JADE ANDROID ADD-ON GUIDE

USAGE RESTRICTED ACCORDING TO LICENSE AGREEMENT.

Last update: 14-July-2010. JADE 4.0

Authors: Danilo Gotta (Telecom Italia), Tiziana Trucco (Telecom Italia), Marco Ughetti (Telecom Italia), Stefano Semeria (Reply Cluster), Cristina Cucè (Univ. Reggio Calabria), Anna Maria Porcino (Univ. Reggio Calabria)

TABLE OF CONTENTS

1 INTRODUCTION

This document describes the rationale behind the Jade Android Add-On and how to install, configure and use it.

JADE ANDROID (version 1.2) is a JADE add-on that provides support for using JADE-LEAP on Android Platform. Android is the software stack for mobile devices including the operating system released by Google within the Open Handset Alliance (Refer to web sites: http://code.google.com/android/ and http://www.openhandsetalliance.com).

The main body of this guide focuses on the process of starting and configuring the JADE ANDROID add-on in order to connect to a running JADE-LEAP platform and to start a split Container with a Jade Agent on Android SDK. A dummy application (Dummy Agent) is also provided. Please refer to the API documentation for a complete description of the API that allow programmatic access to all features.

All bugs, issues and feature requests should be made to the main JADE bug reporting system, or sent to the standard JADE mailing lists.

Version 1.2 of the JADE ANDROID add-on was developed by the JADE Team and is only guaranteed to work with JADE-LEAP release 4.0[1] or later and the Android Platform 1.5, 1.6, 2.1 and 2.2. releases.

The previous JadeAndroid release (1.1) has been tested also on the first Android phone, the HTC G1.

1.1 Target Audience

This document is intended for JADE users who are interested in using JADE on Android SDK and for ANDROID SDK users that are interested in agent development for peer to peer and social applications.

The reader is assumed to be familiar with JADE. People new to JADE are strongly recommended to first read the JADE Programming Tutorial available on the JADE web site (http://jade.tilab.com) first.

1.2 Rationale

Java, and in particular the MIDP profile, is almost a de-facto standard for low cost mobile phones applications, particularly games. However the SandBox model sets a number of limitations when trying to develop applications that require access to device related resources (e.g. Contacts, file system, incoming call, etc.) or user appealing and responsive GUI.

Moreover it is very likely that mobile applications in the coming years will require to manage other kind of resources, such as GPS for localization issues, RFID tag reader to exchange information with physical tagged objects and a variety of additional services like search engines, maps, social networks and so on.

[1] Actually the ANDROID-JADE add-on and the Dummy Agent demo do work with the JADE_LEAP subversion snapshot 6021 available since 25 Feb 2008.

The ANDROID SDK, released by the Open Handset Alliance at November 2007, provides an open platform with Linux based OS and JAVA based programming language and appears to be a good candidate to overcome the limitations mentioned above. The completely new Dalvik JVM, allows accessing all core functionality of the mobile device. In particular on top of a JavaSE-like platform it provides a set of new ANDROID specific API by means of which it is possible to interact with the ANDROID Operating System, control the device hardware and develop ANDROID GUI.

The possibility of combining the expressiveness of FIPA (www.fipa.org) communication supported by JADE agents with the power of the ANDROID platform brings, in our opinion, a strong value in the development of innovative applications based on social models and peer-to-peer paradigms. The JADE-ANDROID add-on has been designed to support this kind of applications. By means of it an ANDROID application can easily embed a JADE agent and therefore become part of a wider distributed system possibly including other mobile devices (not necessarily ANDROID enabled). More in details the add-on provides an interface that allows the application to start a local agent, trigger behaviours and more in general exchange objects with it. It is therefore possible discover remote peers, carry out possibly complex conversations with them, exploit the JADE ontology support to handle structured messages, perform background activities according to the behaviour composition model and in general take advantage of all features of the JADE platform.

In order to be compatible with the Dalvik JVM and to properly cope with the limitations and constraints of mobile devices and wireless networks, the add-on makes use of the JADE-LEAP version for JAVA CDC (or Personal Java) and the split-container execution mode (see Figure 1). This allows limiting communications over the wireless link as much as possible and obtaining very fast communication between mobile peers.

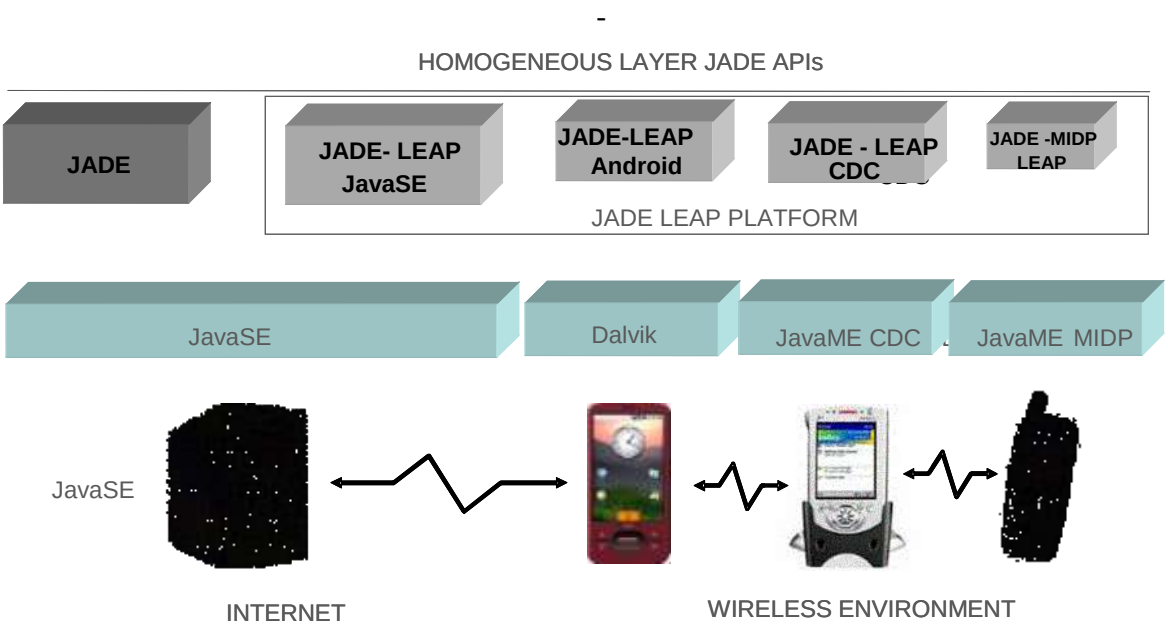

Figure 1: Jade and the Java World

1.3 Current limitations

Version 1.1 of the Jade Android add-on has the following known limitations:

- The Jade ANDROID add-on doesn't support more than one agent running on the split container started on ANDROID SDK.
- The Jade ANDROID add-on supports the management of the Jade Runtime only as a local android service.
- The Demo Application DummyAgent works well only with the HVGA-L skin of ANDROID SDK Emulator

All issues regarding the JADE ANDROID should be addressed to the jade-develop mailing list.

2 JADE ANDROID FILES

The Jade ANDROID Add-On comes with the following directory structure:

- doc: containing this document and the API documentation;
- lib: containing the JadeLeapAndroid.jar file that includes JADE ANDROID add-on and JADE LEAP compiled class;
- src: containing the following files:
    - source files of JadeLeapAndroid.jar not already included in Jade Leap distribution;
- demo: containing a full android project with the DummyAgent application using the JadeLeapAndroid.jar as a library;

3 HOW TO USE THE JADE ANDROID

This section provides instructions of how to set, configure and use the JADE ANDROID add-on. These instructions are related to configure and compile JADE-LEAP for the Android Environment.

3.1 Getting the software

In order to compile JADE-LEAP for a the Android environment it's necessary to download the JADE sources from the "Download" area of the JADE web site, the LEAP add-on and the Jade Android add-on from the "Add-ons" area of the JADE web site
The LEAP and the Jade Android add-ons must be unzipped in the JADE root directory. Once this has been done your JADE directory structure should look like:

```
jade/
 |
 |-add-ons/
     |- …
     |-jade4android
          |- demo
          |- doc
          |- lib
          |- src

 |-leap/
 |    |- …
 |    |-src/ includes the leap add-on source files
 |- …
 |-src/ includes the JADE source files
```

Note that, unlike other add-ons that are un-packaged under the `jade/add-ons/` directory, the LEAP add-on is un-packaged directly under the jade root directory.
We will refer to the `jade/add-ons/jade4android` directory as the "Jade Android root directory" and we will indicate it simply as `android/`.

3.2 Building and Configuring

3.2.1 Preliminary Steps

As for JADE, building JADE-LEAP for Android can be done using the program '**ant**' (version **1.6** or later), a platform-independent version of make. Ant uses the file '`build.xml`', which contains all the information about the files that have to be compiled, and that is located into the android root directory. The 'ant' program must be installed on your computer, and can be freely downloaded from the Jakarta Project at the Apache web site: http://ant.apache.org.

Using ant requires you to set the following environment variables (see the ant documentation for details).

- JAVA_HOME must point to your JDK1.5 or later.
- ANT_HOME must point to where you installed ant.

In order to compile the add-on and the demo, the Android SDK and an Android platform must be installed on your computer. Please refer to the Android web site for detailed instruction on how to install the Android SDK and the desired Android platform (http://developer.android.com/index.html).

3.2.2 Build.properties file

Before building JADE-LEAP for Android it is typically necessary to edit the `build.properties` file included in the Jade Android root directory. This property file includes the properties that depend on the local environment such as

- `Android-sdk-folder` property to point to the directory where the Android SDK is installed.
- `android-target` property specifies the target Android Platform to be used: android-3 for Android Platform 1.5, android-4 for Android Platform 1.6, android-7 for Android Platform 2.1 and android-8 for Android Platform 2.2.
- `jade-home-dir` property to point to the directory where JADE is installed.

3.2.3 Building Jade Android add-on

To build JADE-LEAP for Android, go in the Jade Android root directory and type

```
ant jar
```

3.3 Using Jade Android add-on

The JadeLeapAndroid.jar shall be included in your Android Project in order to add to your Android Application the capability to connect to a remote JADE LEAP platform and to start a split container with a running Jade Agent on ANDROID emulator.

The JADE LEAP ANDROID includes two main java objects:

3.3.1 MicroRuntimeService

jade.android.MicroRuntimeService is an android Service (as a matter of fact it extends android.app.Service) that is responsible for configure the Jade Environment and to start or stop the Jade Runtime when required. In order to make this Service communicating with other android components, the Binder mechanism has been used.

Note that this class SHALL NOT be directly used by the Android Activities of your Android applications but SHALL be only accessed through the Jade Gateway.

3.3.2 JadeGateway

jade.android.JadeGateway is the most important class that an android application developer shall be aware of in order to add jade feature to his android application.

This class provide a gateway between ANDROID applications and a JADE based multi agent system. It maintains an internal JADE Agent of class GatewayAgent that acts as an entry point in the JADE based system. The activation/termination of this agent (and its underlying split container) are completely managed by the JadeGateway class and the android application developers do not need to care about them.

This class provides APIs very similar to the class `jade.wrapper.gateway.JadeGateway` (refer to Jade documentation); the main difference is that this one uses the `jade.core.MicroRuntime` instead of `jade.core.Runtime` in order to start a container in a split mode that is better suitable in a mobile environment.

In the following there is a very short description of its APIs, refer to Javadoc in order to get further details.

It provides the following methods:

```
static connect(String agentClassName,
               String[] agentArgs,
               Properties jadeProfile,
               Context ctn,
               ConnectionListener listener)
```

in order to give to the Jade Environment all configuration data, to start (if necessary) and to bind the activity to the service. The binding procedure produces a call of the `onConnected(JadeGateway gateway)` method of the ConnectionListener interface that returns an instance of the JadeGateway that the activity shall use hereafter to communicate with Jade environment.

`agentClassName` is the fully qualified name of the class that extends jade.android.GatewayAgent and implements the application specific jade agent.

`agentArgs` are the args passed to the jade agent when the Jade Runtime starts it. They can be null.

`jadeProfile` is the set of jade properties to be passed to the Jade Runtime. The minimal set shall be passed to this method is:

`jade.core.Profile.MAIN_HOST` is the hostname or ip address of the Jade Platform Main Container.

`jade.core.Profile.MAIN_PORT` is the port of the Jade Platform Main Container.

`jade.imtp.leap.JICP.JICPProtocol.MSISDN_KEY` is the unique identifier of the split Container and Jade agent that are started on Android Emulator. Typically you can use the IMEI code.

`disconnect(Context ctn)` in order to unbind to the Service. Calling this method produces does not affect the life cycle of Jade Runtime.

`execute(Object command)` in order to start the jade split container and the jade agent (if needed) and to send a command to the jade agent through the `putO2Aobject` mechanism (refer to the Jade Programmer Guide and Jade API). This method behaves like the execute method of class `jade.wrapper.gateway.JadeGateway`, see the Javadoc of this class in order to get further details.

WARNING: In the current version of the JADE ANDROID addon if the MicroRuntime.startJade method (called by JadeGateway.execute()) fails because the Jade Main Container is not reachable, the execute methos will take long time to complete blocking the ANDROID Application.

3.3.3 Short Guideline for ANDROID add-on users

The following are the main steps in order to use the JadeGateway in your Android Application:

1) Create your Jade Agent class extending `jade.wrapper.gateway.GatewayAgent;`

2) Add your application specific behaviour to your Jade Agent and override the `processCommand` method with your application specific implementation;

3) Create an android activity that implements ConnectionListener;

4) Call the `JadeGateway.connect` method in the `onCreate` method of your activity passing all required Jade properties;

5) Implement the `onConnected(JadeGayeway gateway)` method of the ConnectionListener interface in order to get the JadeGateway instance;

6) Call the `execute` method of the JadeGateway instance whenever you shall send a command to the agent;

7) Call the `disconnect` method in order to unbind to the service.

Note that the steps from 3 to 7 shall be performed for each activity of your android application that needs to communicate with the MicroRuntimeService through the JadeGateway.

3.3.4 Editing the Android Manifest

To be able to use Android add-on functionality you have to edit the AndroidManifest.xml inside the project of your Android application. In particular you have to add a service tag inside

the manifest application tag. This is because the Jade Android add-on uses an Android service (called MicroRuntimeService) that must be declared in the application manifest to be used.

For example:

```
…
<application android:icon="@drawable/icon">

    <service android:name="jade.android.MicroRuntimeService" />

    <activity android:label="@string/app_name"
    android:name=".YourApplicationActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
              <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
     </activity>

   …

</application>

…
```

Please note that without this modification you will get an error from android runtime, complaining that the MicroRuntimeService cannot be found.

3.4 Logging on Jade Android add-on

The Jade Android add-on provides the `jade.util.Logger` class to produce printouts "the Jade way". The Android add-on logger implements the same interface as the standard jade.util.Logger class and uses the Android Log class to perform logging.

Basically it provides a wrapper to the Android log mechanism which will result familiar to Jade programmers.

Probably the most used method of the logger class is:

```
void log(int level,
         String message)
```

`level` is desired logging level.
`message` is the message that user wants to log.

3.4.1 Format of the Jade Android logs using Logger.log method.

The default behaviour of the jade android logger is to produce logs printed on the android console that can be retrieved using the *logcat* tool.

The logs have the following format:

- A letter indicating the Android log level (which are: **V**erbose, **I**nform, **W**arning, **E**rror, **D**ebug). Please note that Android has only five log levels while Jade uses 7 log levels, so a mapping between levels was necessary. Levels Logger.FINE, Logger.FINER e Logger.FINEST have been mapped on Log.DEBUG level.
- A tag that is always Jade.util.Logger;
- The process PID;
- A *name* passed to Logger constructor (usually the name of class that creates the logger);
- A custom user *message.*

As an example, the following code:

```
class ExampleLog {

      private final Logger myLogger =
  Logger.getMyLogger(this.getClass().getName());

    public void writeLog(){

        myLogger.log(Logger.INFO, "this is a message log");

     }

  }
```

produces this:

```
I/jade.util.Logger(  695): ExampleLog: this is a message log
```

For a detailed explanation of the other methods please see Android add on API Javadoc.

4 THE DUMMY AGENT DEMO

4.1 Building and Configuring the demo

This demo requires a JADE main container running on a PC (refer to Jade Programming Guide for How to start a Jade Main Container)

Before building Dummy Agent DEMO for Android it is typically necessary to edit the string.xml file in demo/res/values and change the values of the keys host and port to the right <ip address> and <port> of the running jade main container.

WARNING: Compiling the DummyAgent demo for Android Platform 1.5 and 1.6,, ant build shall eventually raise an OutOfMemoryException when converting a large amount of classes.

This seems to be a limitation of the dx tool: the utility that converts Java bytecode to the Dalvik JVM format. The easiest workaround for this issue is to modify the dx.bat file that can be found in <ANDROID_SDK_FOLDER>platform/<android-target>/tools, increasing the heap size for the JVM,adding the line: `set javaOpts=-Xmx1024M`**.**

Please note that exact values for minimum and maximum heap size depend on the amount of memory available on your machine.

To build Dummy Agent DEMO for Android, go in the Jade Android root directory and type

`ant demo` after having performed the steps specified on paragraph 3.2

4.2 Installing DummyAgent DEMO on ANDROID SDK emulator

1) Start the ANDROID emulator

2) Go to jade4android/demo/bin and type:

```
adb install DummyAgent-debug.apk
```

4.3 Starting a Jade Leap Main Container

For example typing:

```
java –cp j2se\lib\JadeLeap.jar jade.Boot -gui
```

For further details refer to the JADE Programming Guide.

4.4 Starting the DummyAgent app on emulator

From the home screen of the Android Emulator:

1) Push the All button (Figure 2a)
2) Push the Jade icon of DummyAgent application (Figure 2b)

a) b)

Figure 2: ANDROID SDK Emulator: Home Application

On the RMA GUI of the Jade Platform you should see a new Container with an agent inside (called *android* from now on). (Figure 3)

On the ANDROID Emulator you should see the screen of the DummyAgentActivity (Figure 3).

If the connection to the platform is successful a notification appears on the Android status bar.

Figure 3: ANDROID SDK Emulator: DummyAgent Application connected to Jade Platform

4.5 The Dummy Agent DEMO User Interface

The main screen shows two tabs: Send Msg and Msg List.

(a) (b)

Figure 4: Android Add-On Demo GUI

The first one (shown in Figure 4a) allows sending messages to other agents defining the following attributes: Sender Name, Receiver Name, Communicative Act, which specifies the message type, and Content.

Figure 5: receiver list context menu

In particular, if user clicks on the first item of the receiver list and keep selecting it for a while, he will see a context menu, as shown in Figure 5, by which he can decide to add, edit and remove one or all the receivers. Please note that to be able to see the menu, the user must select an entry: also when the list is empty the first entry remains always selectable.

Figure 6: AID Dialog

By choosing the *Add* or *Edit* item, the user can see the screen shown in Figure 6, in which he can set the name and address of a new agent or edit an existing one. By clicking the checkbox, the user shall be able to enter a local name (i.e. without the @<*platform id*> suffix).

The other Tab (shown in Figure 4b) element contains the full list of all sent and received messages. The user can click on a message to see its details, as showed in Figure 7a, or on Clear List Button to clean the list. In the message details screen it is also possible to immediately answer to a message by clicking on the Reply button. Doing this will take the user back to the SendMsg tab, where the Sender field shall automatically be filled for him (as in Figure 7b).

(a) (b)

Figure 7: Message details and automatic reply

The application provide a menu (that can be activated by pressing the menu key) that allows the user to shutdown Jade platform and exit from the application.

To exit from the application you can both use this menu and the back key on the emulator. In both case JADE shall be shutdown.

4.6 Testing the DummyAgent DEMO

In order to test the android DummyAgent application you have to:

1. Start the DummyAgent on the Jade Platform (refer to Jade Administrator's Guide), let's call it *da0* (see Figure 8).

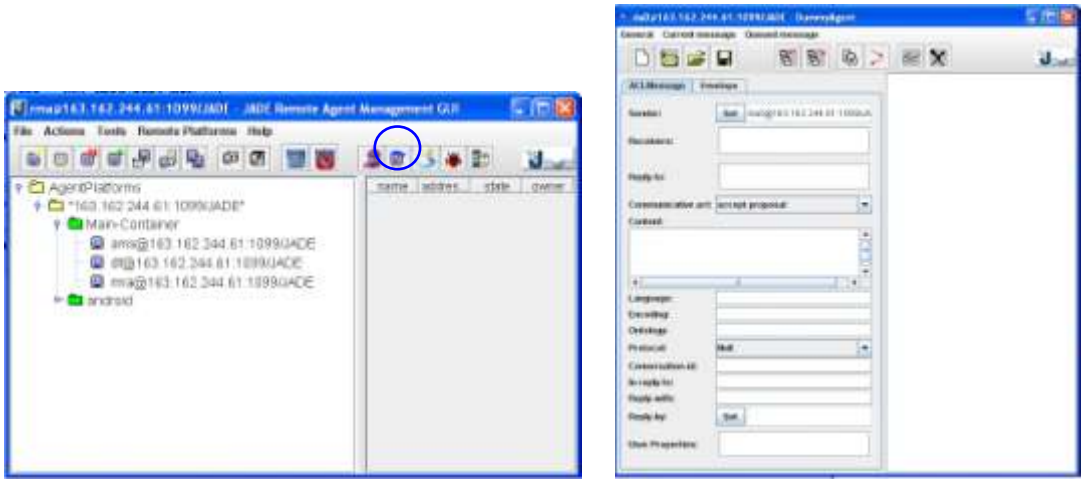

Figure 8: Launching da0

2. Send an ACLMessage from *android* agent to *da0* through the controls of the SendMsg Tab.

3. Verify the presence of the just-sent message in the *da0* GUI as shown in Figure 9.

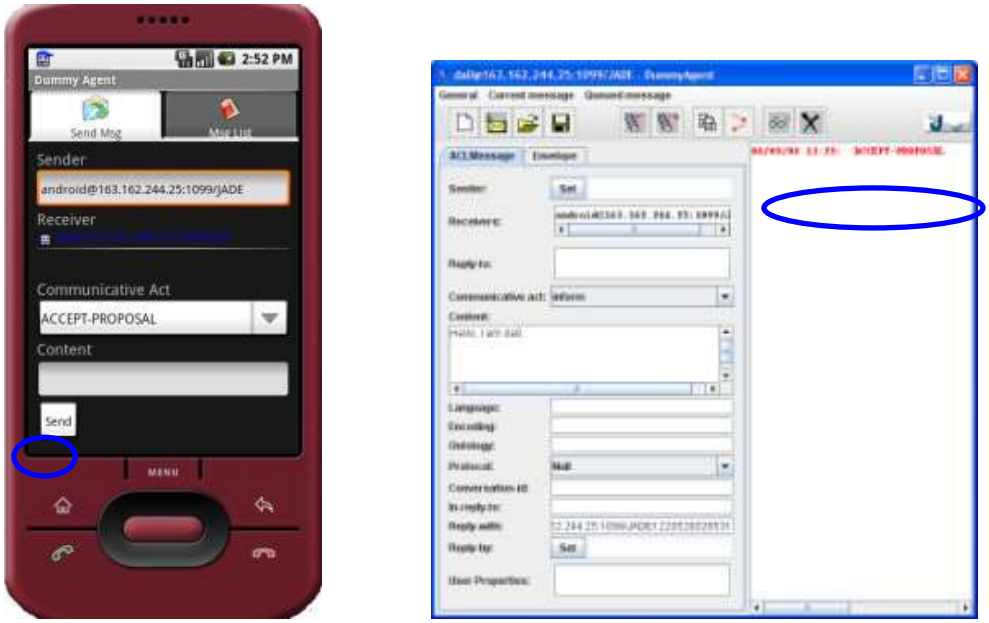

Figure 9: Sending message to da0

4. Send an ACLMessage to *android* from *da0* and verify that it is correctly received checking it in Message List Tab has shown in figure 10.

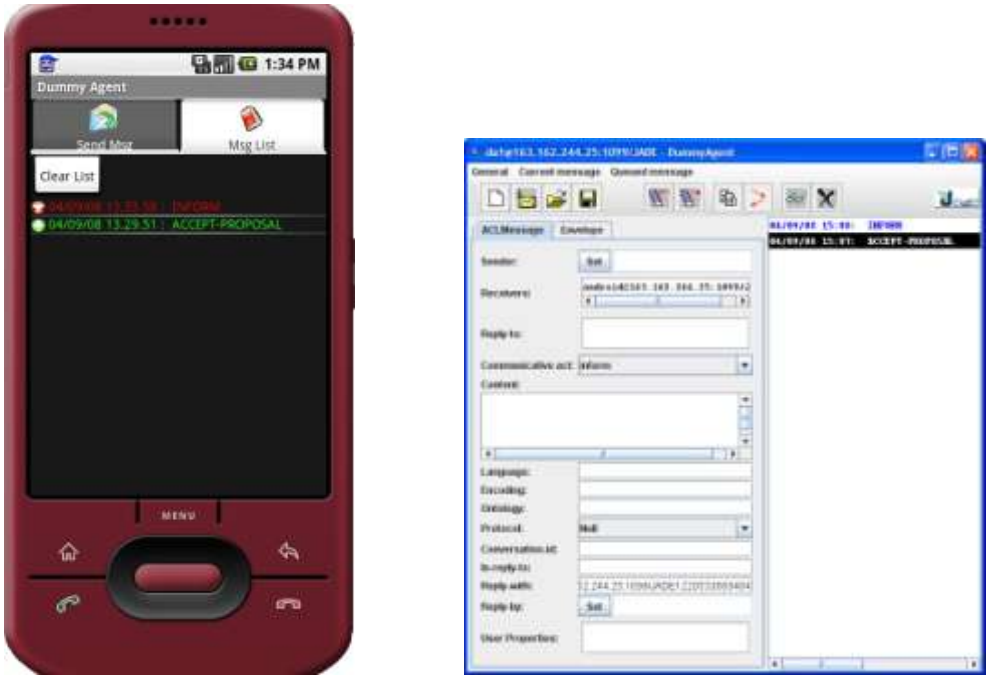

Figure 10: Sending reply message to android

Note that the same test can be performed between two or more agents running on different instances of the ANDROID SDK emulator. Obviously a Jade Main Container running on a PC is still needed. Try yourself and have fun with it.