FIPA-OS Tutorial Step 4

Using the JESS Agent

Reference FIPA-OS Tutorial Step 4
Publication History

23 August 2000
   First release. Forms step 5 of the FIPA-OS tutorial.

6 November 2000
   Re-ordered to be tutorial step 4

12 February 2001
   Updated in line with changes to tutorial agent code
Table of Contents

About this document........................................................................................................ iv
  What is this document................................................................................................... iv
  Intended Audience........................................................................................................ iv
  Reading Guide............................................................................................................... iv
  Conventions used.......................................................................................................... iv
  Terminology................................................................................................................... iv

Tutorial Step 4 ................................................................................................................ 1
  Introduction.................................................................................................................... 1
  What you will learn....................................................................................................... 1
  The JessAgent............................................................................................................... 1
    Methods..................................................................................................................... 1
    Inner class................................................................................................................ 2

Extending the JESS Agent Shell....................................................................................... 3
  FactorialAgent............................................................................................................ 3
    Inner class................................................................................................................ 3
    Running the FactorialAgent...................................................................................... 3
About this document

What is this document?
This document accompanies the JessAgent and FactorialAgent files included with FIPA-OS 1.3.0 and above. It explains how to use FactorialAgent and JessAgent as part of the FIPA-OS tutorial. It will not explain how JESS works, for documentation for JESS, see http://herzberg.ca.sandia.gov/jess.

Intended Audience
All developers using FIPA-OS to develop agent applications, or JESS developers looking for JESS implementation examples..

Reading Guide
It is strongly recommended that the reader should look at the FIPA-OS web site at http://fipa-os.sourceforge.net/ to understand the rationale behind this platform and for information on future updates. The installation, configuration, start-up and test instructions are written assuming that the developer will be using a Windows95/NT or Unix based system to run FIPA-OS.

Developers using FIPA-OS are encouraged to provide extensions, bug fixes and feedback to help improve the planned future releases. All such input should be contributed to the Open Source project via the SourceForge site at http://sourceforge.net/projects/fipa-os/. You are required to register as a developer to access some of the services at the SourceForge site. General issues and thoughts can be discussed via the FIPA-OS mailing list on fipa-os-developers@lists.sourceforge.net although you must register at http://lists.sourceforge.net/mailman/listinfo/fipa-os-developers on this list before you can send and receive messages. An archive of the messages sent to this list can also be viewed from http://www.geocrawler.com/redir-sf.php3?list=fipa-os-developers. Should you experience difficulties using this list, then please contact the FIPA-OS co-ordinators at fipaos@emorphia.com. Please consult the FIPA_OS_Public_Licence.txt file for further details on the requirements for using, extending and evolving FIPA-OS.

It is also recommended that the reader should look at the Java Expert System Shell (JESS) web site at http://herzberg.ca.sandia.gov/jess as the JessAgent described here uses JESS as its reasoning engine. It is important to understand what an expert system shell is and what the RETE algorithm is and how it works.

Conventions used
Within the text filenames appear in *italics*. In examples where users should enter data, the suggested data appears in **bold**. For examples of entering data at the command prompt, variables are encapsulated in `<` and `>` and optional data is encapsulated in `[` and `]`, e.g. `<comms-transport>` is an optional parameter which can be specified at the command prompt.

Terminology
- API: Application Programming Interface
- JESS: Java Expert System Tool
Chapter 1
Tutorial Step 4

Introduction
The JessAgent provides an interface to the Java Expert System Shell (JESS) so that an agent can use a knowledge base to reason. In this way it is hoped that the developers can write their own FIPA-OS Intelligent Agents. This document will explain how the JessAgent works, and to build agents that use the JESS engine.


JessAgent is not a stand-alone agent – i.e. there will never be an agent running called JessAgent. An agent must extend the JessAgent (which extends FIPAOSAgent) to get access to the JESS functionality. FactorialAgent described later is an example of how to extend, use control the JessAgent.

What you will learn
This tutorial aims to show the following techniques used in agent development:

- What the Jess agent is all about
- Extending the Jess agent
- Harnessing the Jess agent – making it do what you want

The JessAgent
The JessAgent has one instance of the JESS Engine – this means that for each instance of the JessAgent there is only one reasoning engine, and therefore all the rules and the facts fed in to the engine will always apply unless the engine is cleared.

Since JessAgent extends the FIPAOSAgent it will give to the extending agent all the normal agent functionalities and additional JESS functionality.

Methods
Constructor of the JessAgent sets up the agent functionality, creates a new Rete object, and removes JESS’ output router so that we don’t get any output from JESS.

Below is a list of the methods that are provided by the JESS agent to the agent extending it. More detailed method information can be found by studying the JESS agent itself and by looking at the javadocs.

protected void reset() resets the engine, which means that all the facts and all activations are removed from the engine, and then all the facts found in deffacts are asserted.

protected void clearKB() clears the knowledge base, which means that all the rules, deffacts, defclobals, deftemplates, facts, activations and the like are deleted.

protected String runEngine() runs the JESS knowledge base returning the output obtained from the engine.

protected Enumeration listActivations() returns all the activations currently in the Rete engine.
protected synchronized boolean addTemplate(Deftemplate template) adds a deftemplate object to the Rete engine returning a boolean indicating whether the operation was successful.

protected synchronized boolean addFacts(Deffacts facts) adds a deffacts object to the Rete engine returning a boolean indicating whether the operation was successful.

protected synchronized boolean assert(Fact fact) asserts a fact object to the Rete engine returning a boolean indicating whether the operation was successful.

protected synchronized boolean assertString(String fact) asserts a fact as a string to the Rete engine returning a boolean indicating whether the operation was successful.

protected synchronized Value engineExecuteCommand(String content) throws JessException executes any JESS command returning a Value object containing the output of the execution. If something goes wrong while executing the command, method throws a JessException. Use this method for running most commands to JESS (excluding adding templates and facts).

protected synchronized Context getGlobalContext() returns the Rete engine’s global context. Global context is used to determine the type of the value objects returned by engine execution.

**Inner class**
JessAgent has one inner class: Package implements the jess.Userpackage interface. This class loads a number of additional packages to the JESS Engine. Presently all packages are loaded, but the class allows for the possibility of including methods and constructors for loading combinations of packages.
Chapter 2
Extending the JESS Agent Shell

FactorialAgent
FactorialAgent is a tutorial agent to show how to use the JessAgent. It has a small knowledge base containing a function that calculates factorials. The function is written with the JESS language:

(deffunction do-factorial (?f)
  (if (= ?f 1)
    then (return 1)
    else
      (if (> ?f 0)
        then (* ?f (do-factorial (- ?f 1)))
        else (return 0)
      )
  )
)

FactorialAgent will use JessAgent by calling engineExecuteCommand(String) twice: first it will load up the knowledge base, and second it will execute the do-factorial function, resetting the engine in between. Returned result (Value object) is converted into an integer. See Figure 1 for a state transition diagram for the agent.

Inner class
FactorialAgent has one inner class – IdleTask (extends Task). This is set as the ‘ListenerTask’ and takes care of incoming requests from other agents.

Running the FactorialAgent
The compiled class file is already in the FIPA-OS tutorial jar files (with version 1.3.2 and above), so you can run the agent using the supplied batch file (agent loader is not working at this point, since in this release there are dependencies on files in certain locations). The batch file (factorial.bat) simply runs the command listed below, but retrieves information about the FIPA-OS profiles directory by running the SetupFIPAOS.bat file. More information about the batch file is included inside it as comments.

You can compile the agent again to a different directory if you want, and run it for yourself. To run the agent direct from the command line, type in the following:

    java fipaos.tutorial.Factorial C:\fipa-os\profiles\platform.profile factorial fipaos

This assumes that FIPA-OS is installed on your computer in C:\fipa-os. It sets the agent name to ‘factorial’, and the owner of the agent to FIPA-OS. You will also need to ensure that the supplied factorial.profile file is in your FIPA-OS profiles directory.

The profile file is also a required part of the agent and can be used to set agent specific information. The FactorialAgent does not have any specific information so a default profile is used. All that is set is there the naming service location, and possible usage of a database. More information about the profile can be seen as comments inside it. The profiles are written in XML, and the comments are enclosed with <!-- and -->.
When FactorialAgent is running, you can test it by sending it a test message. This can be achieved by using the IOTestAgent that comes with FIPA-OS. Simply load the IOTestAgent using the Agent Loader (started via the StartFIPAOS batch/script), and then load in the `factorial.txt` example message included in the ‘examples’ directory of your FIPA-OS tutorial installation. This file can be loaded by using the ‘File->Load from file’ command in the IOTestAgent and browsing for the text file. The ACL message looks something like this (addresses for the agents reflect your platform):

```lisp
(request
 :sender (agent-identifier :name iotestagent@localhost )
 :receiver (agent-identifier :name factorial@localhost )
 :content (10)
 :protocol fipa-request
)
```
This message will ask for the FactorialAgent to calculate the factorial of 10. When you then click ‘Send to agent’ the FactorialAgent should receive the message and respond. You will see the response appear in the display of the IOTestAgent. If the content field contains a result for the factorial, the agent works. The answer should look like this:

```
(inform
  :sender (agent-identifier :name factorial@localhost )
  :receiver (agent-identifier :name iotestagent@localhost )
  :content (3628800)
  :protocol fipa-request
  :conversation-id factorial@iiop://localhost:9000/acc9668655399
)
```