MaSMT 2.0
Development Guide

B.Hettige,
AS Karunananda,
and
G. Revisky

Version 2.0 (Update 17.04. 2016)
Abstract

This document describes how to develop a multi-agent system with the MaSMT2.0 agent platform. MaSMT is a free, lightweight Multi-agent system development framework, design through the Java environment. The MaSMT framework provides two types of agents, namely ordinary agents and manager agents. The framework consists of the infrastructure of the agents and message parsing mechanism. The MaSMT Messages has been implemented to comply with FIFA-ACLs specification.
# Table of Contents

Abstract............................................................................................................................ 1
Table of Contents ................................................................................................................. 2
Introduction .......................................................................................................................... 3
Infrastructure ....................................................................................................................... 3
Agent Model .......................................................................................................................... 3
MaSMTAbstractAgent .......................................................................................................... 4
MaSMT Agent ....................................................................................................................... 4
Life circle ............................................................................................................................. 5
MaSMT Manager .................................................................................................................. 6
MaSMT Root ......................................................................................................................... 9
A MaSMT System ................................................................................................................. 10
MaSMT Messages ............................................................................................................... 11
MaSMT Messages ............................................................................................................... 11
MasMT Settings .................................................................................................................. 13
MaSMT Message Parsing ................................................................................................. 14
Send Messages.................................................................................................................... 14
Read Messages..................................................................................................................... 14
Message Header .................................................................................................................. 14
Application of MaSMT ....................................................................................................... 15
Examples ............................................................................................................................. 15
Hello world Agent ............................................................................................................. 15
Ping agents ......................................................................................................................... 17
Market Simulation System ............................................................................................... 20
References .......................................................................................................................... 24
Introduction

Multi-agent technology is a modern software palindrome that capable of handling the complexity of a software system and providing intelligent solutions through the power of agent communication. A framework is a useful tool to develop a multi-agent system and it saves much programmer’s time and provides standards for the agent development.

MaSMT 2.0 (Multi-Agent System for Machine Translation) released as Open Source under the GNU General Public License (GPL). Hence, the license allows using the software to examine and modify the source code, and to develop applications based on the platform. The framework has been completely developed through the Java and provides Cross-platform capabilities. There is no prerequisite on MaSMT agents.

Infrastructure

This section briefly describes the structure of the MaSMT Agents. The MaSMT framework provides two types of agents, namely ordinary agents and manager agents. The regular agents are the action agents of the framework; precede relevant tasks according to the given messages. A manager agent consists of some ordinary agents within its control. Further, manager agents can directly communicate with other manager agents. Each and every ordinary agent in the system assigns to a particular manager agent. A regular agent in a swarm can directly communicate only with the agents in its swarm and its manager agent. The framework primarily implements infrastructure of the agents and message parsing methods to implement multi-agent system easily.

Agent Model

MaSMT Agent (including Manager) consists of Abstract model of the agents. Each agent in the system consists of group, rule, and agent id. According to the MaSMT architecture, any group consists of one-to-many rules and any rule includes some MaSMT agents. Further, MaSMT agents are capable of changing their rule at run-time. Figure 1 shows abstract agent model of the MaSMT system.
MaSMTAbstractAgent

MaSMTAbstractAgent is an abstract model of each agent in the MaSMT framework. These agents consist of a group, a rule, and an agent id. In another word, each agent of the MaSMT system has an id and assigns for the particular group and a rule. Additional information for the MaSMT Abstract agent including fields and constructors is given below.

MaSMT Agent

MaSMTAgent is the ordinary agent in the framework. MasMT agent consists of an abstract agent, two message queues (in-queue and out-queue), access rule, communication module and ontology. Figure 2 shows the architecture of the MaSMT agent.
Life circle
The MaSMT agent is a Java thread consists of three sections on its life circle such as active, live and end. The active section starts when the agent is going to activate. After completing the active section, the agent moves to its live section. Agent is in the live section while its live property is true. Agent does its all the actions such as read messages reply for the messages are doing in the live section. The end section of the MaSMT agent appears when the agent is going to die. Figure 3 shows the life-circle of the MaSMT agent.
MaSMT Manager
The MaSMT Manager is one of the ordinary agent (a Java thread) of the system with additional features. According to the MaSMT architecture, the manager can fully control its client agents (MaSMTAgent). The manager can create, remove or control (handles through
the agent priorities) its client agent(s). The Manager agent in the MaSMT creates all its client-agent automatically (as required) at the initialization stage or whenever it use. The Manager agent accesses the rule-based (agents’ ontology), and assigns rule(s) for client agents. In another word, the manager creates an agent for a rule which is available in the rule-base (This task can be changed appropriately). Further, the manager can directly access its client agents and send messages to them (add relevant messages on agent’s in-queue). The Manager Agent reads input message from in-Queue and sends relevant messages for its client agent(s). Further, manager agent can control the priority of the agents and the state of the agents. This facility removes the unnecessary workload from its client agents. The in-Queue of the managers is a queue use to store incoming messages. The manager adds messages for the out-queue where the messages need to send to others group. The Agent Controller of the Manager agent supports to control client-agents as required. In Addition to the above basic module, MaSMT manager agent consists of net access agent and message transport agent. The net access agent supports client-server communication for the system. This agent provides facilities to connect any managers on the net through the client-server networking. Message transport agent provides message transport facilities to the manager. Figure 4 shows the architecture of the MaSMT Manager.

![Architecture of the MaSMT Manager](image)

More information about MaSMT Manager agent including fields and constructors provides here.

7 | *MaSMT2.0: Development Guide*
### Constructors

**Constructor and Description**

- `MaSMTManager()`
- `MaSMTManager(MaSMTAbstractAgent abagent, int clients)`
- `MaSMTManager(MaSMTAbstractAgent abagent, int waitingTime, int clients)`
- `MaSMTManager(java.lang.String gp, java.lang.String role, int id, int clients)`

### Method Summary

#### All Methods

<table>
<thead>
<tr>
<th>Modifier and Type</th>
<th>Method and Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>abstract void</td>
<td>active()</td>
</tr>
<tr>
<td>void</td>
<td>activeAllClients()</td>
</tr>
<tr>
<td>void</td>
<td>activeMessageParsing()</td>
</tr>
<tr>
<td>void</td>
<td>activeTCPClient()</td>
</tr>
<tr>
<td>void</td>
<td>activeTCPServer()</td>
</tr>
<tr>
<td>void</td>
<td>activeTCPServer(int p)</td>
</tr>
<tr>
<td>void</td>
<td>addMessage(MaSMTMessage mes)</td>
</tr>
<tr>
<td>void</td>
<td>broadcastToClients(MaSMTMessage mes)</td>
</tr>
<tr>
<td>abstract void</td>
<td>end()</td>
</tr>
<tr>
<td>MaSMTMessage</td>
<td>getMessage()</td>
</tr>
<tr>
<td>int</td>
<td>getWorkingAgentCount()</td>
</tr>
<tr>
<td>boolean</td>
<td>isTaskComplete()</td>
</tr>
<tr>
<td>abstract void</td>
<td>live()</td>
</tr>
<tr>
<td>MaSMTMessage</td>
<td>peekMessage()</td>
</tr>
</tbody>
</table>
**MaSMT Root**

MasMT Root is a particular type of Manager agent (MaSMT Manager) that handles some manager agent as its clients. This MaSMT Root agent can fully control its client agents (MaSMT Managers). Same as Manager Agent, the root agent can create, remove or control (handle agent priority) its client manager(s). The in-Queue of the root agent is a message queue that uses to store incoming messages (messages are coming from another machine). The Root agent also adds messages on the out-queue where messages need to send to others machines. Same as MaSMT Managers The Root agent consists of Message transport agent and net access agent to handle messages. Theoretically, Root agent is the top level manager of the MaSMT system. Therefore, only one root agent available for a system. However, the root agent able to communicate with another root agent through the Net access agent. A brief description of a MaSMT systems including a root agent, number of manager agents, and number of client agents provides in the below.
A MaSMT System
The system consists of 3 layers of agents namely Root, Manager and agent. Figure 5 shows the agent’s diagram of a MaSMT system.

Figure 1: Agent diagram of a MaSMT system
According to the above agent diagram, root agent is the top level Manager, and other managers work under the root. Agents of the MaSMT categorized through agent’s rule and group. The communication between MaSMT agents (message parsing) based on the abstract model of the agents. The next section briefly describes Message structure of the MaSMT.
MaSMT Messages

In Multi-agent systems, agents communicate with each other through message passing. Such communication is the hidden factor behind the success of the multi-agent systems. MaSMT message is a type of agent’s communication message, uses to communicate with agents. The MaSMT Messages design by using FIPA-ACL message standards. MaSMT uses MaSMTMessage (a message) to communicate between MaSMT agents as well as another agent through the FIPA-ACL messages. More information about MaSMT Messages including message fields and constructors provides here.

MaSMT Messages

<table>
<thead>
<tr>
<th>Modifier and Type</th>
<th>Field and Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>java.lang.String</td>
<td>content</td>
</tr>
<tr>
<td>java.lang.String</td>
<td>header</td>
</tr>
<tr>
<td>java.lang.String</td>
<td>message</td>
</tr>
<tr>
<td>java.lang.String</td>
<td>ontology</td>
</tr>
<tr>
<td>MaSMTAbstractAgent</td>
<td>receiver</td>
</tr>
<tr>
<td>MaSMTAbstractAgent</td>
<td>replyTo</td>
</tr>
<tr>
<td>MaSMTAbstractAgent</td>
<td>sender</td>
</tr>
<tr>
<td>java.lang.String</td>
<td>type</td>
</tr>
</tbody>
</table>
### Constructors

<table>
<thead>
<tr>
<th>Constructor and Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaSMTMessage()</td>
</tr>
<tr>
<td>MaSMTMessage(ACLMessage mes)</td>
</tr>
<tr>
<td>MaSMTMessage(MaSMTAbstractAgent NewSender, MaSMTAbstractAgent NewReserver, MaSMTAbstractAgent NewReplyTo, java.lang.String NewMessage, java.lang.String cont, java.lang.String type, java.lang.String header)</td>
</tr>
<tr>
<td>MaSMTMessage(MaSMTAbstractAgent NewSender, MaSMTAbstractAgent NewReserver, MaSMTAbstractAgent NewReplyTo, java.lang.String NewMessage, java.lang.String cont, java.lang.String onto, java.lang.String type, java.lang.String header)</td>
</tr>
<tr>
<td>MaSMTMessage(MaSMTAbstractAgent NewReserver, java.lang.String NewMessage, java.lang.String header)</td>
</tr>
<tr>
<td>MaSMTMessage(MaSMTAbstractAgent NewReserver, java.lang.String NewMessage, java.lang.String NewContent, java.lang.String type, java.lang.String header)</td>
</tr>
<tr>
<td>MaSMTMessage(MaSMTMessage mes)</td>
</tr>
<tr>
<td>MaSMTMessage(java.lang.String newheader, java.lang.String newMessage)</td>
</tr>
</tbody>
</table>

### Method Summary

#### All Methods

<table>
<thead>
<tr>
<th>Modifier and Type</th>
<th>Method and Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACLMessage</td>
<td>getACLMessage()</td>
</tr>
<tr>
<td>java.lang.String</td>
<td>getContent()</td>
</tr>
<tr>
<td>java.lang.String</td>
<td>getHeader()</td>
</tr>
<tr>
<td>java.lang.String</td>
<td>getMessage()</td>
</tr>
<tr>
<td>java.lang.String</td>
<td>getOntology()</td>
</tr>
<tr>
<td>MaSMTAbstractAgent</td>
<td>getReceiver()</td>
</tr>
<tr>
<td>MaSMTAbstractAgent</td>
<td>getReplyTo()</td>
</tr>
<tr>
<td>MaSMTAbstractAgent</td>
<td>getSender()</td>
</tr>
<tr>
<td>java.lang.String</td>
<td>getType()</td>
</tr>
<tr>
<td>java.lang.String</td>
<td>print()</td>
</tr>
</tbody>
</table>
**MasMT Settings**

MaSMT Settings class uses to handle (Set) Environment settings of the Multi-agent system. This configuration required enabling system agents as required. Details of the configuration settings of the MaSMTSettings class is given in the below.

<table>
<thead>
<tr>
<th>Constructor and Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Settings()</td>
</tr>
<tr>
<td>Settings(int newdelay, int newMaxDelay, int newMinDelay)</td>
</tr>
<tr>
<td>Settings(int newdelay, int newMaxDelay, int newMinDelay, java.lang.String newproxy, int newport, java.lang.String newhost)</td>
</tr>
<tr>
<td>Settings(java.lang.String newproxy, int newport, java.lang.String newhost)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modifier and Type</td>
</tr>
<tr>
<td>static java.lang.String</td>
</tr>
<tr>
<td>static boolean</td>
</tr>
<tr>
<td>static int</td>
</tr>
<tr>
<td>static java.lang.String</td>
</tr>
<tr>
<td>static int</td>
</tr>
<tr>
<td>static int</td>
</tr>
<tr>
<td>static java.lang.String</td>
</tr>
<tr>
<td>static int</td>
</tr>
<tr>
<td>static java.lang.String</td>
</tr>
<tr>
<td>static float</td>
</tr>
</tbody>
</table>

The MaSMT Settings class fields provide as public static fields. Therefore, any system agent can easily access these configuration data as required. However, if you not set MaSMT settings before active a manager agent, the manager uses default MaSMT settings. Default configuration settings of the MasMT System is given below.
MaSMT Message Parsing

This section briefly describes message parsing methods for the MaSMT system.

Send Messages
1. To enable message parsing call `activeMessageParsing();` on the managers.
2. Create message and call `addMessage(MaSMTMessage)`
3. To send messages for all clients, call `broadcastToClients(MaSMTMessage)`

Read Messages
1. `getMessage()`: get message from in-queue. If message queue is empty then return null
2. `peekMessage()`: get front message without changing front message
3. `waitForMessage()`: system waits until message read.

Message Header
Agents can easily redirect messages according to it message header.

- **local**: send a message to a particular client(s) who has given group and rule.
- **localgroup**: send a message to a particular client(s) who has a given group
- **localrule**: send a message to a particular client(s) who has a given rule
- **localruleorgroup**: send a message to a particular client(s) who has a given rule or group

```java
Settings.name = "masmt";
Settings.version = 2.1f;
Settings.delay = 50;
Settings.debugMode = false;
Settings.proxy = "0.0.0.0";
Settings.port = 8088;
Settings.appPath = "";
Settings.host = "localhost";
Settings.MaxDelay = 100;
Settings.MinDelay = 10;
```
**broadcast**: send a message to all clients

**broadcasttomanagers**: send message to all other managers

**localmanager**: send a message to a particular Manager(s) who has given group and rule.

**localmanagergroup**: send message to a particular manager(s) who has a given group

**localmanagerrule**: send message to a particular manager(s) who has a given rule

**localmanagerruleorgroup**: send message to a particular manager(s) who has a given rule or group

**Application of MaSMT**

Using the framework number of the multi-agent system has been developed including Natural Language Processing applications such as English to Sinhala Agent-based Multi-agent system [2], Sinhala Ontology Generator [3] and Multi-agent based Morphological Analyzer [4]. In addition to the above Intelligent chatbot [5] and a communication platform for Agricultural domain [6] has already developed through the MaSMT.

**Examples**

The following section briefly gives some example to demonstrate how to use MaSMT framework for developing multi-agent systems.

- HelloworldAgent
- Ping agents
- Market example with manager-client message parsing
- Market example with client-client message parsing

**Hello world Agent**

The objective of this example is to demonstrate how agent follows it life-circle. The HelloWorldAgent executes active, live and end section continuously and going to die. In the live section agent print, a message on 10 times.
public class HelloWorldAgent
    extends masmt2.agent.MaSMTAgent {

        public HelloWorldAgent() {
            super();
        }

        public static void main(String[] args) {
            // Create an agent
            HelloWorldAgent agent = new HelloWorldAgent();
            agent.start();
        }

        public void live() {
            System.out.println("Agent in the live mode..");
            for (int i = 10; i > 0; i--) {
                System.out.println("I am exit in " + i + " seconds");
                wait(500);
            }
            setLive(false);
        }

        public void active() {
            System.out.println("I am " + super.agent);
        }

        public void end() {
            System.out.println("End.");
        }
    }
**Ping agents**
This example demonstrates how agents communicate with each other. The example consists of three agents including a manager agent, and 2 ordinary agents namely PingAgentA and PingAgentB. These two agents communicate together through the message parsing. The manager agent creates two agents with the same group and 2 rule (“roleA” and “roleB”). The PingAgentA send a message to PingAgentB and PingAgentB read this message and reply to pingAgentA again. This reply message also read by the PingAgentA and reply again for the PingAgentB. The following diagram shows agent diagram for the example.

```
public class PingManager extends masmt2.agent.MaSMTManager {
    MaSMTMessage tempmes; // A Temporary message
    public PingManager() {
        super();
    }
    public static void main(String[] args) {
        // Create a Manager agent with 2 clients
        PingManager pma = new PingManager();
        pma.start();
    }
    public void active() {
```

---

17 | *MaSMT2.0: Development Guide*
System.out.println("[Active] MANAGER .... " + super.agent);

// SET Number of Client agents 2
setNumberOfClients(2);

agents[0] = new PingAgentA("group", "roleA", 1);
agents[1] = new PingAgentB("group", "roleB", 2);
activeAllClients();
activeMessageParsing();
}

public void live() {}

public void end() {}

public class PingAgentA extends masmt2.agent.MaSMTAgent {

MaSMTMessage tempmes;

public PingAgentA() {
    super();
}

public PingAgentA(String gp, String role, int id) {
    super(gp, role, id);
}

public void active() {
    System.out.println("[Active] .... " + super.agent);
    If(agent.id== 1){
        MaSMTAbstractAgent reader = new MaSMTAbstractAgent("goup", "roleB",2);
        MaSMTMessage m =new MaSMTMessage(agent, reader, agent, "ping", "AgentA","text", "local");
    }

    Create 2 Ping Agents with same rule and rgoup
    Active Message Parsing
    Set Number of client agent is 2
    Create a new reader
public void live() {
    System.out.println("[live] ...."+ super.agent);
    // Read new Message
    tempmes = waitForMessage();
    MaSMTMessage newm =
        new MaSMTMessage(agent, tempmes.sender, agent,
                          "ping",tempmes.content+"-agentA", "text","local");
    System.out.println(super.agent +"[AgentA GET]" + tempmes);
    sendMessage(newm);
    wait(1000);
}

public void end() {}
public void active() {
}

public void live() {
    System.out.println("[live] ...." + super.agent);
    tempmes = waitForMessage();
    MaSMTMessage newm =
        new MaSMTMessage(agent, tempmes.sender, agent,
                        "ping",tempmes.content+"-agentB", "text","local");
    System.out.println(super.agent +"[AgentA GET] " + tempmes);
    sendMessage(newm);
    wait(1000);
}

public void end() {
    {
    }
}

**Market Simulation System**
This example simulates Market example including a number of the Sales agent and a customer agent. The MaketManager is the manager agent that create all the required agents. The customer agent is a request agent that make a request to get items from the market. The sales agents are a resource and with sales items. According to the customer agent request the resource agents are replied when they can sell items. After Agent communication customer agent is able to select suitable sales request.
This example is also available on the MaSMT download location.

URL: [https://sourceforge.net/projects/masmt/](https://sourceforge.net/projects/masmt/)

```java
public class SalesManager extends masmt2.agent.MaSMTManager {

    MaSMTMessage tempmes;
    public SalesManager() {
        super();
    }
    public static void main(String[] args) {
        // Create an agent to start
        SalesManager sma = new SalesManager();
        sma.start();
    }

    @Override
    public void active() {
        System.out.println("[Active] MANAGER .... " + super.agent);
        setNumberOfClients(5);
        agents[0] = new CustomerAgent("masmt", "cus", 1);
        agents[1] = new SalesAgent("Bread", 48.00f, 25, "masmt", "sale", 2);
        agents[2] = new SalesAgent("Suger", 168.00f, 25, "masmt", "sale", 3);
        agents[3] = new SalesAgent("Toffee", 1.00f, 25, "masmt", "sale", 4);
        agents[4] = new SalesAgent("Chocalate", 35.00f, 25, "masmt", "sale", 5);
        activeAllClients();
        activeMessageParsing();
    }
}
```
public void live() { }

public void end() { }

public class SalesAgent extends masmt2.agent.MaSMTAgent {
    // Sale agent data
    private String item;
    private int quntity;
    private float unit_price;
    private float value;

    // Temporary message
    MaSMTMessage tempmes;

    public SalesAgent() {
        super();
    }

    public SalesAgent(String gp, String role, int id) {
        super(gp, role, id);
    }

    public SalesAgent(String item, float unit_price, int quntity, String gp, String role, int id) {
        super(gp, role, id);
        this.item = item;
        this.quntity = quntity;
        this.unit_price = unit_price;
    }

    @Override
    public void active() {
        System.out.println("[Active] .... " + super.agent);
    }

    @Override
    public void live() {
        // System wait until message read
        tempmes = waitForMessage();

        if(tempmes.message.equals("get_sales")) {
            value = Float.parseFloat(tempmes.content);
            int qount = (int) (value/unit_price);
        }
    }
}
if(qount>0)
{
    String m =""+ agent +" SALE "+ qount + " " + item + "( " + unit_price + " per each)" ;
    System.out.println(m);
}
setLive(false);
}

public void end() {}
public void live() {
    tempmes = waitForMessage();
    System.out.println(""+ tempmes);
}

public void end() {
}

To get more details of the MaSMT project and its applications, use the MaSMT website.

References