1 Introduction

In the context of distributed systems, developers have to spend a significant amount of effort in deploying and controlling all the components of an application. As the working set grows, and the components are heterogeneous (as often happens) the task becomes unmanageable.

The backbone of an LGI system consists of a number of controllers. Any agent connects to a particular controller in order to perform a certain activity. The controllers usually run on different machines, they are started at different times and possibly by different users. 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 Server is a LGI component that acts as a centralized directory for the registered controllers. It maintains updated information regarding the parameters and status of any registered controller. The Controller Server provides program users with a complete message set and human users with a graphical interface in order to request specific information regarding registered controllers. Even though the Controller Server 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 Server as follows: Section 2 describes the general structure of the Controller Server. Section 3 presents the interface provided to general users, while Section 4 presents the functions of the manager interface. Section 5 serves as API reference and presents details on how to run the Controller Server.
2 General Structure

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

- **Web-based users**: The Controller Server has a web-based graphical user interface that allows interaction with two types of users: regular users and managers. The regular users can access information regarding the registered Controllers (details in Section 3) whereas managers can exchange commands with, and configure specific functions of the Controller Server.

- **Program agents**: Using a query/answer protocol, program agents can directly obtain summary information from the controller directory. A complete description of this protocol is provided in Section 5.

- **Native controllers**: The Controller Server 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 Server. The standard output of the controller is passed along this connection.

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

The Controller Server can accommodate any combination and number of users/managers, program agents, and controllers simultaneously.

The rest of the documents will describe the interaction model between Controller Server and each of the entities above.
3 Web Interface

The main function of the Controller Server is to list the controller related information to all the requesting agents. Users can retrieve the controller directory by consulting the Controller Server’s web page. Assuming that the Controller Server runs on the machine alpha on port 1234, its main web page can be accessed through one of the following URLs:

- http://alpha:1234/

Any user accessing the URLs above has unrestricted access to the controller directory information. Figure 2 presents a sample of Controller Server web page. The information inside this page is organized in two tables: 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.
- 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.
- Status: The status of the controller. As a result of periodical testing, the Controller Server updates the status of each controller. Section 4 and Section 5 provide details on the controller testing.
Figure 3: Controller Output View
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 Server.
- **Status**: The status of the controller, tested as above.

Beside 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 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 law developing and testing it is critical to follow the controller debugging information. The Controller Server 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 server 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
4 Manager Interface

The Controller Server accommodates two types of users: regular users who have access to the information described in previous section, and manager users. Manager users can send specific commands and can configure the functionality of the Controller Server 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 Server 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 Server can start.

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

By selecting the Create button of 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:

- **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”
- **Create**: when issuing this command, the Controller connects remotely to the specific hosts (*rsh* 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 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.
Figure 5: Starting and Stopping a Controller
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 Server attempts to kill the controller process. The significance of Reset fields is similar to the one described in Subsection 4.1. A message that indicates the success or failure of the Kill operation is displayed at the bottom of the page.

4.3 Testing Controllers

One of the main functions of the Controller Server is to report the operational status of each registered controller. Although the Controller Server 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 Server directory. However, in order to assess the degree of responsiveness of a controller, further testing is required.

The Controller Server performs a periodical testing, at a interval specified in its configuration (discussed in Subsection 4.4). It also performs a 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
Figure 6: Testing Controllers and Configuring the Controller Server
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 a 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 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 Server

Upon testing a controller, a success/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).

### 4.4 Configuring the Controller Server

The configuration interface allow the manager to retrieve and specify certain generic functions related to the Controller Server (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.
<table>
<thead>
<tr>
<th>Query</th>
<th>Response/Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>query(controllers)</td>
<td>[controller(host(host), port(port)), ...] the list of all registered controllers</td>
</tr>
<tr>
<td>query(native)</td>
<td>[controller(host(host), port(port)), ...] the list of all native controllers</td>
</tr>
<tr>
<td>query(adopted)</td>
<td>[controller(host(host), port(port)), ...] the list of all adopted controllers</td>
</tr>
<tr>
<td>query(controller)</td>
<td>controller(host(host), port(port)) a random selected native controller</td>
</tr>
<tr>
<td>query(a_controller)</td>
<td>controller(host(host), port(port)) a random selected adopted controller</td>
</tr>
</tbody>
</table>

Figure 7: Query/response protocol for program users

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

5 Appendix

This section describes the message interface used in interaction with program users as well as some Controller Server implementation considerations and running details.

5.1 Controller Server message API

Figure 7 presents the message interface a program agent can send to a Controller Server.

The registration message the Controller Server accepts from controllers has the form `system(controller(host(host), port(port)))`.

Any program agent can capture the output of a certain controller by sending the following message to the Controller Server: `output(host(host), port(port), pwd(pwd))`.

All the messages above are sent in String format preceded by 4 bytes integer representing the length of the incoming message.
5.2 Testing Laws

Figure 8 and 9 present the law Controller Server uses for testing controllers with Prolog and respectively Java law engines.

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

\[ http://machine:port/testManager.law \]
\[ http://machine:port/testManager.java \]

---

**Preamble:** PolicyName(testManager)

\[ R1. \text{sent}(X, \text{command}(Y), Z) :\]
\[ \quad \text{do(\text{command}(Y))}. \]

\[ R2. \text{sent}(X, \text{test}, Y) :\]
\[ \quad \text{do(\text{deliver}(X, \text{passed}, X))}. \]

---

Figure 8: Testing law (Prolog)

5.3 Configuration file

In order to run the Controller Server has to take its initialization information from a configuration file. Figure 10 presents such a configuration file model:

- The fields *host* to *pass* represent the default values a user manager encounter when attempting to create a new controller.

- The field *testperiod* represent the testing time interval (in seconds).

- The manager user-names and passwords follow the keyword *accounts*. While the user-names are stored in plain text, the password are stored in “scrambled - hash” format.

- Any line starting with the symbol “#” is considered to be comment.
public class testManager extends Law {
    public testManager() {
        PolicyName("testManager");
        InitialCS("");
    }
    public void sent(String source, String message, String dest) {
        if (message.equals("command(getCS)")) {
            doGetCS(); return;
        }
        if (message.equals("test")) {
            doDeliver(source,"passed",source); return;
        }
    }
}

Figure 9: Testing law (Java)

host = moshe.rutgers.edu
port = 9000
path = /farm/moses/implicit/controllerConf/
pack = implicit.moses.controller.GenericController
type = Prolog/Java
pass = public
testperiod = 1500
accounts =
user(moses) pass(52BD43D37ED62EB4C226E31841BC03DC)
user(anonymous) pass(6540BD43D37ED62EB4C226E180187GH4)
# "52BD43D37ED62EB4C226E31841BC03DC" represents the hash for "moses"

Figure 10: Controller Server Configuration File
5.4 Starting the Controller Server

A typical starting line for Controller Server looks like:

```
java ControllerServer 9025 -pserverConf -fcfg
```

The arguments required in Controller Server starting line are

- **port**: specifies the port where the Controller Server waits for TCP/IP or HTTP requests. Default value 9025, must explicitly specified.

- **-ppath**: path represents the directory used to store different necessary files. The configuration file (see below) is identified relatively to this path. Other files that Controller Server expects to find in this path are the manager and output applet class files. If not specified, current directory is considered.

- **-fcfgfile**: cfgfile represents the configuration file as discussed in Section 5.3 and Figure 10.

The Controller server is designed to run in a environment with several properties. First, the machine running the Controller Server should be able to execute "rsh" on remote computers. More than that, the remote computers should be in the same administrative domain, thus allowing execution of rsh 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 Server host. As a test case to assess the ability of a Controller Server to start native controllers, the following command should be verified:

```
rsh host java classname args ...
```