1 Introduction

LGI is a message-exchange mechanism that allows an open group of distributed agents to engage in a mode of interaction governed by an explicitly specified policy, called the law of the group. The messages exchanged under a given law are called Law-message. The group of agents interacting via Law-message is called a Law-community. The LGI-community ensures the agents to exchange messages in a secure and scalable way.

The backbone of an LGI system consists of a number of controllers. An agent needs to connect to a controller in order to perform certain activities in the LGI-community. The controllers usually run on different hosts, they are started at different times and possibly by different administrators. They might have different workload and different working characteristics as well.

The need for a tool that helps manage the controller pool and maintains a global up-to-date information regarding all the controllers is not only helpful but necessary instead. The Controller Manager is a LGI component that acts as a centralized manager for the registered controllers. It maintains updated information regarding the parameters and status of any registered controller. The Controller Manager has a graphical user interface which displays the current status of each registered controller and provides interactive methods for users who has the manager privilege to create, configure, test, or destroy the controllers. Even though the Controller Manager is a centralized component, it is not placed in the direct communication path. Thus it is not subject to scalability problems.

This document describes the operation and interaction mode of the Controller Manager as follows: Section 2 describes the architecture of the Con-
Figure 1: Controller Manager Architecture Model

troller Manager. Section 3 presents the interface provided to the general users. Section 4 presents the functions of the manager interface. Section 5 serves as reference and explains in details on how to run the Controller Manager.

2 Architecture

As presented in the previous section the Controller Manager serves as a centralized public manager for controllers. Figure 1 presents the interaction model of the Controller Manager. The interacting entities can be classified into the following categories:

- Web-based users: The Controller Manager has a web-based graphical user interface that allows interaction with two types of users: regular users and managers. The regular users can access the general information regarding the registered Controllers (details in Section 3) whereas managers can interactively create, configure, test, and destroy the controllers using the functions provided by the Controller Manager.
Native controllers: The Controller Manager can create and destroy controllers on behalf of managers. The controllers created this way are called native controllers. A remote-shell connection is maintained between each native controller and the Controller Manager. The standard output of the controller is passed along this connection and can displayed in a Controller Manager window if configured properly.

Adopted controllers: Any controller that is not native but registers with the Controller Manager is called adopted controller.

The Controller Manager can accommodate any combination and number of users, managers, and controllers simultaneously.

The rest of the documents will describe the interaction model between Controller Manager and each of the entities above.

3 User Interface

The main function of the Controller Manager is to display the controller related information to all the requesting users. Users can retrieve the controller directory by consulting the Controller Manager’s web page. Assuming that the Controller Manager runs on the machine ramses at Rutgers University on port 7000, its main web page can be accessed through either of the following URLs:

\[
\begin{align*}
\text{http://ramses.rutgers.edu:7000/} \\
\text{http://ramses.rutgers.edu:7000/index.html}
\end{align*}
\]

Any user accessing the URLs above has unrestricted access to the controller directory information. Figure 2 presents a sample of Controller Manager’s main web page. The information inside this page is organized in two tables: Native controllers directory table and Adopted controllers directory table.

Native controllers directory table contains the following fields:

- Host: the host name of the machine where the native controller has been started.
- Port: the port number where the controller waits for connections.
Figure 2: Browser view - directory of controllers

<table>
<thead>
<tr>
<th>Index</th>
<th>Host</th>
<th>Port</th>
<th>Type</th>
<th>User</th>
<th>Output</th>
<th>Start Time</th>
<th>Test Time</th>
<th>Status</th>
<th>Load</th>
<th>Certificates</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>research.rutgers.edu</td>
<td>7080</td>
<td>Prolog/Java</td>
<td>mlk</td>
<td>0-24</td>
<td>03/15/2005</td>
<td>03/15/2005</td>
<td>Test Java_Success</td>
<td>0</td>
<td>View</td>
</tr>
<tr>
<td>1</td>
<td>research.rutgers.edu</td>
<td>7080</td>
<td>Prolog/Java</td>
<td>mlk</td>
<td>0-24</td>
<td>03/15/2005</td>
<td>03/15/2005</td>
<td>Test Java_Success</td>
<td>0</td>
<td>View</td>
</tr>
<tr>
<td>2</td>
<td>rnc.rutgers.edu</td>
<td>9000</td>
<td>Prolog/Java</td>
<td>mlk</td>
<td>0-24</td>
<td>03/15/2005</td>
<td>03/15/2005</td>
<td>Test_Prolog_Failure Test Java_Failure</td>
<td>0</td>
<td>View</td>
</tr>
</tbody>
</table>
• Type: the declared type of the controller. The manager that starts the controller is responsible for declaring a certain controller type. Values like “Prolog” or “Java” denote the type of law engine controller provides.

• User: the user name of the manager that started the current instance of the controller.

• Output: the type of the output the controller provides. Valid values are “public” or “private”. More to be discussed by the end of this section.

• Start Time: The date and time the current instance of the controller was started.

• Test Time: The date and time the current instance of the controller was last tested.

• Status: The status of the controller. As a result of manual testing or automatic testing, the Controller Manager updates the status of each controller. Sections 4 and 5 provide details on the controller testing.

Adopted controllers directory table contains the following fields:

• Host: the declared host name of the registered controller.

• Port: the port number where the declared controller waits for connections.

• Registration Time: the date and time this controller registered with the Controller Manager.

• Status: The status of the controller, tested as above.

Besides the fields presented above, each controller host entry (in both tables) provides a link to the controller user interface. By clicking on the Host entry of a specific controller in the table, a user can create an agent connected to the specific controller.

The native controller table also presents links to the standard output of each controller. Under the Output column, by clicking public or private keyword, the user can access the controller output web page.
The controller output represents the messages the controller displays in debug mode. During initial law developing and subsequent law testing it is critical to follow the controller debugging information. The Controller Manager retrieves the standard output of each controller and makes it available to authorized users. Since all controllers are considered to be shared among multiple users, viewing the output of a controller implies viewing the debugging information regarding all interacting users, which is a privacy concern. Thus, the output is protected through a shared password. The manager that starts the controller chooses its output password. In the case of a public output controller, no password is required in order to view the output of that controller (everybody can see the messages of any user of that controller).

Figure 3 presents the output page for a private and respectively public controller. This page displays a message text area along with the following
controls:

- **Capture**: connects to the controller and starts sending the controller standard output to the browser text area
- **Clear**: clears the text area from any previous message
- **Stop**: stops the browser from receiving the output from the controller
- **Password**: this is the password the users should provide when attempting to capture the output of the controller. This field is shown in private controller output windows

The standard output of any controller can be captured by many users, and a user can capture the output of many controllers in the same time.

## 4 Manager Interface

The Controller Manager accommodates two types of users: regular users who have access to the controller information described in the previous section, and manager users. Manager users can send specific commands and can configure the functionality of the Controller Manager through a Manager Interface. As in Figure 2, the main web page provides two links, ”Manager” (at the top and at the bottom of the page) that both lead to the manager interface.

The Controller Manager requires user-name/password authentication for each manager that attempts to connect to the manager interface (Figure 4). Details on user-name/password setup will be provided in Section 5. After the manager’s authentication, the interaction with the Controller Manager can start.

There are four types of activities a manager can perform: Create(default), Kill/Destroy, Test, and Configure. The remaining part of this section describes these activities.

### 4.1 Creating a Controller

By selecting the Create button from the manager interface menu, a manager can decide to create a new native controller. The upper part of Figure 5 presents the content of the Create user interface:
Figure 4: Manager Login Window

- **Host Name**: the host name where the controller should be started.
- **Port Number**: the port where the controller waits for connections.
- **Path to Files**: the directory where the controller holds its configuration files.
- **Package**: the specific package used for starting a certain controller.
- **Type**: the declared type of the controller (Prolog or Java for current implementation).
- **Controller Output Password**: the manager that creates a controller is responsible for setting the password that protects the standard output of the controller. The usage of this password has been discussed in Section 3.
- **Compound Command**: represents the starting command that will be used to start the controller on the remote host, as it were to be issued at the host’s console. The compound command is computed based on the above terms and is recomputed whenever the manager modifies any of the fields above then hits the key “Enter” on the keyboard.
Figure 5: Starting and Stopping a Controller
• Create: when issuing this command, the Controller connects remotely to the specific hosts (ssh execution) and creates a new controller using the command line from Compound Command field.

• Reset fields: resets the values of the above fields to their default values (more on default values in Section 5).

Upon each attempted controller creation, a status message is displayed at the bottom of the page. This message indicates that the operation was successful or, if not, the reason of the failure.

4.2 Stopping a Controller

A controller can be stopped through the Kill option of the manager interface menu (Figure 5, lower part). The following fields are significant inside this window:

• Host Name: the host-name of the controller to stop.

• Port Number: the port number of the controller to stop.

A controller is uniquely identified by its (host-name, port number) pair. Pressing the Kill button, the Controller Manager attempts to kill the controller process. The significance of Reset fields is similar to the one described in Subsection 4.1. A status message that indicates the success or failure is displayed at the bottom of the page after the Kill operation is performed.

4.3 Testing Controllers

One of the main functions of the Controller Manager is to report the operational status of each registered controller. Although the Controller Manager maintains connections to all registered controllers (native and adopted) this is only enough to assert the running status of each controller process. As long as the process of any controller dies, the controller is removed from the Controller Manager directory. However, in order to assess the degree of responsiveness of a controller, further testing is required.

The Controller Manager performs a periodical testing, at a interval specified in its configuration (discussed in Subsection 4.4). It also performs an
Figure 6: Testing a Controller and Configuring the Controller Manager
on-demand testing procedure. The testing interface (Figure 6, upper part) is concerned with the on-demand testing of the controllers.

A manager has to specify the following items in order to perform a controller test:

- **Host Name**: the host-name of the controller to test.
- **Port Number**: the port number of the controller to test.
- **Test Type**: the testing type *Prolog* or *Java* - currently - specifies the type of law used during testing.

The testing procedure involves creation of a user agent that connects to the controller under a random name and a specific testing law. The agent exchanges sufficient number of messages to assess the status of the controller. The type of testing - *Prolog* or *Java* - serves in deciding what law to choose in testing the controller. The choice depends on the ability of the controller to accommodate Prolog and Java law engines.

A manager can use the following buttons:

- **Reset Fields**: changes the value of the fields to a default value
- **Test**: initiates the testing procedure for the controller specified by (host-name, port number) pair
- **Test All**: initiates a testing procedure sequentially for all the controllers registered with the Controller Manager

Upon testing a controller, a success or fail message is displayed on the bottom of the page. The status of that controller is also updated in the directory of the main web page (in the main web page use *Reload* or *Refresh* browser button to obtain the most up to date information from the directory, including the last test time and the test result).

### 4.4 Configuring the Controller Manager

The configuration interface allow the manager to retrieve and specify certain generic functions related to the Controller Manager (Figure 6, lower part). The functions can be classified as follows:
• Get Output Password: this command allows any manager to retrieve the password necessary to view the output of any controller (multiple managers share equally all the native controllers). The manager has to identify the controller through the (host-name, port number pair).

• Change Testing Period: this command allows a manager to change the time interval used in automatic periodical testing. The manager should specify the new testing interval (in seconds).

• Change User Manager Password: this option allows any user manager to change his own login password. The old password and the new password should be presented.

• Get Current Active Managers: Since the Controller Manager can accommodate multiple concurrent managers, this command displays the number of managing sessions currently open.

5 Detailed Usage

This section describes how to start up the Controller Manager, how to change the configuration file, and the details of the testing laws used by the Controller Manager.

5.1 Starting the Controller Manager

A Controller Manager could be started as following, for example, using port number 7000:

```
java moses.ControllerManager -sp7000 [-debug]
java moses.ControllerManager -sp7000 -p/farm/moses/controllerManager/moses/jars -fcfg [-debug]
```

The arguments required in Controller Manager starting line are:

• port: specifies the port where the Controller Manager waits for TCP/IP or HTTP requests.

• -ppath: path represents the directory used to store different necessary files. The configuration file (see below) is identified relatively to this path. If not specified, current directory is considered.
host = research.rutgers.edu
port = 9080
path =
pack = moses.Controller
type = Prolog/Java
pass = public
testperiod = 600
accounts =
user(moses) pass(52BD43D37ED62EB4C226E31841BC03DC)
user(anonymous) pass(6540BD43D37ED62EB4C226E18O187GH4)
# "52BD43D37ED62EB4C226E31841BC03DC" represents the hash for "moses"

Figure 7: Controller Manager Configuration File

- -fcfgfile : cfgfile represents the configuration file as discussed in Section 5.2 and Figure 7. The name for the configuration file is cfg.

- -debug : optional for turning on the debugging information for the Controller Manager.

The Controller Manager is designed to run in a environment with several properties. First, the machine running the Controller Manager should be able to execute ”ssh” on remote computers. More than that, the remote computers should be in the same administrative domain, thus allowing execution of ssh without password authentication. Another requirement refers to the execution environment of the running command: the location of different packages on remote computers should be known apriori, at the Controller Manager host. As a test case to asses the ability of a Controller Manager to start native controllers, the following command should be verified:

```
ssh -t host java moses.Controller -sp9080 ...
```

5.2 Configuration File

During start up, the Controller Manager needs to take its initialization information from a configuration file. Figure 7 presents such a configuration file model:
• The fields from *host* to *pass* provides the default values a manager encounters when attempting to create a new controller.

• The field *testperiod* represents the default interval for triggering periodic testing (in seconds).

• The manager user-name and password follow the keyword *accounts*. While the user-name is stored in plain text, the password is stored in “scrambled - hash” format. Section 5.3 talks about how to create the hash code for the manager password.

• Any line starting with the symbol “#” is considered to be comment.

5.3 Utility

The administrator of the Controller Manager is responsible for creating each manager’s user-name and password. The user-name and password are stored in the offline configuration file. The password is stored in the format of hash code. The hash code of the password is computed from the original password by using the utility of *GeneratePass*. This utility’s usage is:

```
java GeneratePass original-passwd
```

5.4 Testing Laws

Figures 8 and 9 present the laws that a Controller Manager uses for testing controllers with Prolog and Java law engines.

Since the Controller Manager creates agents that operate under these laws, the Controller Manager acts as law server for the above laws. For example, the URL of the testing laws are:

```
http://ramses.rutgers.edu:7000/testlaw.law
http://ramses.rutgers.edu:7000/testjava.java1
```
Preamble: law(testlaw,language(prolog)).

R1. adopted(par(A),cert(C)) :-
    do(deliver(Self,adopted,Self)).

R2. sent(X,getCS,Y) :-
    do(discloseCS(all)).

R3. sent(X,M,Y) :-
    do(forward).

R4. arrived(X,M,Y) :-
    do(deliver).

R5. disconnected :-
    do(quit).

Figure 8: Testing law (Prolog)
public class testjava extends Law {
    public void adopted(String args) {
        doDeliver(Self, "adopted", Self);
    }
    public void sent(String source, String message, String dest, String destlaw) {
        if (message.equals("getCS"))
            doDiscloseAllCS();
        else
            doForward();
    }
    public void arrived(String source, String message, String dest, String destlaw) {
        doDeliver();
    }
    public void disconnected() {
        doQuit();
    }
}

Figure 9: Testing law (Java)