Oxygen Alliance 2003 Workshop – February 24-28, 2003

Metaglue Tutorial

How to get Metaglue

Tablets

For tablets (and other computers) there is more setup than for the iPaqs. These are the base stations which will run most of the agents we'll be working with simply because they have more power than the iPaqs.

First get the packages needed and install them. Unfortunately the first one is a mouthful.

```
> wget http://server/feeds/debian/lib-rxtx-java_2.1.6-1ai_i386.deb
> dpkg -i lib-rxtx-java_2.1.6-1ai_i386.deb
```

Now the easier one. Answer yes to both questions about MySQL. You'll need it running when using Metaglue

```
> apt-get update
> apt-get install metaglue
```

Database setup

Setup can be a bit more complicated. We will need to setup a MySQL database tohold the information that comes out of the Metaglue agents. Fortunately, all but one of the defaults will be acceptable for this class.

Start the database builder

```
> mg_agent util.BuildDatabase
```

The question to look out for is the third question, "What user name is allowed to create databases?" The default is "roor", but that is blocked by the MySQL server.

You should enter "metaglue".

There is no password for this user, so the next box should stay blank. Then accept the rest of the defaults. Passwords for creating a login to the database are optional

You have reached the end of the database setup when you are presented with a URL for how to reach the database. Save this for later steps!

Configuration

Now we get to the last step of the setup for the tablets. Configure the metaglue system using

```
> mg_config
```

This will prompt you for the database URL which you got from the previous steps. Enter it as is, but without any double quotes ("). Yes, that password you used from up above will be plain text and in your environment, but it is only meant to be protection against accidental database changes.

Activity Source Code

When you installed the Metaglue debian package, it also pulled in some source code for you to work with. Check this into your home directory.

```
> cd $HOME
> get_metaglue
```

This will create a directory $HOME/metaglue where there will be source java files and some Makefiles. See the Compiling and Running section of the References for information on compiling the java files.

iPaqs

For iPaqs, the setup is incredibly simple. You need to install the metaglue package:

```
> ipkg update
> ipkg install metaglue
```

and answer the question “What is the database connection URL?” This is the same url that you should have gotten from the tablet setup. Log out and then back in and you’re ready for business!

Tutorial 1 – Writing a simple agent to control a lamp

This goal of this tutorial is to provide an introduction to Metaglue agents. In this tutorial, we will learn how to use other Metaglue agents and how to use Metaglue’s notification mechanism to respond to messages from other agents. We show how to do these things by writing a simple agent, called `X10BasicLightControlAgent`, that allows us to programmatically turn on and turn off an X10-enabled lamp. The code for a gui with an “On” and an “Off” button is provided; we will write the code that turns the lamp on and off when the buttons on the gui are pressed.

Part 1: Getting a handle to other Metaglue agents.

Goal

In this part of the tutorial, we will learn how to get the RMI stub of another Metaglue agent. As an example, we will write the code to get the RMI stub for the Metaglue agent that provides the programmatic interface to our lamp. We will use this agent later in the tutorial to turn on and off the lamp when the “On” and “Off” buttons on the agent’s gui are pushed.

Basic Information

If you look at the beginning of the code for the `X10BasicLightControlAgent`, you will see a member variable of type `X10Light` named `x10`. This variable is the RMI stub for the actual agent that will turn on and turn off our lamp. Before we can make method calls on this agent, we need to first initialize it in the `X10BasicLightControlAgent`’s constructor.

Right now, the constructor for this agent looks like this:

```
public X10BasicLightControlAgent() throws RemoteException {
// initialize x10 variable here
}
```

To initialize the x10 variable, we need to write only one line of code. This line of code is a method call that will return the RMI stub of an X10 Light agent. There are two things to keep in mind when writing this one line of code:

1. When calling this method, the argument to the method call should be X10Light.class
2. You will need to cast the result of the method call as an X10Light

So, the line of code you write should have the following form:

```
x10 = (X10Light) someMethodCall(X10Light.class);
```

Coding Exercise

Initialize the `x10` variable in the `X10BasicLightControlAgent` constructor.

Part 2: Responding to notifications and using other agents

Goal

In this part of the tutorial we will learn how to respond to Metaglue notifications and how to call methods belonging to other agents. We will write code that uses the `X10Light` agent from Part 1 to turn on and off the lamp when we receive a Metaglue notification that the user pressed the "On" or "Off" button on the gui.

Basic Information

Now that the `x10` variable is initialized, we will use it to respond to Metaglue notifications. Notice in the `startup()` method that the `X10BasicControlAgent` uses the `addSpy` method to subscribe to two different notifications: `newbie.tutorial.LightControlGui.on` and `newbie.tutorial.LightControlGui.off`. The first type of notification is sent out by this agent's gui when the "On" button is pressed; the second type of notification is sent when the "Off" button is pressed. In this part of the tutorial, we will respond to these notifications by telling our `X10Light` to actually turn the lamp on or turn the lamp off.

By default, notifications are processed in an agent's `tell(Secret s)` method. To process the two notifications our agent subscribes to, we will write this method.

There are two things we need to know to fill in this method.

Important fact #1: We need to determine what type of notification we have received. That is, did we receive a `newbie.tutorial.LightControlGui.on` notification, or did we receive a `newbie.tutorial.LightControlGui.off` notification? To do this, we will need to use the `Secret.isA` method. This method takes in one argument, a `String` that names the type of notification we are checking for. So, for example, the following line of code:

```
mySecret.isA(‘‘newbie.tutorial.LightControlGui.on’’);
```

would return the boolean `true` if `mySecret` was a `newbie.tutorial.LightControlGui.on` notification. If `mySecret` was some other type of notification, the code will return `false`.

Important fact #2: Once we have determined what type of notification we have received, we can tell our `X10Light` variable, `x10`, to respond accordingly. There are two methods from `X10Light` that we need: `turnOn()` and `turnOff()`. These two methods do the obvious things; `turnOn()` will turn on our lamp and `turnOff()` will turn the lamp off.

Coding Exercise

Write the `tell(Secret s)` method in the `X10BasicLightControlAgent`. In this method, you should turn on the lamp when you receive a `newbie.tutorial.LightControlGui.on` notification and turn off the lamp when you receive a `newbie.tutorial.LightControlGui.off` notification.

Running this agent

Now we get to try out what you've been working on!

Compile first!

Make sure there is a catalog started! Then you can start this agent either on the command line:

```
mg_agent <your society> <your catalog host> newbie.tutorial.X10BasicLightControl
```

or thru the PowerTester

```
mg_agent <your society> <your catalog host> agentland.debug.PowerTester
```

If you use this second method, enter the agent name in the window that comes up.

Attributes to enter:

houseCode for X10Light: moduleCode for X10Light: These will be on the Light module and will be given during the class. The house and module codes are used by X10 devices to differentiate the modules that are plugged in.

real agent id for agentland.output.X10:

```
agentland.output.X10_CM11
```

The agent in agentland.output.X10 is not one that can be created. It is abstract. Instead, there needs to be an agent that provides the X10 interface. Here it is X10_CM11. This is the particular module

host for X10_CM11: This is the machine that is connected to an X10 computer interface. You will find out what machien it is during the class. You will be asked to start a MetaglueAgent on this machine if it is not currently running. SSH into this machine and start an empty Metaglue platform:

```
mg_agent <your society> <your catalog host>
```

port for `X10_CM11`: `/dev/ttyS1`

This is where the X10 device is plugged in.

Now you can start playing!

Part 3: Running Agents on Multiple Platforms

Goal

The gui for our agent can run move from one host to another. To test this functionality, we will use the `PowerTester`, found in the `agentland.debug` package.

Basic Information

Start up the PowerTester (see the Agent Running Reference below) and enter

```
newbie.tutorial.X10BasicLightControl
```

if you have not done so before. You will need a new window for this.

Make sure that you already have a Metaglue agent (any Metaglue agent will do) in the same society and using the same catalog already running on the new host. If this is not the case, you can start up an empty Metaglue platform by typing the following on the command prompt after loging in by SSH to the new host:

```
mg_agent <your society> <your catalog host>
```

Wait for the Metaglue platform to start on the new host, push the "Call" button on the window that lists the methods exposed by the `X10BasicLightControl` interface. The agent's gui will migrate to the new host.

By default, the gui will start on the host that it last appeared. To move the gui back to its original host, type the name of the original host in the textbox of the window that lists the methods exposed by the `X10BasicLightControl` interface and hit the "Call" button.

Coding Exercise

Use the `PowerTester` to experiment with the `setGuiDisplay` method.

Tutorial 2 – Crickets and Metaglue

Goal

Now we will apply our knowledge of Metaglue notifications to process location notifications sent by Crickets. We will write code to print a message to the console whenever we move to a different location.

Basic Information

The agent that reads from the cricket is called `ipaq.cricket.GoodCricketListener`. Whenever the cricket moves from the area of one cricket beacon to that of another, it sends out a notification called `cricket.location.beaconChanged`. The `Secret` contains the `AgentID` of the cricket beacon. To get this `AgentID`, use the `Secret.details()` method. This method returns a `java.lang.Object`, so you will need to cast the result to an object of type `AgentID`.

To get the actual agent representing the beacon, we can call `reliesOn` with the `AgentID` that we just got as the parameter to the `reliesOn` call. You will need to cast the returned agent to be of type `agentland.device.cricket.CricketBeacon`. Note that this is very similar to what we did in Tutorial 1. The only major difference is that in this section, we are calling `reliesOn` with a parameter of type `AgentID` instead of a parameter of type `Class`.

The beacon agent, `CricketBeacon`, that the `reliesOn` call returns has two methods: `getBeaconName()` and `getBeaconLocation()`. These methods return a `String` that is either the name or the location of the beacon agent.

Coding Exercise

Open the file CricketLearnAgent.java.

Write the `tell(Secret s)` method of `CricketLearnAgent`. When you receive a new `cricket.location.beaconChanged` notification, print the `AgentID` that is returned by the `Secret.details()` method. Then, obtain the `CricketBeacon` agent using this `AgentID` and print out the beacon's name and location.

Running this agent

Compile and make sure there is a catalog running. Same as before but using newbie.tutorial.CricketLearn as the agent (either on the command line or in PowerTester)

Attributes:

host for GoodCricketListener: your ipaq port for GoodCricketListener: 2947 (should be default)

host for Cricketd: your ipaq you will need to login to your ipaq and start a Metaglue platform using your catalog.

Tutorial 3 – Putting It All Together

Goal

We will now use two agents that we saw in the first two tutorials. We will use the `cricket.location.beaconChanged` notification of the `GoodCricketListener` with the `turnOn()` and `turnOff()` methods of the `X10Light` agent. We will write a context-sensitive light switch that will turn on a light when you get close to it.

Basic Information

Each light will have its own cricket beacon next to it. When you receive a `beaconChanged` notification from a cricket, we will obtain the `X10Light` agent corresponding to the new location.

We will do this using the same procedure we did in Tutorial 2: we will get the `X10Light` agent by doing a `reliesOn` call with the `X10Light`'s `AgentID`. To get the `X10Light`'s `AgentID`, we will first find out the occupation and designation of the `X10Light`. We then use this information to construct the `X10Light`'s `AgentID`, which we then pass as an argument to the `reliesOn` call.

The `X10Light` agent's occupation is the string "agentland.device.light.X10Light" (without the quotes) and its designation is the name of the cricket beacon that it is closest to (remember in the previous tutorial, we saw how we can find out the name of a cricket beacon given a `cricket.location.beaconChanged` notification). As an example of what the `X10Light` agent's designation could be, let's say lamp A was next to a beacon named BeaconB. Lamp A's designation would then be BeaconB.

Once we know the `X10Light` agent's occupation and designation, we can construct an `AgentID` using the constructor `AgentID(String occupation, String designation)`. For example, if I wanted the `AgentID` for the lamp with the designation BeaconB, I would write the following line of code:

```
AgentID myID = new AgentID(``agentland.device.light.X10Light'', ``BeaconB'');
```

I could then use `myID` in a `reliesOn` call.

Coding Exercise

Open the file LocationLearnAgent.java.

Write the `tell(Secret s)` method of `LocationLearnAgent`. When you receive a `cricket.location.beaconChanged` notification, you should do the following:

1. Obtain the new beacon's name
2. If you turned on another lamp prior to receiving the `cricket.location.beaconChanged` notification, turn that lamp off.
3. Get the `AgentID` for the lamp that corresponds to the new beacon.
4. Use the `AgentID` for the lamp to get the actual `X10Light` that corresponds to the new beacon.
5. Use the `X10Light` to turn the lamp on.

References

Compiling and Running

There are two commands that you will need:

```
mg_catalog [-purge]
```

```
and
```

```
mg_agent <society> <catalog host> [ <agent name> ] [ <additional names> ]
```

The first command starts the catalog, the central part of the Metaglue system. This is where all agents register so they can be found by other agents in the system. There is an optional

```
-purge
```

argument which will start the system fresh as if there were no agents previously running. Otherwise, the normal starting routine is to attempt to reconnect to the agents the catalog last saw running. This will help if the catalog crashes. Agents will also try to look for the catalog every 10 seconds if the catalog cannot be found.

```
mg_catalog
```

is actually just a shortcut for running

```
mg_agent -catalog
```

.

The second command

```
mg_agent
```

is what actually starts the platform for the metaglue system. This platform will become part of the group named by the society argument. The catalog host is the computer name where the catalog agent is running. Running just this will give a working Metaglue platform which is ready to receive commands for starting agents.

NOTE - Due to the network setup for the class, DO NOT use ĺocalhostás the host name for anything. Use the actual name like t́ablet39ór ípaq301́

We can start agents automatically by adding them in the

```
<agent name>
```

area after the catalog hostname. A good example to know is:

```
mg_agent quig_society tablet39 agentland.debug.PowerTester
```

Remember that the agent name does not end in the word “Agent”! The name must also be the full path name, not just “PowerTester”.

Documentation

Javadocs are available at in the docs directory. These will give you information about the classes being used by `SketchTest.java` and `MyTTT.java` .

```
> mozilla ~/assist/docs/index.html
```

Full Java documentation is at:

```
> mozilla /usr/share/doc/j2sdk1.3-doc/1.3.1/index.html
```

Printing output

Print a line of text

```
System.out.println( "your text here" );
```

Insert an object

You can print data from objects by using string concatenation like in Python:

```
System.out.println( "this object is: " + object );
```

This is an implicit call to `object.toString()` . Methods can be called and what they return will be converted to strings.

```
System.out.println( "the name is: " + object.getName() );
System.out.println( "foo " + object.foo() + " bar " + object.bar() );
```