JLISTEN

PROTOTYPE USER MANUAL

hexagun

Team Members

Vijaya Ganesh.V [Team leader]

Jagadish Prasath R

Pradap K V

Mudit Mathur

Gopinath M C

Nageswar Rao Katta

B.I.T.S Pilani, India

Client

Prof. Aditya P Mathur.

Purdue University. USA

TABLE OF CONTENTS

TABLE OF CONTENTS

JLISTEN PROTOTYPE USER MANUAL

OVERVIEW:

JListen is a toolset for auralisation of a java program. Using this toolset, the programmer can specify the activities, events to be auralised. This involves a specification called Listen Specification Language. Based on the LSL, given java program is auralised and sound signals are generated as and when program is executed and event, activities occur during the execution of program. The prototype implements overall architecture of this toolset.

ARCHITECTURE:

Jlisten has three components.

- Auraliser
- Configuration Server
- Listener

AURALISER:

Auraliser has an *Editor* to make LSL command file and an *Instrumenter* to Instrument the Java source file according to the LSL Specifications.

Using the Editor, a programmer can specify *Events, Activities* to be listened for a particular scope (*entire program, specific class, function*) and *sound variables* to signal the occurrence of *events*.

Instrumenter instruments specified Java Source file according to the LSL Specification. This involves inserting of calls to event generation routines, which in turn delegates requests to Configuration Server.

CONFIGURATION SERVER:

The decorated java code when run by the *auraliser* sends *event signals* to the *Configuration Server*. This in turn delegates *events* or *sound generation requests* to all registered *Listen Sound Servers* distributed across the globe.

Configuration Server also takes care of registration of different Listeners for the list of available programs. Once registration is done, the *Event-Sound mapping* (.LSS) file is sent to the Listener.

Whenever a Listener logs-in (getting connected to Configuration Server), Configuration Server sends the list of Registered Programs and New programs to Listener. Listener in turn can register any new program or unregister registered programs.

LISTENER

Listen Sound Server waits for event requests from Configuration Server. When an event signal arrives, Listener checks for the particular program, the event’s current status (ON/ OFF), and its default sound variable or current sound variable (if default sound variable is overridden by current sound variable) and sends signals to synthesizer routines which in turn generates sound accordingly.

Listen Sound Server can listen to more than one program from different Configuration Servers at any point of time.

During the execution of a program (i.e., in run time), the user can change the sound mappings for any event in a program or even enable/ disable an event.

JLISTEN TOOLKIT:

Jlisten Toolkit has three components in correspondence with Jlisten architecture.

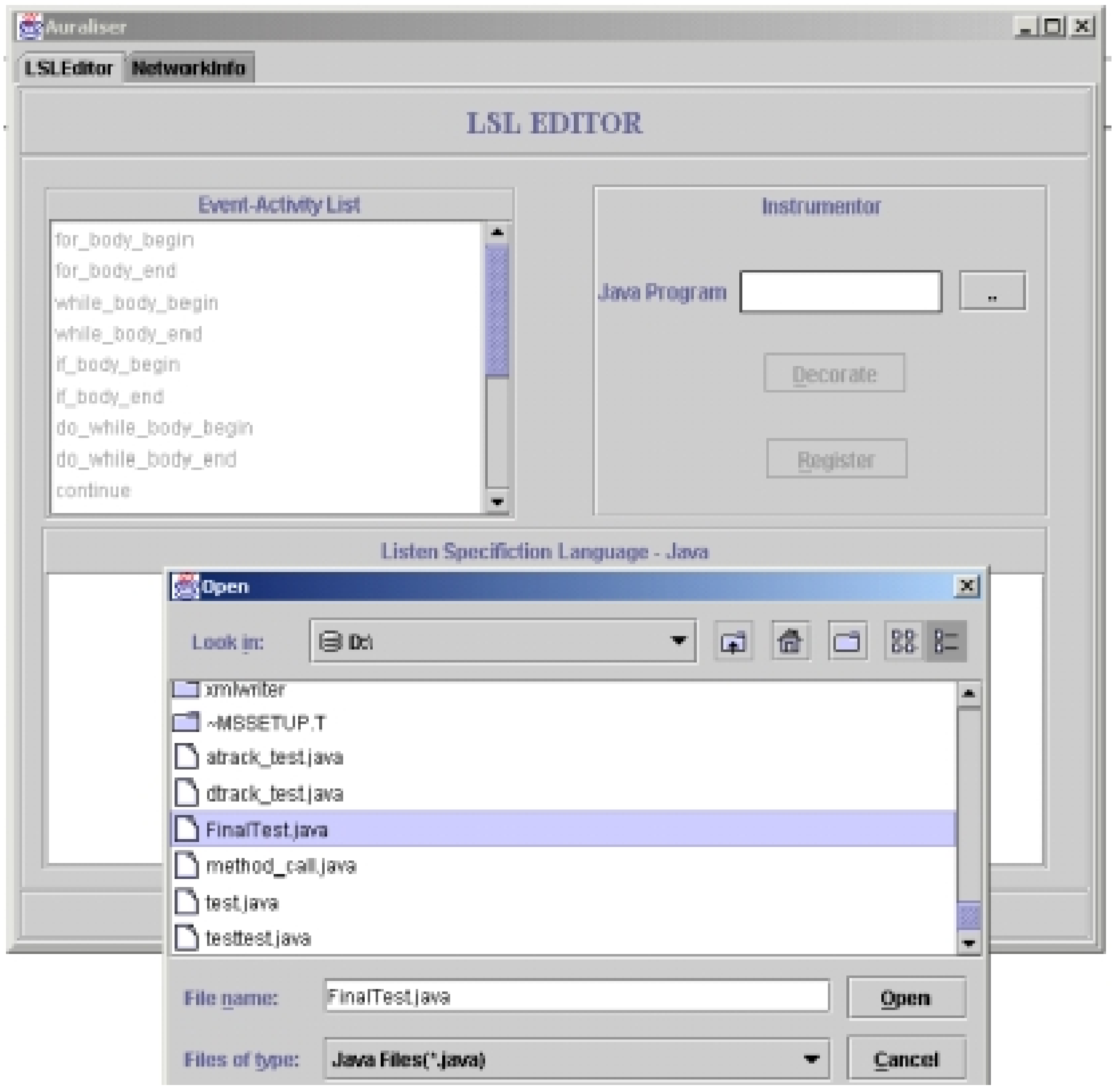
- Auraliser
  - LSL Editor
  - Instrumenter
- Configuration Server
- Listen Sound Server

AURALISER:

LSL Editor:

Figure shown below is used to make LSL commands.

The user has to select the **java** program that need to be *auralised.*



Event-Activity List: for_body_begin, for_body_end, while_body_begin, while_body_end, if_body_begin, if_body_end, do_while_body_begin, do_while_body_end, continue

Look in: D:\

xmlwriter
~MSSETUP.T
atrack_test.java
dtrack_test.java
FinalTest.java
method_call.java
test.java
testtest.java

File name: FinalTest.java

Files of type: Java Files(*.java)

FIG 1: INSTRUMENTOR – SELECT JAVA PROGRAM

Adding LSL Instructions: To add an LSL command, click any ***event/ activity*** in the ***event/ activity pane***, a window is shown with required ***LSL Parameters***. The fields in the window are according to the ***activity/ event*** selected.

As shown in the figure below, the user has to select values for ***Scope***, and the actual values that he wants to ***auralise***, along with ***Instrument*** names.

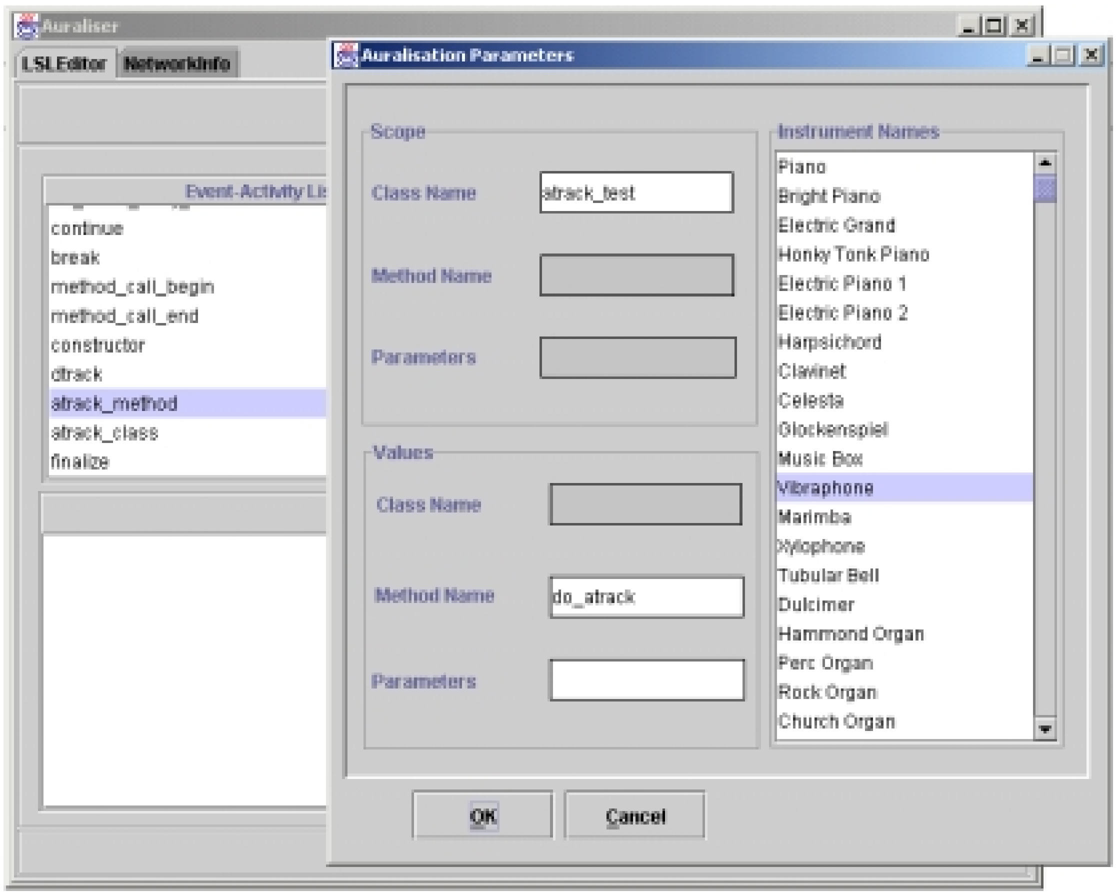
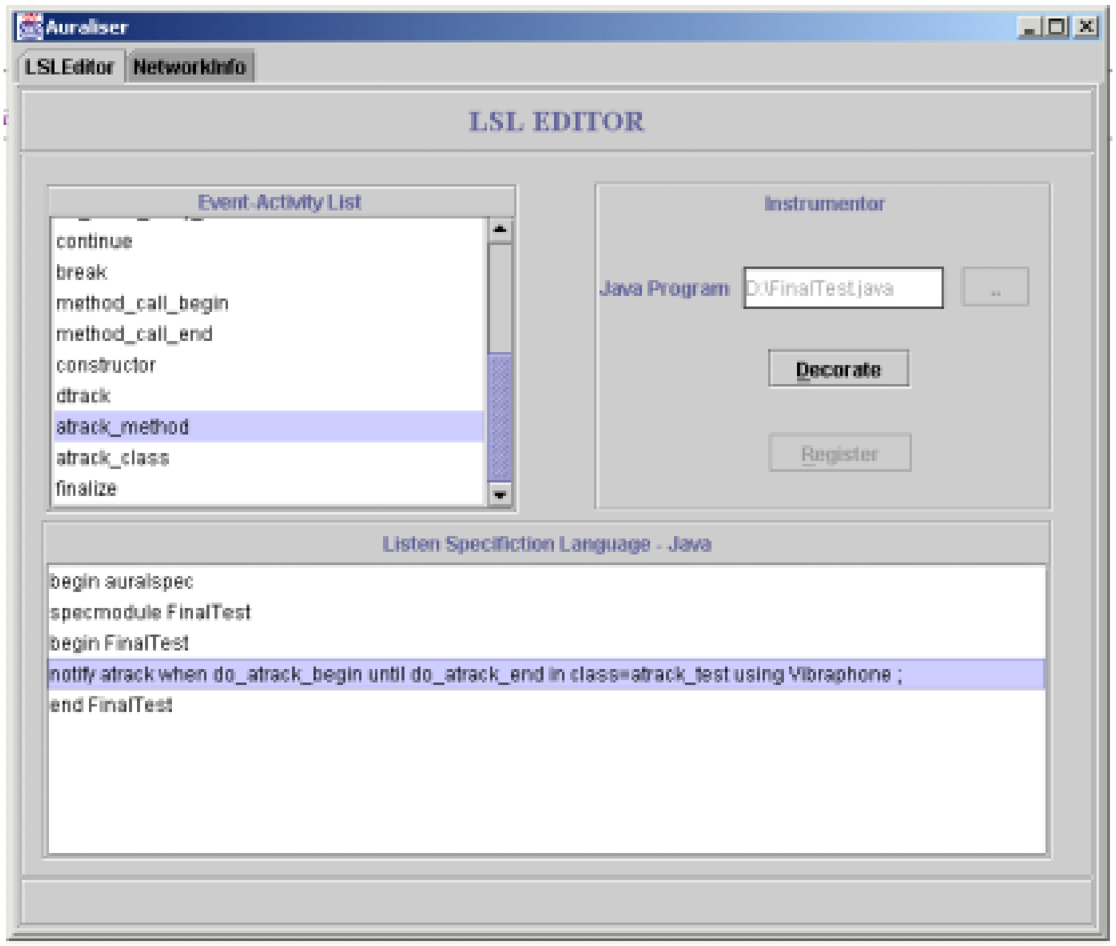


FIG 2: LSL EDITOR – COMPOSE LSL COMMAND

Once the user fills the *LSL Command parameters*, LSL command is generated and shown in ***LSL Editor***.



continue
break
method_call_begin
method_call_end
constructor
dtrack
atrack_method
atrack_class
finalize

D:\FinalTest.java

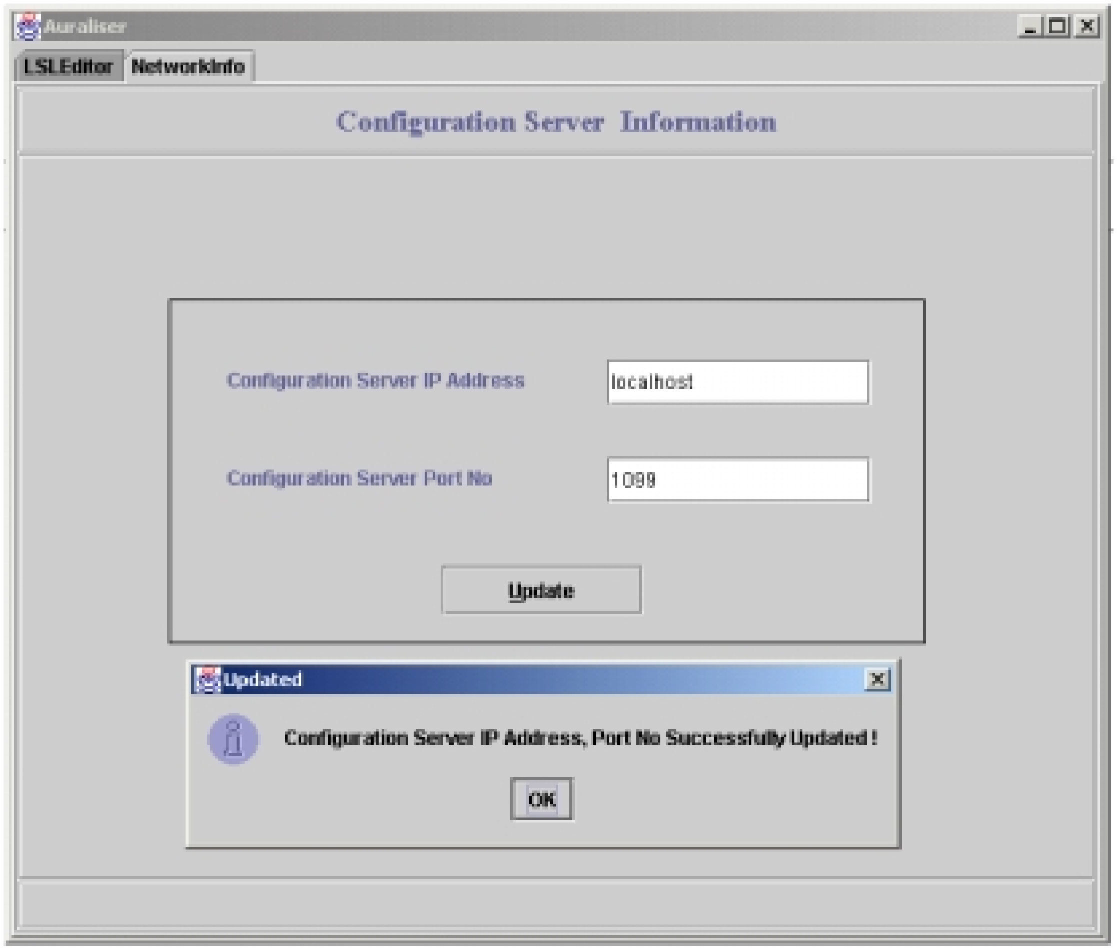
```
begin auralspec
specmodule FinalTest
begin FinalTest
notify atrack when do_atrack_begin until do_atrack_end in class=atrack_test using Vibraphone ;
end FinalTest
```

FIG 3: LSL EDITOR – LSL COMMANDS PANE

Instrumentor:

After adding the LSL Instructions, the user can ***Decorate*** the java program according to the *LSL instructions* specified.

Once ***Decoration*** is done successfully, **Register** button is enabled. [Register button is enabled only when; *Configuration Server* details are filled in **Network Info** Pane]. If Network info is not filled, the pane is shown as follows.



Configuration Server IP Address: localhost

Configuration Server Port No: 1099

Configuration Server IP Address, Port No Successfully Updated !

FIG 4: AURALISER - NETWORK INFORMATION PANE

The user has to register IP address and Port No. of **Configuration Server**.

Registration of Program:

Once the program is ***auralised*** successfully, **Register** button will be enabled in the ***Instrumentor*** panel. User can register the program at the ***Configuration Server*** whose *IP Address and Port No.* are given in the ***Network Info Tab.***

CONFIGURATION SERVER:

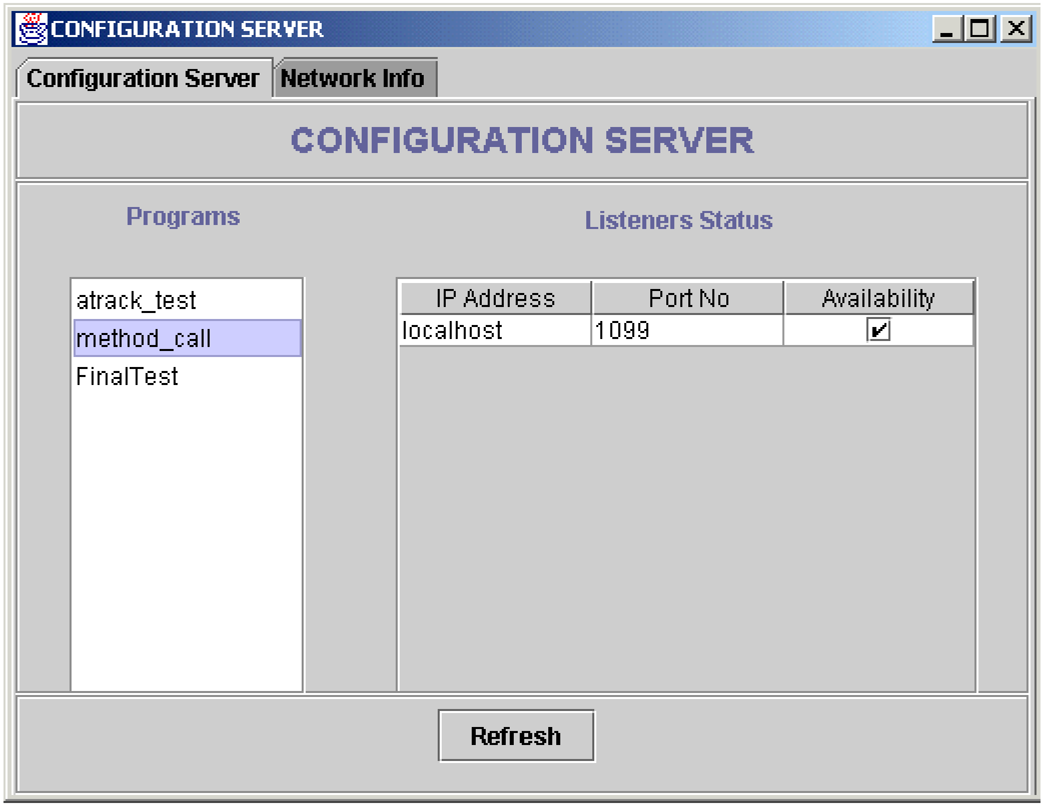
Configuration Server acts as mediator between Auraliser and Listeners.

Configuration Server has the list of *decorated programs* (".PL Files" – ProgramList) for which it delegates event calls to registered Listeners. This ".PL" file is updated whenever a *Program* from *Auraliser* registers at *Configuration Server.*

It displays the list of *registered Listeners*, its current status (ON/OFF) for a particular *program*. This list is taken from a file (".MAP_FILE").

Whenever a *Listener* logs-in, *Configuration Server* responds with a list of *Registered programs, New programs*. If new programs are *registered*, corresponding *event-sound mapping files* (".LSS Files") are transferred to the Listener.

If an *event* request is sent by *auraliser* for *particular program*, *Configuration Server* checks from the list of registered *Listeners* in that particular Configuration File (".MAP_FILE" Files with name *<program-name>.MAP_FILE*) and sends to those registered *Listeners* that are logged-in currently.



atrack_test
method_call
FinalTest

IP Address	Port No	Availability
localhost	1099	✓

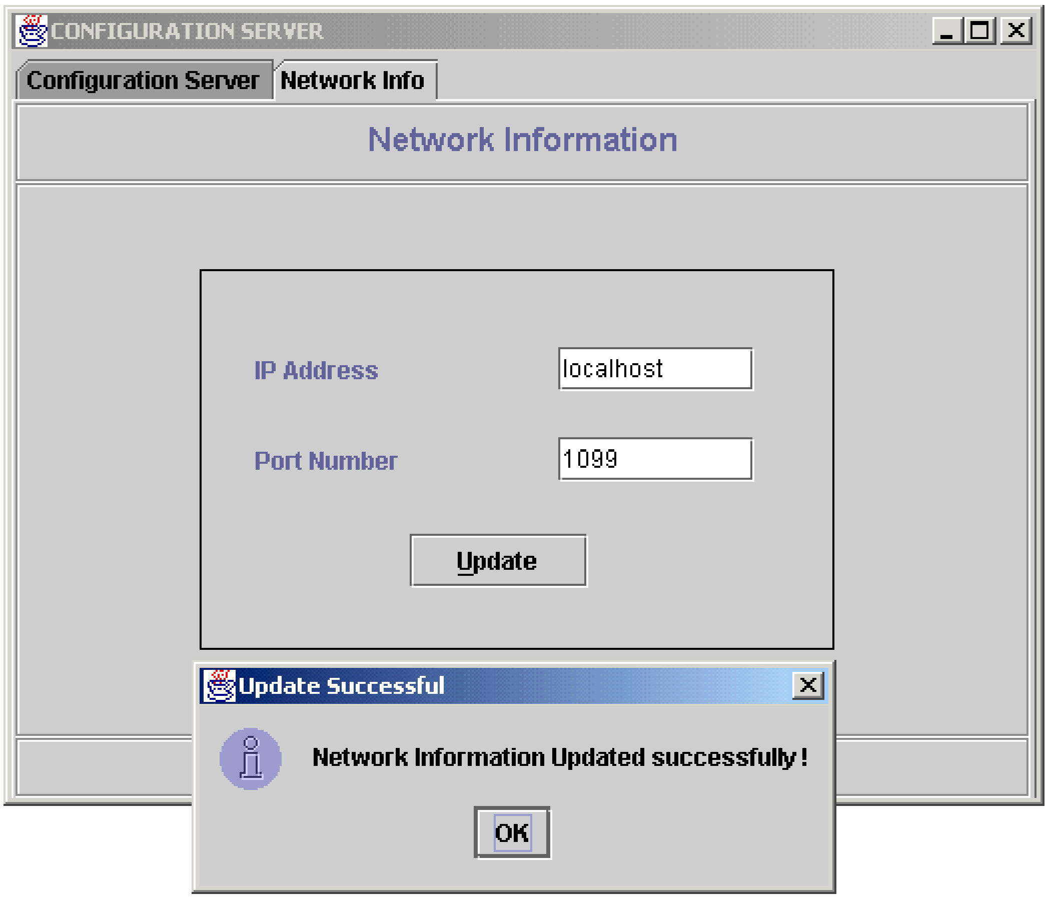
FIG 5: CONFIGURATION SERVER – PROGRAMS LIST & LISTENER STATUS PANE

By selecting a *Program* from ***Programs List box***, user can view the list of *listeners* registered for the *program.* The list shows the *IP Address and Port No* of the *listeners*. This list also shows the ***status*** of the *listeners*, whether they are currently *logged-in* or not.

After the *Configuration Server* has started, *New programs* might have been registered. To view the latest updations, the user can press ***Refresh*** button that will display the entire set of *programs* and their *listeners'* current status.

Configuration Server - Network Info

When the *Configuration Server* is started for first time, user has to fill-in details about IP Address, Port No., so that *Auraliser, Execution platform, Listener* can communicate with the *Configuration Server* through the IP Address, Port No.



IP Address: localhost

Port Number: 1099

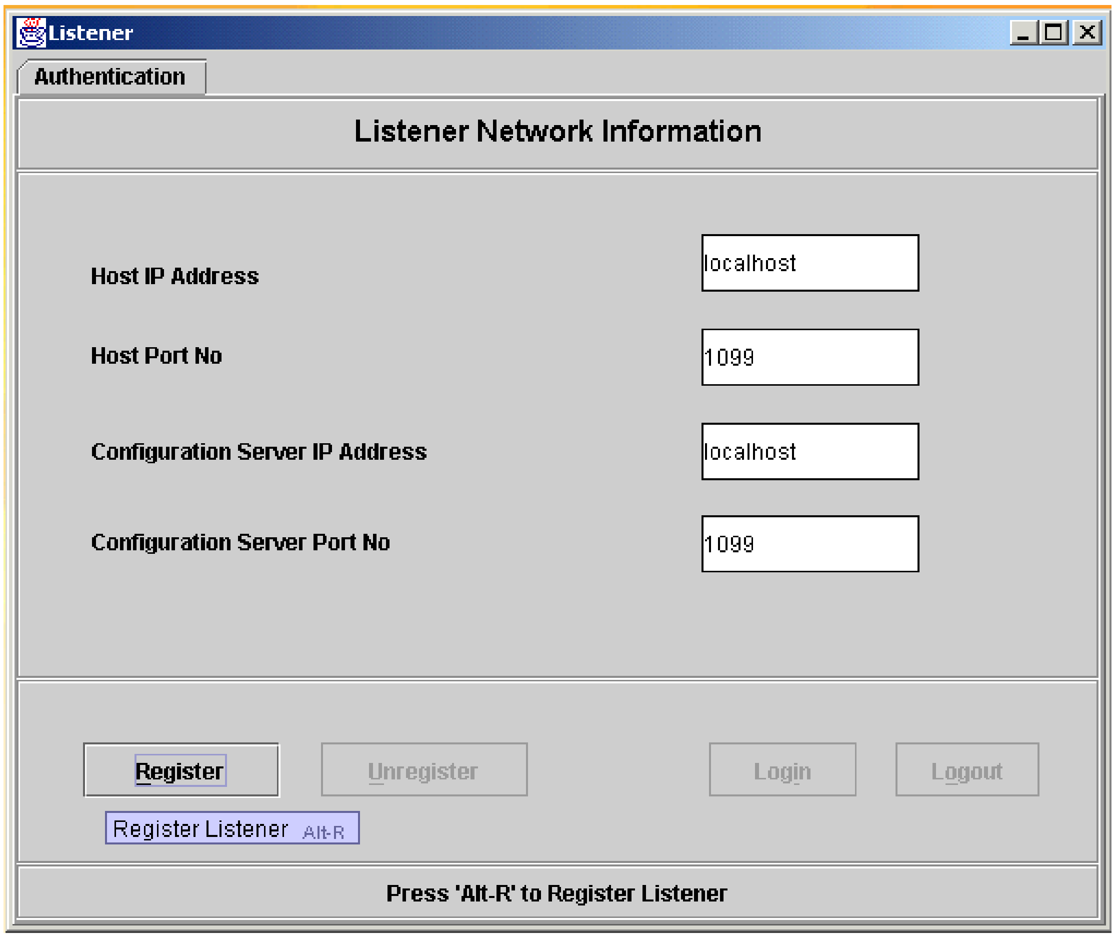
Network Information Updated successfully !

FIG 6: CONFIGURATION SERVER – NETWORK INFORMATION PANE

LISTENER:

Listener has to register at *Configuration Server* to receive events.

Every *Listener* has to login with its identity so as to get connected with the *Configuration Server*.



Host IP Address	localhost
Host Port No	1099
Configuration Server IP Address	localhost
Configuration Server Port No	1099

Press 'Alt-R' to Register Listener

FIG 7: LISTENER – REGISTRATION & LOGIN-LOGOUT PANE

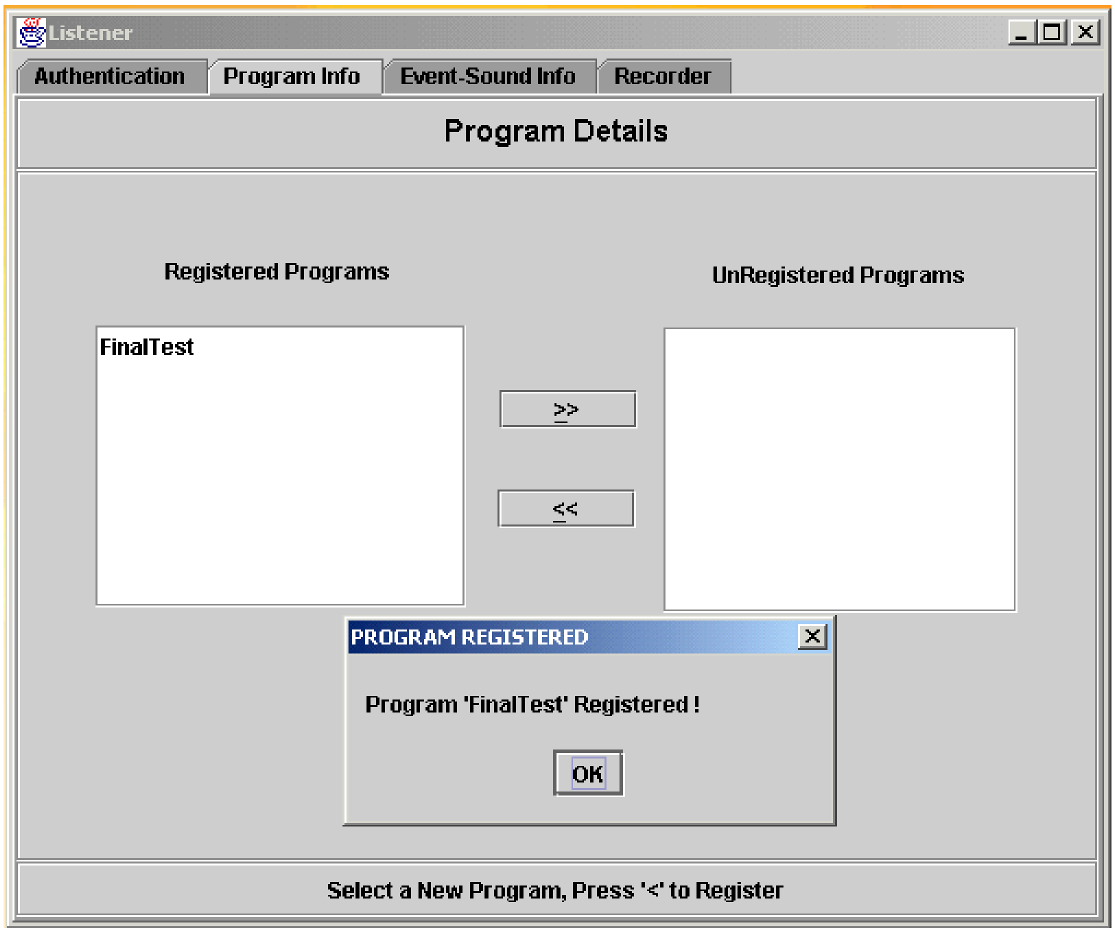
Program Registration:

Once Listener is registered, it can login at Configuration Server and register for programs.

Once the connection is established between Configuration Server and Listener, Listener receives the list of *registered and new decorated programs*.

The lists of registered, new programs are shown in the ***Program details pane*** shown below.

User can *register new programs*, in which case, corresponding ".LSS" files are received from *Configuration Server*. Listener can *unregister* any existing *programs* after which it will not receive any *events* for that particular *program*.



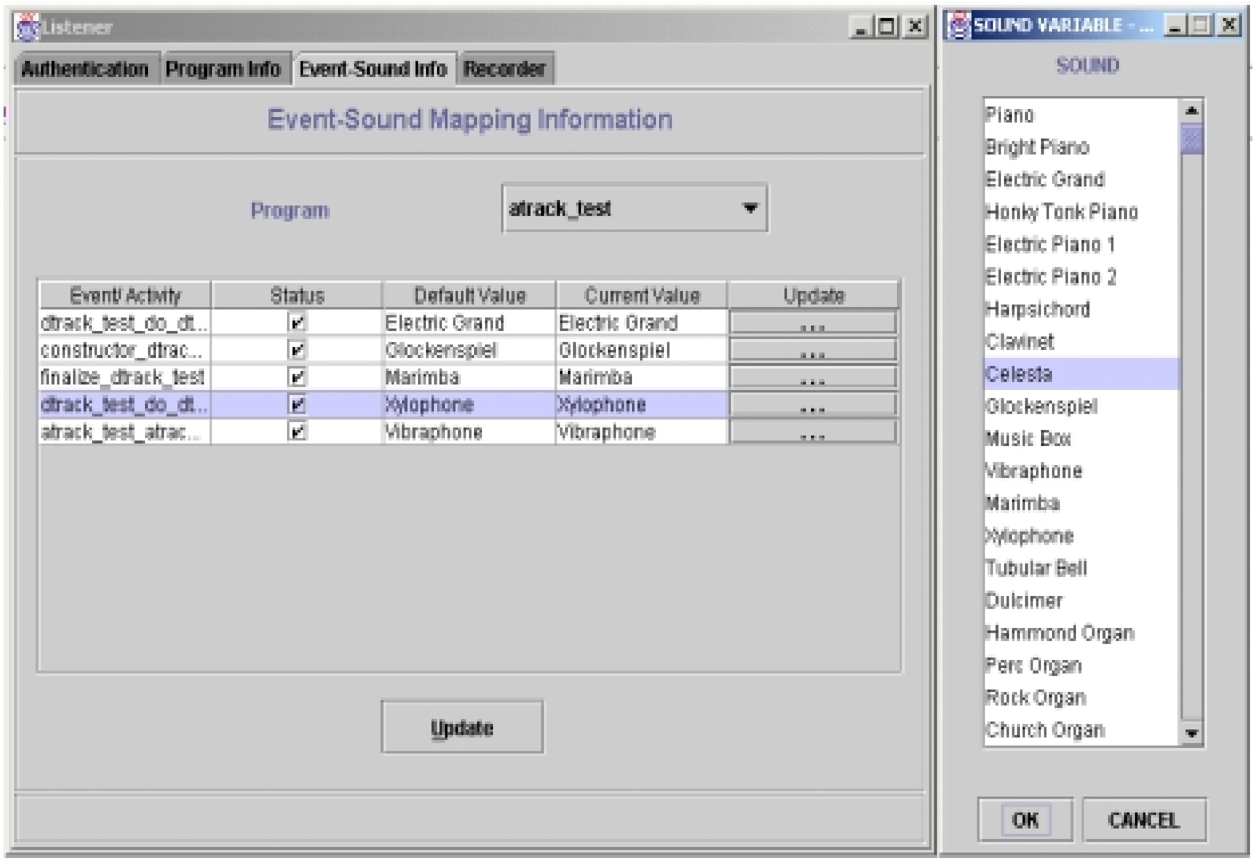
Program 'FinalTest' Registered !

Select a New Program, Press '<' to Register

FIG 8: LISTENER – PROGRAM SELECTION PANE

Event- Sound Mapping:

User can select and view the event-sound mapping (".LSS") files for any *registered program*. This mapping shows the *list of events*, corresponding *default sound mappings*, and *current sound mappings*. The user can modify the *sound mappings, enable/ disable* an *event* even at run-time.



Event/ Activity	Status	Default Value	Current Value	Update
dtrack_test_do_dt...	✓	Electric Grand	Electric Grand	...
constructor_dtrac...	✓	Glockenspiel	Glockenspiel	...
finalize_dtrack_test	✓	Marimba	Marimba	...
dtrack_test_do_dt...	✓	Xylophone	Xylophone	...
atrack_test_atrac...	✓	Vibraphone	Vibraphone	...

Piano
Bright Piano
Electric Grand
Honky Tonk Piano
Electric Piano 1
Electric Piano 2
Harpsichord
Clavinet
Celesta
Glockenspiel
Music Box
Vibraphone
Marimba
Xylophone
Tubular Bell
Dulcimer
Hammond Organ
Perc Organ
Rock Organ
Church Organ

FIG 9: LISTENER – EVENT SOUND MAPPING PANE

Recorder:

Programs can be recorded. Whenever events come from Configuration Server, they will be recorded and can be played offline. The ***Recorder Pane*** is used to set recording options for a program. User can select a *Program* to *record*, deselect a *recorded program.*

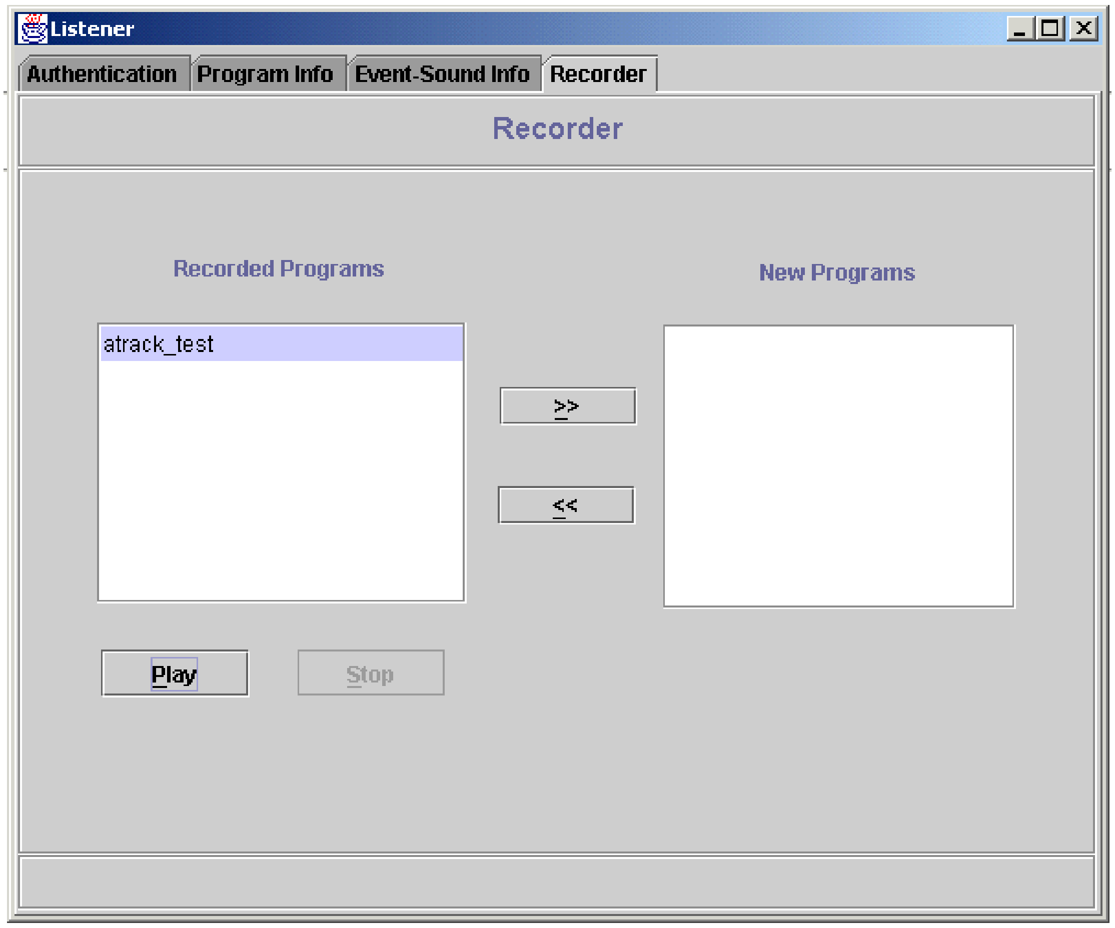


FIG 10: CONFIGURATION SERVER – RECORDER PANE

When the ***Play*** button is pressed, the *recorded program* is played. All the Events received and recorded by the ***Recorder*** are played again. The user can change the ***Event Sound Mappings,*** and recorder will play sounds according to the current settings.

If the ***program*** is not recorded, message will be shown as “Program Not yet recorded”. Once the ***program*** is recorded, the user can play it.

Once the Listener finishes its operations, it logs out of the Configuration Server, either by closing the window or by pressing ***Logout*** button in login pane.

DEPLOYMENT INFORMATION:

SOFTWARE REQUIREMENTS:

Jdk1.3 or above.

SOURCE FILE STRUCTURE:

Various components of J-Listen are
Auraliser(Instrumentor), Configuration Server, Listener

The files include

COMPONENT	Source Code	Batch File	RMI Registry Starter
Instrumentor	Instrumentor.zip	Instrumentor.bat	NOT REQUIRED
Configuration Server	ConfServer.zip	ConfServer.bat	RmiRegistry_ConfServer.bat
Listener	Listener.zip	Listener.bat	RmiRegistry_Listener.bat

We have used third party source code for parsing and instrumentation. (http://www.glenmccl.com/instr/instr.htm).

BATCH FILES:

Batch files are provided separately to start the individual component. (Auraliser, Configuration Server, Listener).

Batch file contents need to be changed according to your jdk path and application(source file) path.

$JAVA_HOME variable represents Java Path eg: d:\jdk1.3
$APPLICATION_PATH variable represents Base directory e.g: d:\flat_package\

INSTRUMENTOR

Instrumentor.bat

Contents:

```
cd $APPLICATION_PATH\instrumentor\
set path=$JAVA_HOME\bin
set classpath=$APPLICATION_PATH\instrumentor\
java view.AuraliserUI
```

CONFIGURATION SERVER

RmiRegistry_ConfServer.bat

Contents:

```
set path=$JAVA_HOME\bin
set classpath=$APPLICATION_PATH\confserver\
start $JAVA_HOME\bin\rmiregistry
```

ConfServer.bat

Contents:

```
cd $APPLICATION_PATH\confserver\
set path=$JAVA_HOME\bin
set classpath=$APPLICATION_PATH\confserver\
java view.ConfigurationServerUI
```

LISTENER

RmiRegistry_Listener.bat

Contents:

```
set path=$JAVA_HOME\bin
set classpath=$APPLICATION_PATH\listener\
start $JAVA_HOME\bin\rmiregistry
```

listener.bat

Contents:

```
cd $APPLICATION_PATH\listener
set path=$JAVA_HOME\bin
set classpath=$APPLICATION_PATH\listener\

start java view.MainFrame
```

EXECUTION PLATFORM:

Once the program is auralised, decorated code will be created and stored in the following folder

JLISTEN\Decorated\PROGRAM_NAME\PROGRAM_NAME.java

The exact location of JLISTEN Folder depends on the $APPLICATION_PATH.

Eg:

Consider the auralisation of the following program: ***dtrack_test.java***

If the application resides in *d:\flat_package\**instrumentor***

Then decorated code will be created in,

D:\JLISTEN\Decorated\dtrack_test\dtrack_test.java

To run the decorated code in command prompt,

Set the path for jdk eg: ***set path=c:\jdk1.4\bin***

Compile **dtrack_test.java** as ***javac dtrack_test.java***

Run **dtrack_test** as ***java dtrack_test***

Execution of the Auralised program will send events to Configuration Server, which in-turn will delegate events to listeners that are logged in at that moment.

ORDER OF INVOCATION:

- *RmiRegistry_ConfServer.bat* at Configuration Server.
- *ConfServer.bat* (Configuration Server need to be up first.)

- *Instrumentor.bat* (Auraliser after decoration, can register program at Conf Server)

- *RmiRegistry_Listener.bat* at Listener
- *Listener.bat* (After program registration, Listener can login and register/ unregister programs)

Then, the decorated code can be executed as explained in BATCH FILES – EXECUTION PLATFORM.

Auralisation Of Sample Programs

Program 1:

/*

atrack_test.java

*/

```
public class atrack_test
{
	public atrack_test()
	{
	}

	public void do_atrack()
	{
		try
		{
			Thread.sleep(2000);
		}
		catch(Exception e)
		{
		}

		int i=0;
		for(;i<100;)
		{
			i+=10;
		}

		try
		{
			Thread.sleep(2000);
		}
		catch(Exception e)
		{
		}


	}


	public static void main(String args[])
	{
		atrack_test objDT = new atrack_test();
		objDT.do_atrack();
	}
```

In the above program, the ***Instrumentation*** points are,

- Data track of ***integer variable "i"***
- Activity track of ***method "do_atrack()"***

Auralization Procedure:

Select "atrack_test.java"

Data track of "i":

- Click on "dtrack" in "Event-Activity List"
- Enter the following details in "Auralisation Parameters" Window

  In SCOPE Panel,
  Class Name: atrack_test
  Method Name: do_atrack

  In VALUES Panel,
  Identifier : i

- Select an INSTRUMENT
- Press OK.

Activity Track of "do_atrack":

- Click on "atrack_method" in "Event-Activity List"
- Enter the following details in "Auralisation Parameters" Window

  In SCOPE Panel,
  Class Name : atrack_test

  In VALUES Panel,
  Method Name: do_atrack

- Select an INSTRUMENT
- Press OK.
- Press Decorate in MAIN WINDOW
- Press REGISTER to register program with ***Configuration Server***.
- Start Listener Component, REGISTER and then LOGIN
- Check for New Programs.
- Select a program for Registration.

Whenever *auralized program* is executed, *events* will be sent to all the registered *Listeners.*

Program 2:

```
public class method_call
{
	public void method_loop()
	{
		for(int iLoop=0; iLoop<10; iLoop++)
		{

		}
System.out.println("Test");

	}



	public static void main(String args[])
	{
		method_call objMC = new method_call();
		objMC.method_loop();

	}

}
```

In the above program, **method call “method_loop”** is the instrumentation point.

Auralisation Parameters:

In VALUES Panel,

Method Name: objMC.method_loop