# XML based Safe and Scalable Multi-Agent Development Framework

Rinkaj Goyal,Pravin Chandra,Yogesh Singh
GGS Indraprastha University Kshmere Gate Delhi-6 (INDIA)

***Abstract:***

*In this paper we describe our efforts to design and Implement an Agent development framework that has the potential to scale to the size of any underlying network suitable for various E-Commerce activities. The main novelty In our framework Is It's capability to allow the development of sophisticated, secured agents which are simple enough to be practical.*

*We have adopted FIPA agent platform reference Model as backbone for Implementation along with XML for agent Communication and Java Cryptographic Extension and architecture to realize the security of communication Information between agents.*

*The advantage of our architecture Is Its support of agents development In different languages and Communicating with each other using a more open standard I.e. XML*

***Keywords****: Agent, Agent Development Framework, Agent Coordination, Security*

## I. INTRODUCTION

Agents can understand user's goals and carry out actions autonomously to fulfill those goals [1,2] Mobile agents are programs that can migrate from host to host in a network, at times and to places of their own choosing. The mobile agent concept grows out of three earlier technologies: process migration, remote evaluation, and mobile objects—all developed to improve on remote procedure calling (RPC) for distributed programming [6]. Agents are an effective choice for the development of applications in distributed systems, for several reasons, including improvements in latency and bandwidth of client-server applications and reducing vulnerability to network disconnection. Although not all applications will need mobile agents, many other applications will find mobile agents the most effective, basically mobile agents find their utilities in three different domains. One is data-intensive applications where the data is remotely located, is owned by remote service provider, and the user has specialized needs. Here the user sends an agent to the server storing the data. The second domain is where agents are launched by an appliance and third is for extensible servers, where a user can ship and install an agent representing him more permanently on a remote server[4].

Research Institutes as well as companies develop high-quality prototype systems for mobile systems, a brief summary is given in Table nol yet, these systems typically

Satisfy not all requirement of mobile agents for a full environment [3]. As given in the table few frameworks support Knowledge query and manipulation language (KQML) based agent communication, in KQML the major drawback is a lack of standardization in the actual transport of messages [16]. In addition, the semantics of the language have not been rigorously defined, which can lead to interoperability issues. Further, as the semantic web emerges, it is likely that web based agents will communicate with each other using a more open standard. In few frameworks i.e. open Agent Architecture [3], sometimes facilitator itself becomes a communication bottleneck, or a critical point of failure.

| SNo. | Наше | Developers | Lsisuaees | MaiorApplicationAreas |
|---|---|---|---|---|
| 1 2 | JADE EETIII | TelecomltaliaLab, Motorola CaniegU MellonUmversity | Java KQML | Developing distributed nrulli agent system CHentfServer model |
| 3<br>4 | OAA<br>JATLite | SSI International Standford University | C,C++,Java ,VB<br>Java | General Blip»! hiiarmaioriretiieval |

| | | | | |
|---|---|---|---|---|
| 5 6 | MADKIT AISLAND | URMM PraJHtJXTA | JlVlft Fytbn Java Script | Used in Grapkcal Interface Multi Docume nt Interface |
| 7 8 | MAST AGILE | CAI Fernandez | C++.ПРА-АСЦЖ Java, FIPA-ACL | Coordination of S/W Agent Collal oration and interactive vital environment |
| 9 10 | Ajanta Aglet | Minnesota UnivEcily | lava lava | General Purpose Internet |

**Table 1:** Comparative Study of various Existing Framework

This paper describes a framework in which various agents can be developed and deployed further an agent hierarchy can be maintained to facilitate an agent to accomplish its goal.

The rest of paper is organized as follows: Section 2 introduces the key issues in the selection of implementation language. Section 3 describes the architecture of Framework. Section 4 demonstrates the development of various agents. Section 5 implements the agent hierarchy and Appendix describes that how the new agents can be developed on this framework. Finally we make conclusions of Developed System

## II. SELECTION OF IMPLEMENTATION LANGUAGE

Java along with Java Cryptographic Architecture and Extension is a natural choice for the development of safe framework because of its Multiplatform support, write once, run-anywhere policy and the ubiquity of the Java virtual machine may someday facilitate dissemination of mobile agents throughout the Internet.

It is the language of choice for many multi-agent systems i.e. Concordia, Odyssey, and Voyager, [14, 15].

The JCE framework in the Java 2 SDK, v 1.4 and in JCE 1.2.2) includes an ability to enforce restrictions regarding the cryptographic algorithms and maximum cryptographic strengths available to applications in different jurisdiction contexts (locations) The Java Cryptography Architecture (JCA) provides extensible architecture to manage keys. This architecture is embodied in java security as a KeyStore. The Java KeyStore follows the existing JCA architecture which provides a framework and implementations for a KeyStore

To construct a safe infrastructure public and private key pairs is required. These unique key pair combinations provide the facility to sign and encrypt data in an authenticated, verifiable, and secure fashion. Public keys are typically stored in certificate objects, rather than alone. Sender and receiver may deal with many certificates and may have more than one private key that they use to sign and encrypt or decrypt data. We reveal an important vault of information in Java security, the KeyStore, which allows agents to consolidate and manage their various certificates and keys.

A KeyStore is a database of private keys and their associated certificates or certificate chains. The certificate chains aid in authenticating end entity certificates

## III. ARCHITECTURE OF DEVELOPMENT FRAMEWORK

After investigating and analyzing the various implemented Agent Systems [5,14,15], it has definitely assisted in understanding the requirements of a Framework. Our framework differs from the other systems due to the following reasons:

1. Heterogeneity of the various agents is acceptable. Which means that the agents can be developed in any language provided; there operation is portable with the JAVA programming language?
2. 2.The agents thus developed are hybrid in nature thus they have both compiled as well as interpreted codes executing on the various platforms as well as the JVM.
3. Though the Core modules have been developed in Java but the messaging architecture is socket based, thus all other programming language having socket-based communication can easily form a part of the environment.
4. Agent Hierarchy can be deployed using special XML based Agent property files.

The basic architecture is given in Fig 1. And its integral components are explained in the following sub-sections

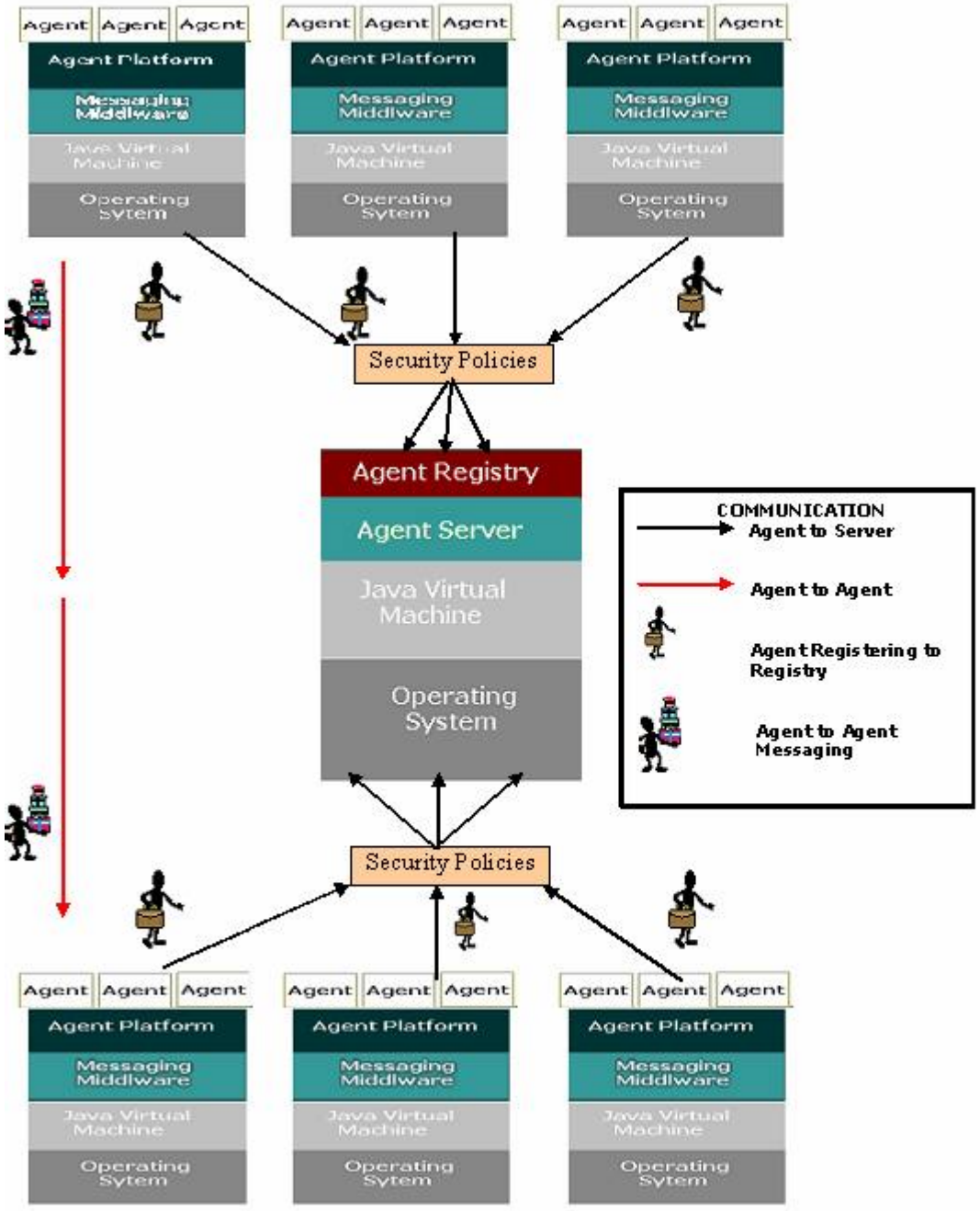


**Figure 1**: Generic Development Platform overview

*A. Platform design*

It provides the basic architecture for messaging between the various agents. Its registry feature predominantly helps the agent location and tracking process. It consists of the following components.

Global Server: The Global Server is responsible for the following actions.

- It registers all the active agents in a global agent registry.
- It shares the registry information with all the other active middleware's for easy routing of messages.

The global registry keeps track of the identification and current location of the agents. The registry stores the agent Id's and the Location Details of the various agents.

Agents: The agents, who execute on the client side comes under this category. These agents are to design in a Prescribed

Format so that they can be easily loaded into the runtime environment.

Clients: The user interfaces connected to the runtime platform as well as the various other agents comes under this category.

Messaging Middleware: This forms the core of the inter agent Communication between the various agents. The messaging middleware is Responsible for the routing of the messages across the various client nodes in the Generic Platform network.

*B. Client*

Client is also called the active distributed component of the system. It provides all the messaging support (through Client Middleware or messaging Middleware). The major components of the client are -:

- Messaging Middleware
- Local Agent Registry
- Communication engine
- Agents

Local agent registry: Registry forms the backbone of the agent information service by providing and maintaining the details of the location of all the active agents. It has the following components

1. Agent name: A unique name that identifies each agent.
2. Physical address: The network address where the agent is located.

Messaging Scheme: The message has following components

1. Agent Name: Unique name / number identifying agent.
2. Message ID: Unique number identifying the message.
3. Message Type: The type of message that is being send.
4. Message Contents: The contents of message transmitted.

*C Agent Communication*

In development framework communication plays an important role for agent co-ordination. The agents use the Message Class to create message objects. The message Class has the constructor Message (String source, String dest, String message, int id, int type) to pass the Message source address, the recipient address, the message content and agent id and type. All the communication is socket based and the message transfer and coordination is controlled by the use of Message queue by the messaging middleware.

The message objects are passed over the TCP/IP sockets, using the Object Serializabilty and in the encrypted form.

The receipt of the message by the various agents is an important event in the context of messaging co-ordination and selective reception. We propose our own customized "message event" handling for reception of messages for the agents. The proposed model is explained as under:

- All the agents register to the message event class to accept any message event, whenever it occurs.
- Whenever a message is received by the middleware a message event is pushed into the java event queue, our customized message event handler handles this event.
- The agent message listener interface provides for a function called void messageReceived(AgentMessageEvent) which has to be implemented by all the registered message listeners to perform suitable action after the receipt of the message.

*D .Encryption Algorithm Design in Agent Communication*

To realize the Security of communication information, public-key encryption is used to encrypt the communication information [10, 12]. Developed agents will be interacting with

security policy agents. Public -Private key pair will be generated by Java Cryptographic Architecture and extension API's. Fig 2 and 3 illustrate the preparatory steps

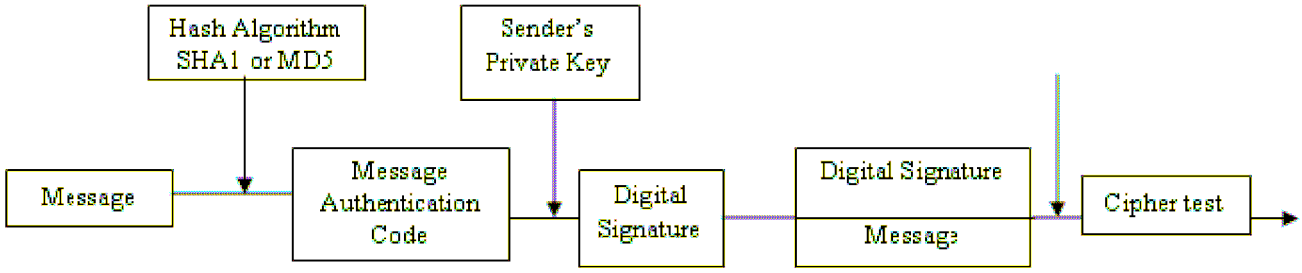


Fig 2: Encryption of messages

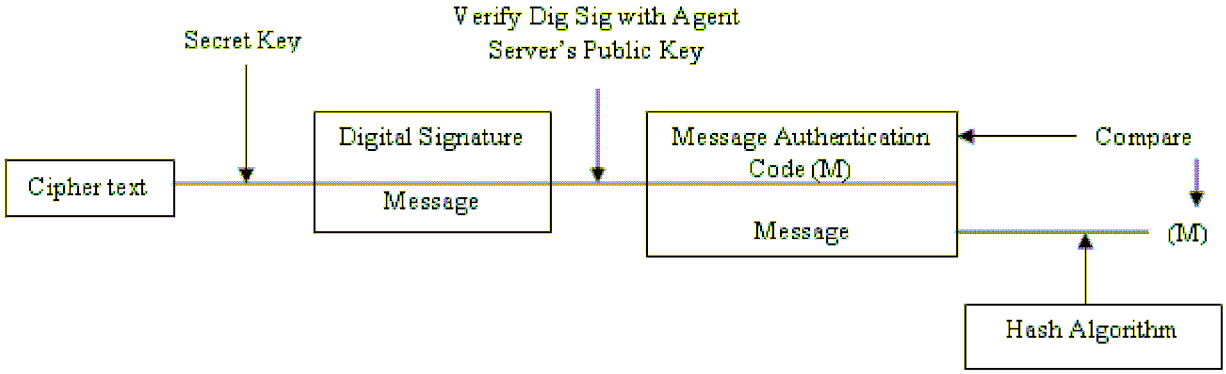


Fig 3 : Decryption of messages

## IV. VARIOUS AGENTS DEVELOPED:

Indexing agent: The indexing agent facilities the work of search and file agent by providing a ready index of files that can be searched. The search gets speeded up as already index of files is there, it prepares an index of files that user wishes to share on network and Provides support for the faster search for Search Agent.

Search agent: Search Agent uses the results provided by Indexing Agent thus showing the hierarchal approach of the Generic Deployment Framework architecture developed, it Initiate a search on the network through the messaging service provided by Framework for search initiation, monitoring and result acquisition. It Helps the File Agent to know whereabouts of a required file. Functioning is illustrated in Fig 4

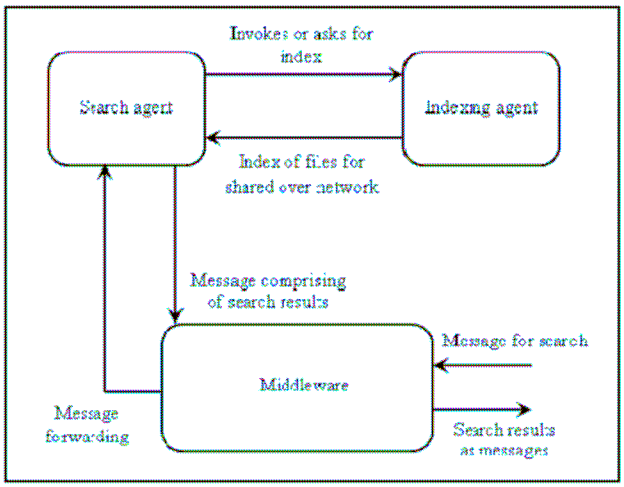


Fig 4: Search Agent

File agent: File Agent is build over the hierarchy of agents and uses the functionality provided by the agents over which it is build. It Uses search agent to retrieve the list of files available on the network and

Allows the user to get files interactively by using the messaging scheme of Framework architecture to initiate a peer connection for the file transfer. Functioning is illustrated in Fig 5

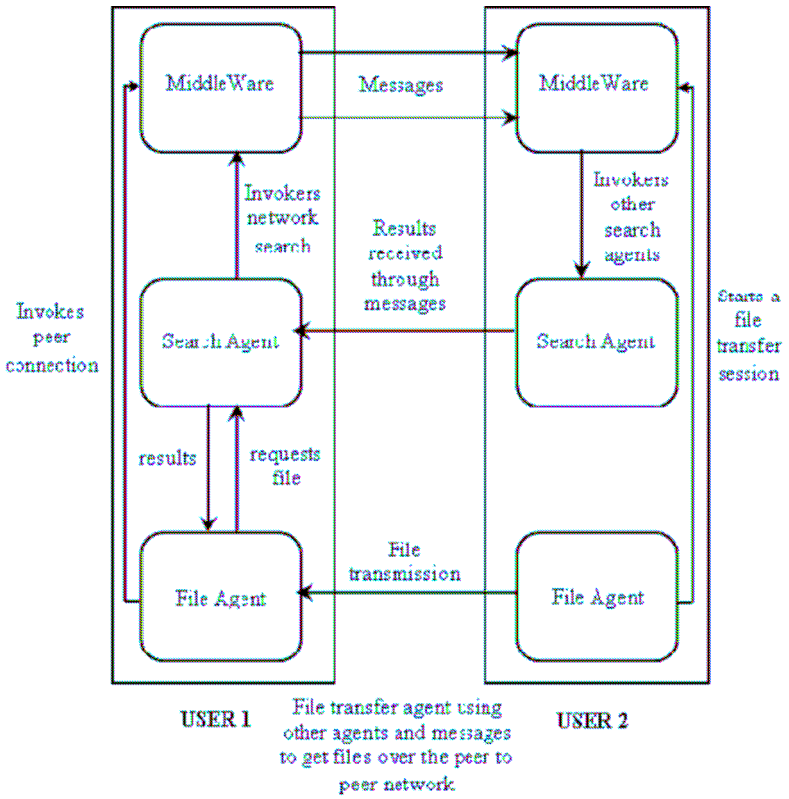


Fig 5: File Agent

*A. Agent Hierarchy*

Agents can build over a hierarchal model where agents depend over other agents for the functionality. The information of the dependency is stored in an XML document as an example XML below shows. XML can also contain optional agents.

```
- <!--

        Document   : calcagent.xml
        Created on : March 17, 2004, 11:54 PM
        Author     : root
        Description:
            Purpose of the document follows.
  -->
- <agent>
    <jarname value="calcagent.jar"/>
    <classname value="com.agent.calcagent.CalculatorAgent"/>
    <depends-on agent-name="xyz"/>
    <depends-on agent-name="abc"/>
  </agent>
```

A tree depicting the hierarchy of the dependency of the agents is build whenever an agent is run. The loader builds a tree according to which the agents are loaded and this allows an agent, which is on the top of hierarchy to use the services provided by other agents for performing its activities.

## V. CONCLUSION

Our Framework differs from many of the available agent framework because of its simplicity and small size. This system is inspired by Aglet, Odyssey, Concordia [14] [15] and JATLite and support FIPA standards. Customized Messaging Scheme and XML based architecture helps in agent communication as well as coordination, the custom agents like file search and transfer agents have been developed and Appendix has been provided for the users who wish to design their own custom agents. As with other agent development models, this model also has scope of future extension like the inclusion of better failure management, management of network congestion etc.

## APPENDIX: CREATION OF NEW AGENTS:

*A. Steps to Build a Sample Agent*

Agent's name plays an important role in the developed framework as the agent platform loads the agent by its name. Also name of the agent is sent in the message. So, it is necessary to stick to a particular name for an agent all the time. All agents are assumed to be in a package SCADE.agent. This allows us a consistent packaging scheme which allows us an easier way to manage java docs.

1. Create a public class by name Hello Agent that is the core of the hello agent class. Inherit all the features from the SCADE.bin.agents.Agent class (This class provides basic agent functionalities). Also, the agent class implements the AgentMessageListener and Runnable interface which makes agent threaded and also capable to use existing platform to listen to messages. packages java.lang.reflect, packages need to be imported.

The sample code to create class is

```
package SCADE.agent.helloagent;
import SCADE.bin.messaging.*;
import SCADE.bin.gui.*;
        public class HelloAgent extends SCADE.bin.agents.Agent implements
        SCADE.bin.messaging.AgentMessageListener, Runnable {
…….
…… }
```

2. Implement the methods provided by the interfaces

```
public class HelloAgent extends SCADE.bin.agents.Agent implements
SCADE.bin.messaging.AgentMessageListener,
Runnable{
          public void messageReceived (AgentMessageEvent evt) {
…..…..
}
public void run () {
…. }
        }
```

3. Select a unique agent Id for your agent that has not been used. Open the SCADE\bin\agents\Agent.java file and check for the last used agent Id. Add a new id for your agent as...
public static final int HELLO_AGENT = 7 ;( supposing last used id was 6)

In the message received function a core of three types of messages need to be handled. (Other types of messages are handled if your require them).

For new message add the message type to the interface SCADE\bin\messaging\MessageType.java. Add a integer message id which is not previously used.

public static final int MESSAGE_HELLO_AGENT = 22; (considering last message id was 21). Following Messages need to be handled in Message received function

1. AGENT_GUI_INITIATE: This message corresponds to the event that the user has used the GUI to start the agent.

2. INITIATE_TASK: This corresponds to message from another system to initiate an agent.

3. TASK_RESULT: This corresponds to message that result is send to the agent.

Whenever the GUI will initiate the agent it will send a AGENT_GUI_MESSAGE to the agent at its system. Considering, our helloagent will be using AGENT_GUI_INITIATE and INITIATE_TASK messages. If the system GUI initiates the helloagent then it broadcasts message to all the systems it knows.

```
// code to check Message type and Agent ID of the message send from the message.
Message m = evt.getMessage ();
// m is message object retrieved from AgentMessageEvent Object evt
if (m.getDestAgentId == super.getAgentId && m.getMessageType ==
MessageType.AGENT_GUI_INITIATE) {
// broadcasts message to all the systems
// create a new message
Message m1 = new Message (“source address”, “destination address”, null, super.getAgentId,
MessageType.AGENT_MIDDLEWARE_BROADCAST, super.getAgentId);
// source address (system ip address as string that sends message)
// destination address (for broadcasts set dest address to source address as it gets replaced)
// message payload (here null)
// source agent id (id of the sending agent)
//Message type (Broadcast message)
//destination agent id (id of destined agents) super.sendMessage (m1);
//sends the message to middleware
}
//Handling of INITIATE_TASK message
Else if (m.getDestAgentId == super.getAgentId &&
m.getMessageType == MessageType.INITIATE_TASK) {
// code to write hello
SCADE.bin.gui.mdgui.jTextArea1.append (“Hello from” + m.getSourceAddress);
// A text area jTextArea1 provided to all the agents for appending there data
}
//In the run method send a message to the system to initiate
agent and message type AGENT_GUI_INITIATE
```

```
public void run () {
Message m1 = new Message (“source address”, “dest address”, “helloagent”, super.getAgentId,
MessageType.AGENT_GUI_INITIATE, super.getAgentId);
super.sendMessage (m1);
}
```

## REFERENCES


1. Foner L., What is an agent anyway? : A Sociological Case Study, MIT Media Lab, Cambridge, MA, 1993
2. Russel Stuart & Norvig Peter, Artificial Intelligence: A Modern Approach, Prentice Hall, 1995
3. Alter Brenner, Rudiger Zarnekow and Hartmut Wittig “Intelligent Software Agents: Foundation and Applications” Springer 1998
4. Caglayan Alper & Harrison Colin “Agent a Sourcebook: A Complete Guide to Desktop, Internet, and Intranet Agents”. John Wiley & Sons, Inc. 1998
5. Tecuci Gheorghe “Building Intelligent Agents: An Apprenticeship Multi strategy Learning Theory, Methodology, Tool and Case Studies.” Academic Press. 1998
6. David Wong, N. P., Dana Moore "Java-Based Mobile Agents." Communication Of The ACM March 1999/Vol. 42 No. 3.
7. Chunsheng Li, Chengqi Zhang, "MA-IDS Architecture for Distributed Intrusion Detection using Mobile Agent." Proceedings of the Second international conference on Information Technology for application (ICITA) 2004.
8. Haetmut Vogler, T. K., Marie-Louise Moschgath "An Approach for mobile Agent Security and Fault Tolerance using Distributed Transactions." IEEE. Transaction 1997
9. Dragana Cvetkovie, Milja Pesic, Dejan Petkovie, Veljko Milutinovie, Petar kocovie and Vlada Kovacevie "Architecture of the Mobile Environment for Intelligent Genetic Search and Proxy Caching." IEEE. Transaction 2002
10. Qiang XUE,Jizhou SUN,Zunce WEI ” TJIDS: an intrusion Detection Architecture for dikstributed Network” IEEE Transaction 2003
11. Shad-chun Zhong, Qing-Feng Song, Xiao-Chun chang ,Yan Zhang ” A safe Mobile Agent system For distributed Intrusion and Detection” IEEE Transaction 2003
12. Guy G. Helmer, J. S. K. W., Vasant Honavar, Les Miller, Yanxin Wang "Lightweight Agent for Intrusion Detection." The journal of Systems and software 2002
13. Timon c.du,Eldon Y.Li,An-Pin chang “ Mobile Agents in Distributed Network Management” Communications of the ACM july 2003/Vol 46 No 7.
14. Odyssey white paper. General Magic Corp., Cupertino, Calif., 1998. Voyager white paper. ObjectSpace Corp., Dallas, Tex., 1998.
15. Deepika Chauhan, JAFMAS: A Java-based Agent Framework for Multi agent Systems Development and Implementation, ECECS Department, University of Cincinnati, 1997
16. Finin T, Peng Y and Labrou Y. (1999) Agent Communication Languages: The current Landscape. IEEE Intelligent Systems, March/April 1999, pages 45-52.


**Article received**: 2005-12-16