install the Mobicents Converged Application Server. Unzip the Mobicents binary distribution zip file to the location where you want it, thus completing the install.

2. Building the Mobicents 1.2.x Series

Besides simply installing the latest binary Mobicents release, you can also build the Mobicents Converged Application Server from source. There are several reasons for building Mobicents from source, including getting access to new features such as the latest resource adapters, bundled servers and examples, getting full support for debugging, and accessing extra documentation.

Note that building Mobicents from source requires a working installation of JBoss 4.2. This section on building Mobicents from source also includes the steps necessary for installing the JBoss 4.2.2 binary distribution.

Hardware Requirements for the Source Distribution. Both the Mobicents Converged Application Server and the JBoss Application Server are 100% Java. Mobicents will run on the same hardware that the JBoss Application Server runs on.

Pre-Install Requirements for the Source Distribution. You should ensure a few things before downloading and installing the JBoss binary distribution and the Mobicents source:

- You must have sufficient disk space in order to install the Mobicents source as well as the JBoss binary distribution. The Mobicents source, once fetched from the development Subversion repository, will consume about 230 MB of disk space. In addition to this, the JBoss Application Server binary distribution requires approximately 140 MB, meaning that you should probably have *at least* 400 MB of free disk space available. Keep in mind also that disk space requirements may change with subsequent updates to the development repository.
- A working installation of Java 1.5 or higher is required in order to run Mobicents and the JBoss Application Server.
- A way to download files. We will use the wget utility in the subsequent sections, but any common web browser will do.
- The Subversion version control system.
- The Apache Ant build tool. Mobicents includes this in its distribution, but we will need Ant in order to then build the Ant that comes bundled with the distribution.

Downloading and Installing the Jboss Binary Distribution. Once the pre-install requirements have been met, you are ready to start. We will split the entire procedure, which isn't long but may take a while depending on the speed of your Internet connection, into two parts: in the first we will install JBoss; in the second, we will download and build Mobicents.

Procedure 2.1. Downloading and Installing JBoss

1. First, move to the directory that you want to install JBoss into, and download the JBoss 4.2.2 General Availability (GA) binary distribution [2]:

```
cd <directory_to_install_jboss_into>
wget
 http://surfnet.dl.sourceforge.net/sourceforge/jboss/jboss-
4.2.2.GA.zip
```

2. Next, assuming you want to install JBoss in the same directory you downloaded the zip file distribution to, unzip it:

```
unzip jboss-4.2.2.GA.zip
```

3. Finally, we must set the `JBOSS_HOME` environment variable. The quick-and-easy way to set it for the current session, but which will not last beyond the next reboot, is to `export` the variable on the command line, and then to `echo` the variable to make sure that it contains the right directory. For example:

```
export JBOSS_HOME="${PWD}/jboss-4.2.2.GA"
echo $JBOSS_HOME
/home/my_name/jboss/jboss-4.2.2.GA
```

   If the `echo` command doesn't produce any output, make sure that you are in the directory that you unzipped the JBoss binary distribution zip file to, and that the name of that directory is correct in the `export` command (i.e., that the file name hasn't changed in some subtle way).

   Alternatively, you can set the `JBOSS_HOME` environment variable permanently in the startup file for your shell. If you are using the Bash shell, this would mean inserting the following line into your `~/.bashrc` file (and making sure that the pathname is absolute):

```
export JBOSS_HOME="<unzip_directory>/jboss-4.2.2.GA"
```

Now that the JBoss Application Server has been successfully installed, we will download the source to the Mobicents Converged Application Server with Subversion, and build it.

Downloading the Source and Building Mobicents.

1. First, move to the directory that you want to install Mobicents into, create a directory named `mobicents` into which you'll install everything, and then use Subversion to get the latest sources:

```
cd <directory_to_install_mobicents_source_into>
mkdir mobicents
svn co http://mobicents.googlecode.com/svn/trunk mobicents
```

2. Once Subversion has finished acquiring the latest revision of the sources, which may take a while, then we will then proceed to build Mobicents:

```
ant -f mobicents/core/build.xml
BUILD SUCCESSFUL
Total time: 6 seconds
```

TBD: Close with a sentence and direct the reader with a ulink to the Running Mobicents (which was installed from source) chapter. Issues that need to be addressed in this chapter include: installing GWT 1.3, which only comes in source for x86 (not version 1.4) and then building the management console.This chapter will have to be expanded with those instructions.

3. Installing EclipSLEE: The JAIN SLEE Eclipse Plugin

The Mobicents project's EclipSLEE Plugin for the Eclipse IDE enables and facilitates working with Mobicents and JAIN SLEE. In particular, EclipSLEE is designed to help in the creation of the following JAIN SLEE components:

- Profile Specifications
- Events
- Service Building Blocks (SBBs)
- Service XML Descriptors
- Deployable Units

With the EclipSLEE plugin, developers can construct complete services quickly and easily. The plug-in takes care of ensuring that the XML descriptors are correct, as well as creating skeleton Java classes to which developers can add service logic.

How to Install EclipSLEE Eclipse Plugin. EclipSLEE can be installed simply by adding the URL of the Update site. In Eclipse, click on **Help** # **Software Updates** # **Find and Install...**. The `Install/Update` dialog box will pop up. Choose the second option, **Search for new features to install**, and click next to be presented with the `Install` dialog. On the right-hand side, click the **New Remote Site...** button to be presented with the `New Update Site` dialog. For the Name field, enter **`EclipSLEE`**, and in the `URL` field, enter the URL of the remote site: **`http://people.redhat.com/vralev/eclipslee/update/`**. Press **OK** to return to the `Install` dialog. Check the just-added `EclipSLEE` site and click the **Finish**

button. Eclipse will then prompt you to read and accept the license agreement, and will also ask you whether you want to install all components. After following the instructions of Eclipse's plugin install wizard, the EclipSLEE plugin will be downloaded and installed (usually to the shared Eclipse plugins directory, unless you have directed Eclipse to install to another location). Restarting Eclipse is required in order to initialize the EclipSLEE plugin so that it can be used.

Running Mobicents

1. Running the Mobicents Application Server

Once installed, running Mobicents simply consists in changing to the `<topmost_mobicents_directory>/server/bin` directory and invoking `run.sh` (Linux) or `run.bat` (Windows) The precise instructions depend on which platform you are running Mobicents on.

Platform Mobicents is Running On

Linux

1. Change your working directory to the `<topmost_mobicents_directory>/server/bin` directory:

```
cd <topmost_mobicents_directory>/server/bin
```

2. Ensure that the `run.sh` file is executable. If it isn't, modify its permissions so that it is:

```
chmod +x run.sh
```

3. Finally, start the Mobicents Application Server:.

```
./run.sh
```

Note

The above command is a shortened version of this equivalent command:

```
./run.sh -c all -b 127.0.0.1
```

Windows

1. Change your working directory to the `<mobicents_folder>\server\bin` folder.

2. Start the Mobicents Application Server:

```
run.bat
```

Note

The above command is a shortened version of this equivalent command:

```
run.bat -c all -b 127.0.0.1
```

Working with the Mobicents Management Console

Once Mobicents is running, you can access the management console by pointing your browser to *http://localhost:8080/management-console/* . If you are working with a remote machine, replace **localhost** with the name of the remote machine's host, and **8080** with the correct port.

The management console screen has the following components:

- The Main Menu, on the left.
- The Current View, on the right.
- The Log Console, on the bottom.

Figure 4.1. The Mobicents Management Console, initial view

New views my be selected by clicking on the Main Menu items. The Log Console displays important and relevant notifications when operations are executed, such as error and status messages.

The initial management console view, which can also be accessed by clicking on SLEE at the top of the Main Menu, displays a status message indicating whether Mobicents is currently running or not, and provides controls to start, stop, and shut down the Mobicents Converged Application Server.

Deployable Units. This view can be selected by clicking on Deployable Units from the Main Menu, and shows a list of all deployable units that can be deployed with the Mobicents Converged Application Server.

Figure 4.2. The Deployable Units view

A tab bar at the top of the view allows one to browse, search for, and install deployable units.

Components. The next view on the Main Menu is the Components view, which displays a list of component types, such as services, SBBs, resource adapters, etc., and their current count. Clicking on a component type will cause the management console to show a list of components for that type. From there, you can click on the different components to see their details.

Figure 4.3. The `Compenents` view

`Services.` The `Services` view naturally shows the list of available services, their state, whether `ACTIVE` or `INACTIVE`, and all currently possible actions for that resource adapter, such as `activate`, `deactivate` or `remove`.

Figure 4.4. The `Services` view

The tab bar at the top of the `Services` view also allows you to control the usage parameters of services.

`Resource Adapters.` Choosing `Resources` from the `Main Menu` will put the currently-deployed resource adapters in view. There, you can see their name, type, vendor and version. Clicking on one of the resource adapters will provide further information such as the name of the deployable unit, its state, whether `ACTIVE` or `INACTIVE`, and the currenty possible actions for specific services, such as activating, deactivating, or removing them.

Figure 4.5. The `Resources` view

`Activity Contexts.` The Activity Contexts view can be displayed by clicking on `Activities` from the `Main Menu`, and shows the current activity contexts. The `TTL` field shows how much time the activity can remain idle before being marked for garbage collection.

Figure 4.6. The `Activities` view

Resource Adapters

1. Deploying and Undeploying Resource Adapters

1.1. Before You Begin: Resource Adapters and Deployable Unit Files

Resource adapters are JAIN SLEE components that can be deployed (i.e. “loaded” or “installed”) into the Mobicents Converged Application Server, and undeployed (“uninstalled”) from it. At the file level, resource adapters consist of specially-named files called deployable unit files. Two of these deployable unit files must be loaded/installed before the Mobicents Converged Application Server will deploy the resource adapter. The next few sections will teach you how to locate where resource adapters and their files live, and how to deploy them and undeploy them through several different, though equivalent ways: via the command line; by copying the correct files to a special directory, or removing them from it; or with the Mobicents Management Console.

Which of these various ways of accomplishing the same task is easier depends on you. If you are comfortable with the command line, or like to script behavior, then deploying via the command line will probably be the preferred way for you. If you would prefer to simply copy or delete some files with the file manager, you can do it that way. And for those who prefer graphical user interfaces (GUIs), the Management Console provides an interface to the Mobicents Converged Application Server, and can easily be used to accomplish such tasks both locally and remotely. However, no matter which method you eventually choose, it will be necessary to read the following explanatory paragraphs, on where to find the directories corresponding to the resource adapters you want to install, and on how to recognize deployable unit files, and on the correct order in which to install the deployable unit files, before reading any or all of the method sections.

Where to Find Resource Adapters in the Mobicents Installation. The files belonging to resource adapters, and which you will need to access in order to deploy and undeploy them, are located in `<topmost_mobicents_directory>/resources`. It is trivial to figure out which directory in `resources` corresponds to the resource adapter that you wish to deploy or undeploy. For example, the `sipra` directory holds the files necessary for running the SIP resource adapter, and the `xmppra` directory holds the XMPP and Google Talk resource adapter.

How to Recognize Deployable Unit Files. Deployable unit files are JAR (Java ARchive) files. There are two deployable unit files in each resource adapter's directory. The following two patterns describe the file names:

Deployable Unit File Name Conventions

`<resource_adapter_abbreviation>-ra-DU.jar`
...where *ra* stands for “resource adapter”, and *DU* stands for "Deployable Unit (file). For example, if we want to deploy the SIP resource adapter, and we are located in `<topmost_mobicents_directory>/resources/sipra`, then this deployable unit file will be named `sip-ra-DU.jar`.

`<resource_adapter_abbreviation>-ratype-DU.jar`
...where *ratype* stands intuitively for “resource adapter type”, and *DU* again for “Deployable Unit (file)”. The file name for the “ratype” deployable unit file for the SIP resource adapter is therefore `sip-ratype-DU.jar`.

How to Determine the Correct Order for Installing Deployable Unit Files. Deployable unit files must be installed in a certain order, starting with the *ratype* deployable unit file, whose file name has the form: `<resource_adapter_abbreviation>-ratype-DU.jar`.

The second (and last) deployable unit file you must install is the *ra* file: `<resource_adapter_abbreviation>-ra-DU.jar`.

For example, if you wanted to deploy the SIP resource adapter, then you would want to first install the deployable unit file named `<resource_adapter_abbreviation>-ratype-DU.jar`, followed by the `<resource_adapter_abbreviation>-ra-DU.jar` file.

1.2. Deploying Resource Adapters

Note

The introductory section titled *Section 1.1, “Before You Begin: Resource Adapters and Deployable Unit Files”* should be read before proceeding to one of the following sections.

Section 1.2.1, “Deploying Resource Adapters Via the Command Line”

Section 1.2.2, “Deploying Resource Adapters By Copying Deployable Unit Files”

Section 1.2.3, “Deploying Resource Adapters With the Management Console”

1.2.1. Deploying Resource Adapters Via the Command Line

Resource adapters can be deployed by simply issuing an ant task command from the topmost Mobicents directory. Make sure that the Mobicents Converged Application Server has been started, and then change to the topmost directory:

```
cd <topmost_mobicents_directory>
```

Run the ant build-and-deploy script, remembering to replace `<resource_adapter>` with the correct directory name (refer to *Where to Find Resource Adapters in the Mobicents Installation* if you are unsure):

```
tools/ant/bin/ant -f resources/<resource_adapter>/build.xml
 ra-deploy
```

For example, the directory name in `<topmost_mobicents_directory>/resources` corresponding to the JAIN SIP resource adapter is (currently) named `sipra` ("SIP Resource Adapter"). To deploy the SIP resource adapter, therefore, you would issue the following command:

```
tools/ant/bin/ant -f resources/sipra/build.xml ra-deploy
```

If the resource adapter builds successfully, then a message similar to the following will be written to standard output:

```
ra-activate:

ra-deploy:

BUILD SUCCESSFUL
Total time: 5 seconds
```

To undeploy a resource adapter via the command line, refer to *Section 1.3.1, "Undeploying Resource Adapters Via the Command Line"*.

1.2.2. Deploying Resource Adapters By Copying Deployable Unit Files

It is essential to have read the previous sections on locating resource adapter directories, recognizing deployable unit files, and determining the correct order for installing the files, before continuing. See *Section 1.1, "Before You Begin: Resource Adapters and Deployable Unit Files"*.

You can deploy resource adapters by simply copying deployable unit files to a special directory with your file manager, or by doing the same via the command line. These instructions assume that you are using the file manager. You can also copy the two necessary files simultaneously, or one-after-the-other in the correct order, which is the way we will do it.

Warning

Make sure that you *copy* the deployable unit files, instead of *moving* them. If you accidentally move them, then make sure you copy them

back to the correct resource adapter directory where they came from. If you accidentally move them and then delete them, you will probably have to reinstall Mobicents, so be careful to copy instead of move!

Open your file manager and move to the directory corresponding to the resource adapter you want to deploy. Find the `<resource_adapter_abbreviation>-ratype-DU.jar` file and copy it.

Copying the first deployable unit file, the ratype file, in order to deploy the SIP resource adatper.

Figure 5.1. Locating and Copying the First Deployable Unit File

Where you will paste it to depends on whether you installed the Mobicents binary release, or built Mobicents from source. If you installed the binary version, then the correct directory to copy the files into is `<topmost_mobicents_directory>/server/server/default/deploy` (remembering that the bundled JBoss installation that comes with the binary Mobicents release is located in `<topmost_mobicents_directory>/server/server/default`; you are actually copying the file into JBoss's `deploy` directory). If, on the other hand, you installed Mobicents and JBoss separately and set the `JBOSS_HOME` environment variable, then the correct directory to copy the file to is `$JBOSS_HOME/deploy`.

After copying the *ratype* deployable unit file, Mobicents will send a message like the following to standard output, indicating that the file was successfully installed.

```
19:59:03,651 INFO  [RaTypeDeployer] Added RA Type with id
 ResourceAdaptorTypeID[JAIN SIP#javax.sip#1.2]
19:59:03,683 INFO  [STDOUT] 19:59:03,682
 INFO  [DeploymentMBeanImpl] Deployable
 unit with URL
 file:/home/silas/Desktop/temp/apps/mobicents/mobicents-all-
1_2_0_BETA2/server/server/default/deploy/sip-ratype-DU.jar deployed
 as DeployableUnitID[0]
```

After successfully copying the `<resource_adapter_abbreviation>-ratype-DU.jar` file, proceed to copy the `<resource_adapter_abbreviation>-ra-DU.jar` file to the same directory. Copying the *ra* file (after the *ratype* file, of course) will lead to successful installation of the resource adapter, and the Mobicents Converged Application Server will output information similar to the following to standard output:

```
19:59:59,940 INFO  [STDOUT] 19:59:59,940 INFO
 [ResourceManagementMBeanImpl] Activated RA Entity SipRA
```

```
20:00:00,191 INFO  [STDOUT] 20:00:00,190 INFO
 [ResourceManagementMBeanImpl] Created Link between RA Entity SipRA
 and Name SipRA
```

The easier way is simply to select and copy both deployable unit files and paste them simultaneously into `<topmost_mobicents_directory>/server/server/default/deploy` (or `$JBOSS_HOME/deploy` if you installed JBoss separately). When you do it that way, Mobicents can figure out in which order it should install the files.

To undeploy a resource adapter by removing or deleting deployable unit files, refer to *Section 1.3.2, "Undeploying Resource Adapters By Removing Files"*.

1.2.3. Deploying Resource Adapters With the Management Console

The Mobicents Management Console provides a graphical way to deploy and undeploy resource adapters. This section requires that you are familiar with the Management Console; for an introduction to the Console, refer first to *Chapter 4, Working with the Mobicents Management Console*.

Important

The following instructions also assume that you have already read introductory sections on locating resource adapter directories, recognizing deployable unit files, and determining their correct order for installation. It is essential to read those sections before attempting to deploy a resource adapter with the Management Console. See *Section 1.1, "Before You Begin: Resource Adapters and Deployable Unit Files"*

Once the Mobicents Converged Application Server is running, point your browser to `http://localhost:8080/management-console/` to open the Management Console. Click on **Deployable Units** in the Main Menu; this will open the **Deployable Units** main view, where you will see a list of currently-deployed units under the **Browse** tab, which will read `No deployable unit found` at first. Two other tabs provide the possibility of searching for and installing deployable units. Click on the **Install** tab, and you will see a **Package file:** prompt and a text-entry field in which the name of the deployable unit file you want to install will be listed. *But first we must locate the correct deployable unit file, so click the **Browse** button to the right of the text-entry field.*

Figure 5.2. The Install tab of the Deployable Units view in the Management Console

Before continuing, make sure that you know which deployable unit file to install first by reading the section titled *How to Determine the Correct Order for Installing Deployable Unit Files*.

Once armed with this knowledge, click the **Browse** button to find and install the first deployable unit file, whose file name will appear in the text entry field, and then click the **Install** button. Then install the second deployable unit file. If you install the files in the incorrect order, a message similar to the following will appear in the **Log Console** at the bottom of the Management Console main view:

```
[ERROR] javax.slee.management.DeploymentException: Could
 not deploy: No DeployableUniDeploymentDescriptor descriptor
 (META-INF/deployable-unit.xml) was found in deployable...
```

If you receive this error message, try installing the other deployable unit file first. Installing the right files in the correct order will lead to `[INFO] Deployable unit installed` messages in the **Log Console**.

If you see the name of the resource adapter you wanted to install listed in the **Resources** view of the Management Console, then that resource adapter has been successfully deployed.

Figure 5.3. Successful Deployment of the SIP Resource Adapter with the Mobicents Management Console

To undeploy a resource adapter with the Management Console, refer to *Section 1.3.3, "Undeploying Resource Adapters With the Management Console"*.

1.3. Undeploying Resource Adapters

Note

The introductory section titled *Section 1.1, "Before You Begin: Resource Adapters and Deployable Unit Files"* should be read before proceeding to one of the following sections.

Section 1.3.1, "Undeploying Resource Adapters Via the Command Line"

Section 1.3.2, "Undeploying Resource Adapters By Removing Files"

Section 1.3.3, "Undeploying Resource Adapters With the Management Console"

1.3.1. Undeploying Resource Adapters Via the Command Line

It is as easy to undeploy resource adapters via the command line as it is to deploy them that way: by simply issuing an ant task command from the topmost Mobicents

directory, after the Mobicents Converged Application Server has been started, of course.

```
cd <topmost_mobicents_directory>
```

You will run the same ant script that you ran to build and deploy the resource adapter, but note that the last argument has changed from `ra-deploy` to `ra-undeploy`:

```
tools/ant/bin/ant -f resources/<resource_adapter>/build.xml
 ra-undeploy
```

For example, to undeploy the SIP resource adapter you would issue the following command:

```
tools/ant/bin/ant -f resources/sipra/build.xml ra-undeploy
```

Upon success, a message similar to the following will be written to standard output:

```
ra-uninstall:
     [echo] Uninstalling SipRA.
[slee-management] Apr 8, 2008 8:49:12 PM
 org.mobicents.ant.tasks.UninstallTask run
[slee-management] INFO: No response
[slee-management] Apr 8, 2008 8:49:12 PM
 org.mobicents.ant.tasks.UninstallTask run
[slee-management] INFO: No response

ra-undeploy:

BUILD SUCCESSFUL
Total time: 2 seconds
```

1.3.2. Undeploying Resource Adapters By Removing Files

If you deployed a resource adapter by copying the deployable unit files from the resource adapter's directory to the `<topmost_jboss_directory>/deploy` directory[1], then you can undeploy that same resource adapter merely by removing the two deployable unit files from the `deploy` directory. In the file manager or on the command line, simply go to `<topmost_jboss_directory>/deploy` and either move the correct two deployable unit files out of it, or delete both of them from it.

[1] The location of the `<topmost_jboss_directory>/deploy` directory depends on whether you installed the binary distribution of Mobicents, in which case it is `<topmost_mobicents_directory>/server/server/default/deploy`, or downloaded and installed JBoss yourself, in which case it is `$JBOSS_HOME/deploy`.

Don't Delete if You Didn't Copy!

If you choose to delete the two deployable unit files, *make sure* that you indeed *copied* them into the `deploy` directory, and that they both still exist in the directory of the resource adapter you want to undeploy!

Successfully removing or deleting both files from the `deploy` directory will result in a long string of messages sent to standard output, and ending with a line similar to: `17:48:10,705 INFO [STDOUT] 17:48:10,705 INFO [DeploymentMBeanImpl] Uninstalled DU with id DeployableUnitID[6]`. It is also possible to simply remove or delete only the `<resource_adapter_abbreviation>-ra-DU.jar` file and have Mobicents undeploy the resource adapter, but it is recommended to always remove or delete both files.

Leaving Resource Adapters Deployed

One convenient way to have the Mobicents Converged Application Server deploy the correct resource adapters every time you start the server is to leave the deployable unit files in the `<topmost_jboss_directory>/deploy` directory. Upon startup, Mobicents will notice that the two files are still there, and will automatically deploy the resource adapter.

1.3.3. Undeploying Resource Adapters With the Management Console

If you are looking to undeploy a resource adapter with the Management Console, then it is assumed that you have first read *Section 1.1, "Before You Begin: Resource Adapters and Deployable Unit Files"* and *Section 1.2.3, "Deploying Resource Adapters With the Management Console"*.

You can undeploy resource adapters from the **Deployable Units** view in the Management Console by uninstalling the two deployable unit files. Click on **Deployable Units** from the Main Menu, and then make sure that the **Browse** tab is selected at the top of the view. You will see a list of deployable unit files for the currently-installed resource adapters. Find the deployable unit file that corresponds to the `<resource_adapter_abbreviation>-ra-DU.jar` pattern, and click on the **Uninstall** link in the **Actions** column. This will uninstall the file, and a message similar to `[INFO] Deployable unit sip-ra-DU uninstalled` will appear in the **Console Log** below. After uninstalling the *ra* file, then uninstall the `<resource_adapter_abbreviation>-ratype-DU.jar` file, which will print a success message like the previous one to the **Console Log**.

If you try to uninstall the *ratype* deployable unit file before the *ra* file, then you will receive a message similar to the following one, alerting you to the mistake: `[ERROR] javax.slee.management.ManagementException: Exception removing deployable Unit`.

2. SIP Resource Adapter

The Current Mobicents JSIP Version 1.2 implementation. The Mobicents SIP Resource Adapter is a Mobicents sub-project that aims to create a high-performance SLEE extension for the Session Initiation Protocol (SIP).

The most recent version of the SIP Resource Adapter includes several enhancements which allow for more convenient development of SIP resources for a wide range of applications. The type name of this latest version is `JSIP v1.2`, the type vendor is `net.java.slee.sip` and the type version is `1.2`.

The JAIN SIP Resource Adapter Version 1.1. In Java Specification Request 22 (JSR-22), this resource adapter type name is denoted as `JAIN SIP`, and the type vendor as `javax.sip`. This resource adapter type version is 1.1. Mobicents provides a fully-compliant implementation of this specification.

2.1. Installing the JAIN SIP Resource Adapter

To install the JAIN SIP Resource Adapter, you must first acquire the latest version. Navigate to the *Mobicents Sourceforge project page* [https://sourceforge.net/projects/mobicents] and click on the *Download Mobicents* link. On the next page, click on the *SLEE SIP RA* link so that it expands to reveal the necessary jar file, which you can click to download[2] As of the time of this writing, the latest version is *1.2 RC1 - core rc2*.

2.2. Example: End-to-End SIP

2.3. Example: Wake-Up Call

2.4. Example: Third-Party Call Control

2.5. Example: Back-to-Back User Agent

3. The XMPP and Google Talk Resource Adapter

[2] As of the time this was written, the jar file was named `jsipv1.2ra-package-10-02-2007.jar`.

3.1. Example: Google Talk Bot

4. Asterisk Resource Adapter

4.1. Installing the Asterisk Resource Adapter

The Asterisk resource adapter can be installed by referring to *???*.

Remember to replace `<resource_adapter>` with the name of the directory in `<topmost_mobicents_directory>/resources` corresponding to the Asterisk resource adapter (currently this directory is named `asteriskra`, but this could change in the future).

5. Parlay and Parlay-X Resource Adapter

5.1. Installing the Parlay and Parlay-X Resource Adapter

The Parlay and Parlay-X resource adapter can be installed by referring to *???*.

Remember to replace `<resource_adapter>` with the name of the directory in `<topmost_mobicents_directory>/resources` corresponding to the Parlay and Parlay-X resource adapter (currently this directory is named `parlayxra`, but this could change in the future).

5.2. Example: Simple Call-Blocking

6. J2EE and CA Resource Adapter

7. Diameter Resource Adapter

7.1. Installing the Diameter Resource Adapter

The Diameter resource adapter can be installed by referring to *???*.

Remember to replace `<resource_adapter>` with the name of the directory in `<topmost_mobicents_directory>/resources` corresponding to the Diameter resource adapter (currently this directory is named `diameterra`, but this could change in the future).

8. Media Resource Adapter

8.1. Installing the Media Resource Adapter

The Media resource adapter can be installed by referring to ???.

Remember to replace `<resource_adapter>` with the name of the directory in `<topmost_mobicents_directory>/resources` corresponding to the Media resource adapter (currently this directory is named `media-ra` , but this could change in the future).

9. SMPP Resource Adapter

9.1. Installing the SMPP Resource Adapter

The SMPP resource adapter can be installed by referring to ???.

Remember to replace `<resource_adapter>` with the name of the directory in `<topmost_mobicents_directory>/resources` corresponding to the SMPP resource adapter (currently this directory is named `smppra` , but this could change in the future).

10. Media Gateway Resource Adapter

11. JCC INAP Resource Adapter

12. HTTP-Server Resource Adapter

12.1. Installing the HTTP-Server Resource Adapter

The HTTP-Server resource adapter can be installed by referring to ???.

Remember to replace `<resource_adapter>` with the name of the directory in `<topmost_mobicents_directory>/resources` corresponding to the HTTP-Server resource adapter (currently this directory is named `http-servlet-ra` , but this could change in the future).

13. HTTP-Client Resource Adapter

13.1. Installing the HTTP-Client Resource Adapter

The HTTP-Client resource adapter can be installed by referring to ???.

Remember to replace `<resource_adapter>` with the name of the directory in `<topmost_mobicents_directory>/resources` corresponding to the HTTP-Client

resource adapter (currently this directory is named *http-client-ra* , but this could change in the future).

14. Persistence Resource Adapter

14.1. Installing the Persistence Resource Adapter

The Persistence resource adapter can be installed by referring to *???*.

Remember to replace *<resource_adapter>* with the name of the directory in *<topmost_mobicents_directory>*/resources corresponding to the Persistence resource adapter (currently this directory is named *persistencera* , but this could change in the future).

15. XCAP Client Resource Adapter

15.1. Installing the XCAP Client Resource Adapter

The XCAP Client resource adapter can be installed by referring to *???*.

Remember to replace *<resource_adapter>* with the name of the directory in *<topmost_mobicents_directory>*/resources corresponding to the XCAP Client resource adapter (currently this directory is named *xcapclientra* , but this could change in the future).

16. Rules Resource Adapter

16.1. Installing the Rules Resource Adapter

The Rules resource adapter can be installed by referring to *???*.

Remember to replace *<resource_adapter>* with the name of the directory in *<topmost_mobicents_directory>*/resources corresponding to the Rules resource adapter (currently this directory is named *rulesra* , but this could change in the future).

17. TTS Resource Adapter

17.1. Installing the TTS Resource Adapter

The TTS resource adapter can be installed by referring to *???*.

Remember to replace *<resource_adapter>* with the name of the directory in *<topmost_mobicents_directory>*/resources corresponding to the TTS resource

adapter (currently this directory is named `ttsra` , but this could change in the future).

18. SAP Resource Adapter

19. Third-Party Resource Adapters

Other Examples

Mobicents Servers

1. Mobicents SIP Servlets

2. Mobicents Media Server

2.1. Overview of the Mobicents Media Server

The Reasoning and Need for Media Servers. With the continued progress of globalization, more corporations than ever before have workgroups spread across countries and continents across the world. To support and increase the productivity of remote and telecommuting workgroups, communications companies are considering more cost effective network solutions that combine voice, wireless, data and video functionality. Businesses like these expect that the services they select and eventually implement will have call quality comparable to conventional telephone service, and they expect those services to boost productivity and reduce overall communications costs. Acquiring these desired network services requires connections from the Internet and wireless and wireline networks to Public Switched Telephone Networks (PSTNs) using a flexible, robust, scalable and cost-effective media gateway. The ability of such gateways to reduce overall communications costs for dispersed workgroups forms the foundation for media services and servers.

Media Gateways Bridge Multiple Technologies. Today, all communications can be routed through computers. Widespread access to broadband Internet and the ubiquity of Internet Protocol (IP) enable the convergence of voice, data and video. Media gateways provide the ability to switch voice media between a network and its access point. Using Digital Subscriber Line (DSL) and fast-Internet cable technology, a media gateway converts, compresses and packetizes voice data for transmission back-and-forth across the Internet backbone for wireline and wireless phones. Media gateways sit at the intersection of the PSTNs and wireless or IP-based networks.

The Justification for Media Gateways for VoIP. Multiple market demands are pushing companies to converge all of their media services using media gateways with VoIP capabilities. Companies expect such a convergent architecture to:

Company Expectations of a Convergent Architecture

Lower Initial Costs
: Capital investment is decreased because low-cost commodity hardware can be used for multiple functions.

Lower Development Costs
: Open system hardware and software standards with well-defined applications mean lower costs, and Application Programmable Interfaces (APIs) accelerate development.

Handle Multiple Media Types
: Companies want VoIP solutions today, but also need to choose extensible solutions that will handle video in the near future.

Lower the Costs of Deployment and Maintenance
: Standardized, modular systems reduce training costs and maintenance while also improving uptime.

Enable Rapid Time-to-Market
: Early market entry hits the window of opportunity and maximizes revenue.

What Is the Mobicents Media Server? The Mobicents Media Gateway is an open source Media Server based on the Java Media Framework and aimed to:

- Deliver competitive, complete, best-of-breed media gateway functionality of the highest quality.
- Meet the demands of converged wireless and wireline networks, DSL and cable broadband access, and fixed-mobile converged VoIP networks from a singleand singularly-capablemedia gateway platform.
- Increase flexibility with a media gateway that supports a wide variety of call control protocols and scales to meet the demands of enterprises and small-carrier providers.

2.2. Media Server Architecture

Media services have played an important role in the traditional Time Division Multiplexing (TDM)-based telephone network. As the network migrates to an Internet Protocol (IP)-based environment, media services are also moving to new environments.

One of the most exciting trends is the emergence and adoption of complementary modular standards that leverage the Internet to enable media services to be developed, deployed and updated more rapidly than before in a network architecture that supports the two concepts we will call provisioning-on-demand and scaling-on-demand.

General Design Overview. Mobicents Media Server is developed on top of existing Java technologies. The Java platform is the ideal platform for network computing. It offers a single, unifying programming model that can connect all elements of a business infrastructure. The modularization effort is supported by the use of the Java Management Extension (JMX) API, and the industry-standard Service Logic Execution Environment (SLEE) container. Using JMX enables easy

management of both the server's media components and the control modules hosted by SLEE.

This high degree of modularity benefits the application developer in several ways. The already-tight code can be further trimmed down to support applications that must have small footprints. For example, if PSTN (Public Switched Telephone Network) interconnection is unnecessary in your application, then simply take the D-channel feature out of the server. If you later decide to deploy the same application within a Signaling System 7 (SS7) network, then simply enable the appropriate endpoint. Another example is the freedom you have to drop your favorite media control protocol directly into the SLEE container.

Figure 7.1. Media Server Overview

The components that are involved in media processing or endpoints are implemented with JMF, the Java Media Framework. JMF provides a platform-neutral framework for displaying time-sensitive media. The JMF API:

- Scales across different protocols and delivery mechanisms
- Scales across different types of media data
- Provides an event model for asynchnronous communication between JMF and applications or applets

The JMF architecture allows advanced developers nto create and integrate new types of controllers and data sources. The JMF API declares abstract interfaces which are used for handling new media sources and sinks such as audio and video files, PSTN cards, webcams, etc.

The Media Server architecture assumes that call control intelligence is outside of the Media Server and handled by an external entity. The Mebia Server assumes that call controllers will use control procedures such as MGCP, Mecago or MSML, among others. Each specific control module can be plugged in directly to the server as a standard SLEE deployable unit. The usage of a SLEE container for the implementation of the control protocol-specific communication logic provides simple deployment and also the easy deployment of such control modules. The developer will not have to mess with low-level transaction and state management details, multi-threading, connection-pooling and other similar complex, low-level APIs.

Note

The Mobicents Media Server uses SLEE for implementing its own communication capabilities. The SLEE container doesn't serve here as a call controller.

In addition to control protocol modules, the SLEE container is aimed at providing high-level features like Interactive Voice Response (IVR) and Drools or VoiceXML engines.

The mobdules deployed under SLEE control interact with the Media Server Service Provider Interface (SPI) through the Media Server Control Resource Adapter, or MSC-RA. The MSC-RA follows the recommendations of JSR-309 and implements asynchronous interconnection with the Media Server SPI stack. This local implementation is restricted and does not all the use of high-level abstractions like VoiceXML dialogs, etc..

Typical Deployment Scenario. Mobicents Media Server offers a complete solution for all aspects of being a media gateway and server. This includes the Digital Signal Processors required to covert and compress TDB voice circuits into IP packets, announcement access points, conferencing, high-level IVR engines, etc. The gateway is able to provide signaling conversation and can operate as a Session Border Controller at the boundaries of Local Access Networks. The Media Server is always controlled by an external JBCP application server which implements the call control logic.

Endpoints. It is convenient to consider a media gateway as a collection of endpoints. An endpoint is a logical representation of a physical entity such as an analog phone or a channel in a trunk. Endpoints are sources or sinks of data and can be physical or virtual. Physical endpoint creation requires hardware installation whil esoftware is sufficient for creating a virtual endpoint. An interface on a gateway that terminates a trunk connected to a PSTN switch is an example of a physical endpoint. An audio source in an audio content server is an example of a virtual endpoint.

The type of the endpoint determines its functionality. Our analysis, so far, has led us to isolate the following basic endpoint types:

- Digital Signal 0 (DS0)
- Analog line
- Announcement server access point
- Conference bridge acces point
- Packet relay
- Asynchronous Transfer Mode (ATM) "trunk side" interface

This list is not exhaustive: there may be other types of endpoints defined in the future, for example test endpoints that could be used to check network quality, or frame-relay endpoints that could be used to manage audio channels multiplexed over a frame-relay virtual circuit.

Description of Various Access Point Types

Announcement Server Access Point
: An announcement server endpoint naturally provides access to an announcement server. Upon receiving requests from the call agent, the announcement server will "play" a specified announcement. A given announcement endpoint is not expected to support more than one connection at a time. Connections to an announcement server are typically one-way, or "half-duplex":the announcement server is not expected to listen to the audio signals from the connection.

Interactive Voice Response Access Point
: An Interactive Voice Repsonse (IVR) endpoint provides access to an IVR service. Upon requests from the call agent, the IVR server will "play" announcements and tones, and will "listen" to responses, such as (DTMF) input or voice messages, from the user. A given IVR endpoint is not expected to support more than one connection at a time.

Conference Bridge Access Point
: A conference bridge endpoint is used to provide access to a specific conference. Media gateways should be able to establish several connections between the endpoint and the packet networks, or between the endpoint and other endpoints in the same gateway. The signals originating from these connections shall be mixed according to the connection "mode" (as specified later in this document). The precise number of connections that an endpoint supports is characteristic of the gateway, and may in fact vary according to the alloction of resources within the gateway.

Packet Relay
: A packet relay endpoint is a specific form of conference bridge that typically only supports two connections. Packet relays can be found in firewalls between a protected and an open network, or in transcoding servers used to provide interoperation between incompatible gateways, such as gateways which don't support compatible compression algorithms, or gateways that operate over different transmission networks such as IP and ATM.

Signal Generators (SGs) and Signal Detectors (SDs). This endpoint contains a set of resources which provide the media-processing functionality. It manages the interconnection of media streams between the resources, and arbitrates the flow of media stream data between them. Media servicesor commands are invoked by a client application on the endpoint; that endpoint causes the resources to perform the desired services, and directs events sent by the resources to the appropriate client. The resources in the endpoint include a primary resource and zero or more secondary resources. The primary resource is typically connected to an external media stream, and provides the data from that stream to the secondary resources. The secondary resources may process that stream (e.g. by recording it and/or

performing automatic speech recognition on it), or may themselves generate generate media stream data (e.g. by playing a voice file) which is then transmitted to the primary resource.

Figure 7.2. tbd

A resource is statically prepared if the preparation takes place at the time of creation. A resource is dynamically prepared if preparation of a particular resource (and its associated media streams) does not occur until it is required by a media operation. Static preparation can lead to less efficient usage of a server's resources, because resources may tend to be allocated for a longer time before use. However, once a resource has been prepared, it is guaranteed to be available for use. Dynamic preparation may utilize resources more efficiently because of Just-In-Time (JIT) allocation algorithms may be used.

An endpoint is divided logically into a Service Provider Interface (SPI) that is used to implement specific a endpoint, and a management interface which is used for implementing manageable resources of that endpoint. All endpoints are plugged into the Mobicents SLEE server by registering with the MBean server. The kernel in that sense is only an aggregator, and not a source of actual functionality. The functionality is provided by the SPI implementation of the Mbeans, and in fact, all major endpoints are manageable MBeans interconnected through the MBean server. The best way to add endpoints to a Media Server is to write a new JMX bean which provides implementation of the endpoint's SPI.

The SPI layer is an abstraction that endpoint providers must implement to enable their media-processing features. An implementation of SPI for an endpoint is referred to as an *Endpoint Provider*.

Figure 7.3. EndpointManagementMBean UML diagram

3. Mobicents XML Document Manager Server

3.1. Introduction to the XML Document Manager Server

Mobicents XDM is a free and open source implementation of the XML Document Management server as defined in Open Mobile Alliance specifications. This functional element of next-generation IP communications networks is responsible for handling the management of user XML documents stored on the network side, such as presence authorization rules, static presence information, contact and group lists (a.k.a. resource lists), policy data, and many others.

3.2. XDM Server Architecture

The XML Document Manager Server runs on top of the Mobicents Converged Application Server and is also dependent upon the XCAP (XML Configuration Access Protocol) and SIP resource adapters.

Architecturally, the XDM Server is essentially an XCAP server that employs a SIP interface, which allows the server to learn about document updates.

The XDM Server can also use an element named the Aggregation Proxy, which is responsible for user authentication and result-caching on XCAP requests.

An XDM Server abstracts the element that stores XML documents as a data source.

Figure 7.4. The Architecture of the Mobicents XML Document Management Server

Figure 7.5. The JAIN SLEE Architectural Implementation of the XDM Server

Appendix A. Revision History

Revision History

Revision 1.0	Mon Feb 19 2008	Douglas Silas
Initial Release		