UNIVERSITE DE TECHNOLOGIE DE COMPIÈGNE
Département de Génie Informatique
UMR CNRS 7523 Heudiasyc

JADE-OMAS Connections Tutorial

Jean-Paul Barthès

BP 349 COMPIÈGNE
Tel +33 3 44 23 44 66
Fax +33 3 44 23 44 77

Email: barthes@utc.fr

N297L v1.0
March 2014

Warning

This document explains how to set up a connexion between OMAS and JADE using the FIPA direct exchanges.

Keywords

Multi-agent Systems, FIPA, JADE, OMAS

Revisions

Version	**Date**	**Author**	**Remarks**
1.0	Mar 13	Barthès	Draft

Contents

1 Introduction

This document explains how to implement a connection between the JADE platform and the OMAS platform. Both platforms JADE and OMAS can be installed on the same machine (Fig.1) or on different machines. When they are installed on the same machine they can be used as a virtual server for the two platforms.

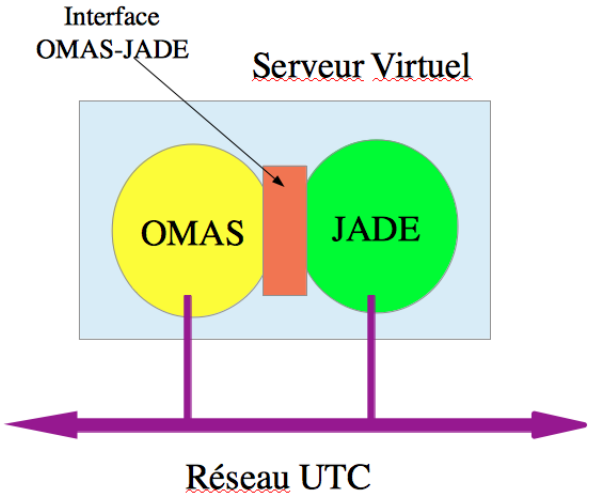

Figure 1: Virtual server containing JADE platform and an OMAS platform

Here we show how to use the transport mechanism of JADE to handle the connections. We will call this approach the *FIPA approach*.

The mechanism has been tested with version 10.0.4 of OMAS/MOSS that one can download from

`http://www.utc.fr/~barthes/OMAS/`

With the FIPA approach, one can send messages from OMAS agents to JADE agents directly and vice-versa. It requires to install a specific postman on the OMAS side and to use the Allegroserve web server. The following sections detail how to do that.

2 Installing the OMAS FIPA Postman

2.1 Copying the NB-FIPA file

Installing the right OMAS postman allows building a gateway between JADE and OMAS. To do so one should first copy the file NB-FIPA.lisp to the local OMAS coterie and declare the agent :NB-FIPA by editing the *local-coterie-agents* variable in the agents.list file, e.g.

```
(setq *local-coterie-agents*
  '(:nb-fipa :mini-contact :albert))
```

Here we have 3 agents NB-FIPA, MINI-CONTACT and ALBERT. The actual content of the corresponding agents.lisp file is give in the Appendix 6.

2.2 Setting up the Right Parameters

This is the tricky part and should be done carefully.

The content of the NB-FIPA file is complex and uninteresting, but the main information is at the beginning of the file. Two sections are of interest: Creating the Postman and the Globals section.

2.2.1 Creating the Postman

The current definition of the NB-FIPA postman is the following:

```
(omas::defpostman :NB-FIPA
    :server T
  :site :NB  ; notebook
  :http nil    ; do not start Allegroserve to handle HTTP connections
  ; this will be done by the goal. This is required to put the proper
  ; URI filter to filter JADE messages (patch done by Marcio)
  :internal-name "JEAN-PAULBAC4F6"
  )
```

where:

- :NB-FIPA is the name of the Postman and should be the same as the name of the file
- :server T means that we use Allegroserve
- :site :NB is the name of the site (coterie) NB (for notebook) is an arbitrary name and you can choose a name more meaningful for your application
- :http nil, means that we do not want the server to start automatically, as required by the JADE interface
- :internal-name is the name of your machine on your network.

2.2.2 Initial Parameters

One needs to specify a few parameters for the connection to function. This is located in the first few lines of the DHCP subsection.

```
(defparameter *host-name* "JEAN-PAULBAC4F6" "JPB Notebook")

;;; get host IP for this stupid notebook host name (virtual PC machine)
(defparameter *host-ip*
  (socket:ipaddr-to-dotted (socket:lookup-hostname "JEAN-PAULBAC4F6"))
  "can change due to DHCP connexion")

(defparameter *web-port* 80)  ; default HTTP port is 80
(defparameter *jade-port* 7778) ; default JADE port is 7778 for MTTP
(defparameter *jade-agent-port* 1099) ; used in AIDs
(defparameter *jade-host* *host-ip* "JADE installed on the JPB notebook")
;(defparameter *UTC-web-proxy* "proxyweb.utc.fr:3128" "when inside UTC")
```

You need to modify the following variables:

- *hostname* to be set to your machine name.
- *jade-host* to be the IP of the machine containing the JADE platform if different from the one running the OMAS platform.

Do not change the port names unless you have a good reason for doing it. The ports are the standard JADE port names for accessing the JADE platform.

If you need a proxy, then it is better to ask someone for help.

2.2.3 The Agent Names

The OMAS postman receives all messages send by any agent of the OMAS coterie. In order to forward the ones sent to JADE agents it needs to know the names of such agents. This is done in the *fipa-agents* variable, as follows:

```
(defparameter *fipa-agents*
  `((:KC-MUL-1 "kc-mul-1@JEAN-PAULBAC4F6" "http://JEAN-PAULBAC4F6:1099/acc")
    (:KC-MUL-2 "kc-mul-2@JEAN-PAULBAC4F6" "http://JEAN-PAULBAC4F6:1099/acc")
    (:JADE2OMAS6 ,(format nil "jade2omas6@~A:1099/JADE" *jade-host*)
                 ,(format nil "http://~A:~S/acc" *jade-host* *jade-port*)))
  "External agents, updated dynamically.")
```

Here we have 3 agents, KC-MUL-1, KC-MUL-2, and JADE2OMAS6. The best way to proceed is to remove the two first ones and to replace the name :JADE2OMAS6 by the name of your JADE agent that will receive the OMAS messages. If you want to send message to more than one JADE agent, simply duplicate this entry and add another name. Note that the value of *fipa-agents* is a list of agent addresses.

2.2.4 Exchange Protocol

The last interesting part of this file is the following:

```
(defparameter *content-language* :OMAS-CL-JSON
  "default content language is OMAS-CL with a JSON format")
```

This means that the exchange language is OMAS-CL-JSON.

2.3 Loading OMAS

Once all the parameters have been set you can load OMAS as usual. You should not have to change the settings unless you add new agents in the coterie or move the platforms to other machines.

3 Sending a message from JADE to OMAS

We assume here that the OMAS platform is running, a special NB-FIPA postman has be activated and the Allegroserve started with an entity serving a page with address /acc on the port 80.

3.1 On the JADE Side

If a JADE agent wants to send a message to ALBERT, an OMAS agent, it simply needs to write the following:

```
...
//send a message to an OMAS agent
// make message
ACLMessage omasMsg = new ACLMessage(ACLMessage.INFORM);
AID omasTarget = new AID("albert@192.168.1.17",!AID.ISLOCALNAME);
//omasTarget.clearAllAddresses();
omasTarget.addAddresses("http://192.168.1.17:80/acc");

omasMsg.addReceiver(omasTarget);
omasMsg.setContent("(:action :tell :args (\"Aaaahhh, salut from JADE.\")
                      :natural-language :fr)");
System.out.println("...omasMsg: " + omasMsg);
send(omasMSG);
...
```

Here the JADE agent builds an INFORM message containing a string. The receiver name is albert@192.168.1.17 and is a global AID, the receiver address is http://192.168.1.17:80/acc, which specifies that the HTTP protocol should be used. The message is sent and no more is required.

3.2 On the OMAS Side

On the OMAS side the message will be received by ALBERT as if it came from another OMAS agent.

3.3 Test

In this test we send an INFORM message, the content of which is a string, from JADE to the OMAS ALBERT agent. JADE runs in the ECLIPSE toolkit. Both ECLIPSE and OMAS run in the same machine, i.e. JPB's notebook, in an emulated Windows 7 partition (through Parallels Desktop). The OMAS postman is named NB-FIPA-SERVER. The ALBERT agent does not exist but we'll be able to trace the decoding of the message.

The JADE agent is called JADE2OMAS6 and is coded as follows:

```
public class JADE2OMAS6 extends Agent {
  /*
  * JADE2OMAS56 send a string to an OMAS agent using the HTTP protocol and
```

```
  * the OMAS agent address 192.168.1.12:80/acc
  * the page should have been published by AllegroServe
  */

  private static final long serialVersionUID = 1L;

  private static final String contentLanguage = "OMAS-CL";
  private static final String albertName = "albert@192.168.1.12";
  private static final String albertAddress = "http://192.168.1.12:80/acc";

  // Put agent initializations here
  protected void setup() {

    //send a message to an OMAS agent
    // make message
    ACLMessage omasMsg = new ACLMessage(0);
    AID omasTarget = new AID(albertName,!AID.ISLOCALNAME);
    //omasTarget.clearAllAddresses();
    omasTarget.addAddresses(albertAddress);

    omasMsg.addReceiver(omasTarget);
    omasMsg.setContent("(:action :tell " +
          ":args (:data \"Aaaahhh, salut from JADE.\" :language fr))");
    omasMsg.setPerformative(ACLMessage.INFORM);
    omasMsg.setLanguage(contentLanguage);
    System.out.println("...omasMsg: " + omasMsg);

    send(omasMsg);

    System.out.println("... message sent");

    }
```

Note that we give the name of ALBERT as a non local agent (not belonging to the JADE local container, we specify the performative, the content, and the content language (OMAS-CL).

ECLIPSE prints the following messages:

```
sept. 11, 2013 12:37:12 PM jade.core.Runtime beginContainer
Infos: ----------------------------------
    This is JADE 4.2.0 - revision 6574 of 2012/06/20 15:38:00
    downloaded in Open Source, under LGPL restrictions,
    at http://jade.tilab.com/
----------------------------------------
Retrieving CommandDispatcher for platform null
sept. 11, 2013 12:37:13 PM jade.imtp.leap.LEAPIMTPManager initialize
Infos: Listening for intra-platform commands on address:
- jicp://192.168.1.12:1099
```

```
sept. 11, 2013 12:37:14 PM jade.core.BaseService init
Infos: Service jade.core.management.AgentManagement initialized
sept. 11, 2013 12:37:14 PM jade.core.BaseService init
Infos: Service jade.core.messaging.Messaging initialized
sept. 11, 2013 12:37:14 PM jade.core.BaseService init
Infos: Service jade.core.resource.ResourceManagement initialized
sept. 11, 2013 12:37:14 PM jade.core.BaseService init
Infos: Service jade.core.mobility.AgentMobility initialized
sept. 11, 2013 12:37:14 PM jade.core.BaseService init
Infos: Service jade.core.event.Notification initialized
sept. 11, 2013 12:37:15 PM jade.mtp.http.HTTPServer <init>
Infos: HTTP-MTP Using XML parser com.sun.org.apache.xerces.internal.jaxp.SAXParserImpl$JAXPSAXPar
sept. 11, 2013 12:37:15 PM jade.core.messaging.MessagingService boot
Infos: MTP addresses:
http://JEAN-PAULBAC4F6.home:7778/acc
...omasMsg: (INFORM
 :receiver  (set ( agent-identifier :name albert@192.168.1.12  :addresses (sequence http://192.16
 :content  "(:action :tell :args (:data \"Aaaahhh, salut from JADE.\" :language fr))"
 :language  OMAS-CL )
sept. 11, 2013 12:37:15 PM jade.core.AgentContainerImpl joinPlatform
Infos: ----------------------------------------
Agent container Main-Container@192.168.1.12 is ready.
------------------------------------------------
... message sent
```

On the OMAS side traces are printed into the Lisp console.
First the message as received by the transfer-to-omas function is printed:

```
;===== transfer-to-omas /receiving message:
"This is not part of the MIME multipart encoded message.
--16a2b7ad26ca57339b2e063c7115d89
Content-Type: application/xml

<?xml version=\"1.0\"?>
<envelope><params index=\"1\"><to><agent-identifier><name>albert@192.168.1.12</name><addresses><u
--16a2b7ad26ca57339b2e063c7115d89
Content-Type: application/text

(INFORM
 :sender  ( agent-identifier :name jade2Omas6@192.168.1.12:1099/JADE  :addresses (sequence http:/
 :receiver  (set ( agent-identifier :name albert@192.168.1.12  :addresses (sequence http://192.16
 :content  \"(:action :tell :args (:data \\\"Aaaahhh, salut from JADE.\\\" :language fr))\"
 :language  OMAS-CL )
--16a2b7ad26ca57339b2e063c7115d89--
"
```

Note that the envelope is printed on a single line. Then the extracted content:

```
;===== content:
 "(INFORM
```

```
 :sender  ( agent-identifier :name jade2Omas6@192.168.1.12:1099/JADE  :addresses (sequence http:/
 :receiver  (set ( agent-identifier :name albert@192.168.1.12  :addresses (sequence http://192.16
 :content  \"(:action :tell :args (:data \\\"Aaaahhh, salut from JADE.\\\" :language fr))\"
 :language  OMAS-CL )
"
```

Then the transformation of the content into a list:

```
;===== transfer-to-omas /...converted to list:
(INFORM :SENDER
 (AGENT-IDENTIFIER :NAME JADE2OMAS6@192.168.1.12&1099/JADE :ADDRESSES
  (SEQUENCE HTTP&//JEAN-PAULBAC4F6.HOME&7778/ACC))
 :RECEIVER
 (SET (AGENT-IDENTIFIER :NAME ALBERT@192.168.1.12 :ADDRESSES
       (SEQUENCE HTTP&//192.168.1.12&80/ACC)))
 :CONTENT
 "(:action :tell :args (:data \"Aaaahhh, salut from JADE.\" :language fr))"
 :LANGUAGE OMAS-CL)
```

Note that the columns have been transformed into & signs to avoid package conflicts. Then we have the final translation of the OMAS message object:

```
;===== transfer-to-omas /OMAS translation:
; #<MESSAGE 12:37:15 :JADE2OMAS6 :ALBERT :INFORM :TELL ((:DATA . "Aaaahhh, salut from JADE.") (:L

=====
  TYPE: :INFORM
  DATE: 3587884635
  FROM: :JADE2OMAS6
  TO: :ALBERT
  ACTION: :TELL
  ARGS: ((:DATA . "Aaaahhh, salut from JADE.") (:LANGUAGE . :FR))
  TIME-LIMIT: 3600
  PROTOCOL: :BASIC-PROTOCOL
  STRATEGY: :TAKE-FIRST-ANSWER
=====
```

The message is sent on the network to an non existing Albert agent.

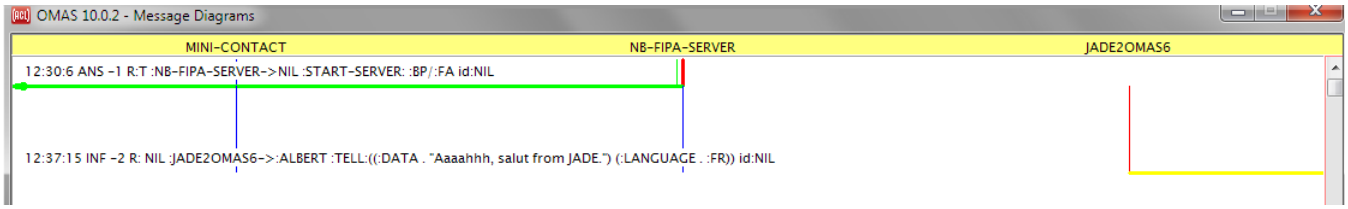

Figure 2: INFORM message sent from the JADE2OMAS postman to an external agent

4 Sending a Message from OMAS to JADE

4.1 On the OMAS Side

One simply sends a message as if the JADE agent was an OMAS agent. In particular one can do that from the OMAS Console using the new message mechanism, or have predefined messages for testing (in the Z-messages.lisp file).

4.2 On the JADE Side

On the JADE side things are very simple. One should define a behavior to recover messages from the mailbox, e.g.:

```
/*
 * Inner class JADE2OMAS. This is the behaviour used by the JADE agent to
 * check incoming requests
 */

private class ReceiveFromOMAS extends CyclicBehaviour {
 /**
  *
  */
 private static final long serialVersionUID = 1L;

 public void action() {

   // template to retrieve a REQUEST message
   MessageTemplate mt = MessageTemplate
          .MatchPerformative(ACLMessage.REQUEST);
   ACLMessage msg = myAgent.receive(mt);

   if (msg != null) {
   // REQUEST Message received. Print it as a debug information
   String argString = msg.getContent();
   System.out.println("Jade message reçu : " + argString);

     } else {
      block();
      } // end message test

   } // end action
} // End of inner class ReceiveFromOMAS
```

And call this behaviour as in:

```
  ...
  System.out.println("... message sent");

  // Add the behaviour serving addresses from other agents
  addBehaviour(new ReceiveFromOMAS());
  }
```

The content of the received message must then be processed by the agent.

4.3 Test

In this test we send a REQUEST message, the content of which is a string, from OMAS to the JADE JADE2OMAS6 agent. JADE runs in the ECLIPSE toolkit. Both ECLIPSE and OMAS run in the same machine, i.e. JPB's notebook, in an emulated Windows 7 partition (through Parallels Desktop). The OMAS postman is named NB-FIPA-SERVER.

The following message is sent by the agent :MINI-CONTACT on the coterie loop:

```
(defmessage :O2J :from :mini-contact :to :JADE2OMAS6 :type :request :action :tell
  :args ((:data . "Hello from OMAS...")(:language . :EN)))
```

The NB-FIPA-SERVER postman sees the message and discovers that it is intended for the JADE JADE2OMAS6 agent. It then processes the message. One can follow what happens in the Lisp console:

```
;===== NB-FIPA-SERVER static-send /sending message...
; OMS original:

=====
  NAME: :O2J
  TYPE: :REQUEST
  DATE: 3587916457
  FROM: :MINI-CONTACT
  TO: :JADE2OMAS6
  ACTION: :TELL
  ARGS: ((:DATA . "hello from omas...") (:LANGUAGE . :EN))
  TIME-LIMIT: 3600
  PROTOCOL: :BASIC-PROTOCOL
  STRATEGY: :TAKE-FIRST-ANSWER
  TASK-ID: -22
=====
```

and translates it into a FIPA structure :

```
; FIPA translation:
 "(REQUEST :sender (agent-identifier :name MINI-CONTACT@JEAN-PAULBAC4F6 :addresses (sequence
http://192.168.1.12:80/acc)) :receiver (set (agent-identifier :name
jade2omas6@192.168.1.12:1099/JADE :addresses (sequence http://192.168.1.12:7778/acc)))
:content \"(:TELL (:DATA . \\\"hello from omas...\\\") (:LANGUAGE . :EN))\" :language :OMAS-CL
:protocol FIPA-Request :reply-with :MINI-CONTACT$-22)"
```

The message is then send formatted as an HTTP message:

```
+++++ Sending message to JADE:
  "POST /acc HTTP/1.1
Host: 192.168.1.12
Connection: close
Accept: */*
Content-length: 1259
Content-type: multipart/mixed;boundary=\"a26859d24c38ac5882771bdbb191f5a\"
```

```
--a26859d24c38ac5882771bdbb191f5a
Content-type: application/xml

<?xml version=\"1.0\"?>
<envelope><params index=\"1\"><to><agent-identifier><name>jade2omas6@192.168.1.12:1099/JADE
</name><addresses><url>http://192.168.1.12:7778/acc</url></addresses></agent-identifier>
</to><from><agent-identifier><name>MINI-CONTACT@JEAN-PAULBAC4F6</name><addresses><url>
http://192.168.1.12:80/acc</url></addresses></agent-identifier></from><acl-representation>
fipa.acl.rep.string.std</acl-representation><payload-length>389</payload-length><date>
20130911Z2127370000</date><intended-receiver><agent-identifier><name>
jade2omas6@192.168.1.12:1099/JADE</name><addresses><url>http://192.168.1.12:7778/acc
</url></addresses></agent-identifier></intended-receiver></params></envelope>
--a26859d24c38ac5882771bdbb191f5a
Content-type: application/text

(REQUEST :sender (agent-identifier :name MINI-CONTACT@JEAN-PAULBAC4F6
:addresses (sequence http://192.168.1.12:80/acc))
:receiver (set (agent-identifier :name jade2omas6@192.168.1.12:1099/JADE
:addresses (sequence http://192.168.1.12:7778/acc)))
:content \"(:TELL (:DATA . \\\"hello from omas...\\\") (:LANGUAGE . :EN))\"
:language :OMAS-CL :protocol FIPA-Request :reply-with :MINI-CONTACT$-22)
--a26859d24c38ac5882771bdbb191f5a--
"
```

Note that the XML and the content are contained in a single line.

On the JADE side the retrieved content is the following:

```
Jade message reçu : (:TELL (:DATA . "hello from omas...") (:LANGUAGE . :EN))
```

Normally the content should be coded as a JSON object and the JADE agent will do whatever must be done with the retrieved content.

5 Protocol and Content Language

The protocol is a subset of FIPA and th content language should be OMAS-CL-JSON as defined in the following memos:

```
N287L-OMAS-JADE-fiches (en français)
N290L-OMAS-JADE-fiche-JSON
```

More detailed information is available in the Technical manual, however it mainly concerns the OMAS side.

6 Appendix A - Example of an agents.lisp File

```
;;;============================================================================
;;;13/09/30
;;;               A G E N T S (file :NB-FIPA-SERVER:agents.lisp)
;;;
;;;                      Copyright JP BArthès @UTC, 2012-2013
;;;
;;;============================================================================

;;; This file contains the name of the agents that must be loaded into the local
;;; coterie. It is called when OMAS is initialized (v5 and above)

;;; intended to be loaded onto the old notebook to check communications with KC
;;; (pegasos computer) using JADE HTTP protocol and FIPA messages

(in-package :omas)

;;; the global variable *local-coterie-agents* is defined in the OMAS load file
;;; agents should be referred to by a keyword (e.g. :book-buyer)

(setq *local-coterie-agents*
  '(:nb-fipa :mini-contact :albert))

;;; you can also define variables to control the initialization/debugging process

:EOF
```