# RL-Glue Java Codec 2.0 Manual

Brian Tanner

## Contents

# 1 Introduction

This document describes how to use the Java RL-Glue Codec, a software library that provides socket-compatibility with the RL-Glue Reinforcement Learning software library.

For general information and motivation about the RL-Glue[1] project, please refer to the documentation provided with that project.

This codec will allow you to create agents, environments, and experiment programs in Java.

This software project is licensed under the Apache-2.0[2] license. We're not lawyers, but our intention is that this code should be used however it is useful. We'd appreciate to hear what you're using it for, and to get credit if appropriate.

This project has a home here:
`http://rl-glue-ext.googlecode.com/`

## 1.1 Software Requirements

To run agents, environments, and experiments created with this codec, you will need to have RL-Glue (or a surrogate) installed on your computer.

Compiling and running components with this codec requires Java 1.5 or higher. You can find out what version you have by doing the following at the command-line:

[1] `http://glue.rl-community.org/`
[2] `http://www.apache.org/licenses/LICENSE-2.0.html`

```
>$ java -version
```

Recompiling this codec requires the Apache Ant[3] build system. You probably need JUnit and Subversion installed as well. We'll try to make those optional dependencies in the future. The good news is that you probably **don't need to recompile the codec**. Unlike C/C++, we can distribute a java JAR archive of the Java codec, and that is all you really need.

### 1.2 Getting the Codec

The codec can be downloaded either as a tarball or can be checked out of the subversion repository where it is hosted.

The tarball distribution can be found here:
`http://code.google.com/p/rl-glue-ext/downloads/list`

To check the code out of subversion:
`svn checkout http://rl-glue-ext.googlecode.com/svn/trunk/projects/codecs/Java Java-Codec`

Technically all you really **need** is the JAR archive of the codec:
`svn export http://rl-glue-ext.googlecode.com/svn/trunk/projects/codecs/Java/products/JavaRLGlueCodec.jar`

### 1.3 Installing the Codec

There is no real "installation" for the codec per-se. The important thing is to put the `JavaRLGlueCodec.jar` somewhere that is easy to get at. If you want to be fancy, you can probably put `JavaRLGlueCodec.jar` into a directory on your Java system classpath. For the rest of this document, we'll assume you've put `JavaRLGlueCodec.jar` in a subdirectory of your homedirectory called JavaCodec: `~/JavaCodec/JavaRLGlueCodec.jar`.

**POSSIBLE CONTRIBUTION**: IF someone wants to investigate the options for having a script "install" `JavaRLGlueCodec.jar` into a system classpath, we would like to have your help!

## 2 Agents

We have provided a skeleton agent with the codec that is a good starting point for agents that you may write in the future. It implements all the required functions and provides a good example of how to compile a simple agent.

The pertinent file is:

`examples/skeleton/SkeletonAgent.java`

This agent does not learn anything and randomly chooses integer action 0 or 1.

You can compile and run the agent like:

[3] `http://ant.apache.org/`

```
>$ cd examples/skeleton
>$ javac -classpath ~/JavaCodec/JavaRLGlueCodec.jar SkeletonAgent.java
>$ java -classpath ~/JavaCodec/JavaRLGlueCodec.jar:. SkeletonAgent
```

You will see something like:

```
RL-Glue Java Agent Codec Version: 2.0 (Build:192:239M)
     Connecting to 127.0.0.1 on port 4096...
```

This means that the SkeletonAgent is running, and trying to connect to the `rl_glue` executable server on the local machine through port 4096!

You can kill the process by pressing `CTRL-C` on your keyboard.

The Skeleton agent is very simple and well documented, so we won't spend any more time talking about it in these instructions. Please open it up and take a look.

**POSSIBLE CONTRIBUTION**: If you take a look at the agent and you think it's not easy to understand, think it could be better documented, or just that it should do some fancier things, let us know and we'll be happy to do it!

# 3 Environments

We have provided a skeleton environment with the codec that is a good starting point for environments that you may write in the future. It implements all the required functions and provides a good example of how to compile a simple environment. This section will follow the same pattern as the agent version (Section 2). This section will be less detailed because many ideas are similar or identical.

The pertinent file is:

```
examples/skeleton_environment/SkeletonEnvironment.java
```

This environment is episodic, with 21 states, labeled $\{0, 1, \ldots, 19, 20\}$. States $\{0, 20\}$ are terminal and return rewards of $\{-1, +1\}$ respectively. The other states return reward of 0. There are two actions, $\{0, 1\}$. Action 0 decrements the state number, and action 1 increments it. The environment starts in state 10.

You can compile and run the environment like:

```
>$ cd examples/skeleton
>$ javac -classpath ~/JavaCodec/JavaRLGlueCodec.jar SkeletonEnvironment.java
>$ java -classpath ~/JavaCodec/JavaRLGlueCodec.jar:. SkeletonEnvironment
```

You will see something like:

```
RL-Glue Java Environment Codec Version: 2.0 (Build:192:239M)
     Connecting to 127.0.0.1 on port 4096...
```

This means that the SkeletonEnvironment is running, and trying to connect to the `rl_glue` executable server on the local machine through port 4096!

You can kill the process by pressing `CTRL-C` on your keyboard.

The Skeleton environment is very simple and well documented, so we won't spend any more time talking about it in these instructions. Please open it up and take a look.

**POSSIBLE CONTRIBUTION**: If you take a look at the environment and you think it's not easy to understand, think it could be better documented, or just that it should do some fancier things, let us know and we'll be happy to do it!

# 4 Experiments

We have provided a skeleton experiment with the codec that is a good starting point for experiment that you may write in the future. It implements all the required functions and provides a good example of how to compile a simple experiment. This section will follow the same pattern as the agent version (Section 2). This section will be less detailed because many ideas are similar or identical.

The pertinent files are:

`examples/skeleton/SkeletonExperiment.java`

This experiment runs `RL_Episode` a few times, sends some messages to the agent and environment, and then steps through one episode using `RL_step`.

```
>$ cd examples/skeleton
>$ javac -classpath ~/JavaCodec/JavaRLGlueCodec.jar SkeletonExperiment.java
>$ java -classpath ~/JavaCodec/JavaRLGlueCodec.jar:. SkeletonExperiment
```

You will see something like:

```
Experiment starting up!
RL-Glue Java Experiment Codec Version: 2.0 (Build:192:239M)
     Connecting to 127.0.0.1 on port 4096...
```

This means that the SkeletonExperiment is running, and trying to connect to the `rl_glue` executable server on the local machine through port 4096!

You can kill the process by pressing `CTRL-C` on your keyboard.

The Skeleton experiment is very simple and well documented, so we won't spend any more time talking about it in these instructions. Please open it up and take a look.

**POSSIBLE CONTRIBUTION**: If you take a look at the experiment and you think it's not easy to understand, think it could be better documented, or just that it should do some fancier things, let us know and we'll be happy to do it!

# 5 Putting it all together

At this point, we've compiled and run each of the three components, now it's time to run them with the `rl_glue` executable server. The following will work from the examples directory if you have them all built, and RL-Glue installed in the default location:

```
>$ cd examples/skeleton
>$ rl_glue &
>$ java -classpath ~/JavaCodec/JavaRLGlueCodec.jar:. SkeletonAgent &
>$ java -classpath ~/JavaCodec/JavaRLGlueCodec.jar:. SkeletonEnvironment &
>$ java -classpath ~/JavaCodec/JavaRLGlueCodec.jar:. SkeletonExperiment
```

If RL-Glue is not installed in the default location, you'll have to start the `rl_glue` executable server using it's full path (unless it's in your `PATH` environment variable):

```
>$ /path/to/rl-glue/bin/rl_glue &
```

You should see output like the following if it worked:

```
>$ rl_glue &
RL-Glue Version 3.0-beta-1, Build 848:856
      RL-Glue is listening for connections on port=4096

>$ java -classpath ~/JavaCodec/JavaRLGlueCodec.jar:. SkeletonAgent &
RL-Glue Java Agent Codec Version: 2.0 (Build:192:239M)
     Connecting to 127.0.0.1 on port 4096...
     Agent Codec Connected
     RL-Glue :: Agent connected.

>$ java -classpath ~/JavaCodec/JavaRLGlueCodec.jar:. SkeletonEnvironment &
RL-Glue Java Environment Codec Version: 2.0 (Build:192:239M)
     Connecting to 127.0.0.1 on port 4096...
     Environment Codec Connected
     RL-Glue :: Environment connected.

>$ java -classpath ~/JavaCodec/JavaRLGlueCodec.jar:. SkeletonExperiment
Experiment starting up!

RL-Glue Java Experiment Codec Version: 2.0 (Build:192:239M)
     Connecting to 127.0.0.1 on port 4096...
     Experiment Codec Connected
     RL-Glue :: Experiment connected.

RL_init called, the environment sent task spec: 2:e:1_[i]_[0,20]:1_[i]_[0,1]:[-1,1]


----------Sending some sample messages----------
Agent responded to "what is your name?" with: my name is skeleton_agent, Java edition!
Agent responded to "who is your daddy and what does he do?" with: I don't know how to respond to your message

Environment responded to "what is your name?" with: my name is skeleton_environment, Java edition!
Environment responded to "who is your daddy and what does he do?" with: I don't know how to respond to your message


----------Running a few episodes----------
Episode 0  10 steps  -1.0 total reward  1 natural end
Episode 1  10 steps  -1.0 total reward  1 natural end
Episode 2  10 steps  -1.0 total reward  1 natural end
Episode 3  10 steps  -1.0 total reward  1 natural end
Episode 4  10 steps  -1.0 total reward  1 natural end
```

```
Episode 5  1 steps  0.0 total reward  0 natural end
Episode 6  10 steps  -1.0 total reward  1 natural end


----------Stepping through an episode----------
First observation and action were: 10 and: 0


----------Summary----------
It ran for 10 steps, total reward was: -1.0
```

# 6 Who creates and frees memory?

The RL-Glue technical manual has a section called *Who creates and frees memory?*. The general approach recommended there is to make a copy of data you want to keep beyond the method it was given to you. The same rules of thumb from that manual should be followed when using the Java codec.

# 7 Advanced Features

This section will explain how to set custom target IP addresses (to connect over the network) and custom ports (to run multiple experiments on one machine or to avoid firewall issues).

Someone should write this later (Opportunity to contribute!).

# 8 Codec Specification Reference

This section will explain how the RL-Glue types and functions are defined for this codec. This isn't meant to be the most exciting section of this document, but it will be handy.

Instead of re-creating information that is readily available in the JavaDocs, we will give pointers were appropriate.

## 8.1 Types

### 8.1.1 Simple Types

Unlike the C/C++ codec, we will not be using `typedef` statements to create special labels for the types. In Java:

- *reward* is `double`
- *terminal* is `int` (1 for terminal, 0 for non-terminal) We hope to replace these with boolean eventually.
- *messages* are `String`
- *task specifications* are `String`

### 8.1.2 Structure Types

All of the major structure types (observations, actions, random seed keys, and state keys) extend the `RL_Abstract_Type` class.

All of the types are listed in the `org.rlcommunity.rlglue.codec.types` package.

## 8.2 Functions

### 8.2.1 Agent Functions

All agents **should implement** `org.rlcommunity.rlglue.codec.AgentInterface`.

### 8.2.2 Environment Functions

All environments **should implement** `org.rlcommunity.rlglue.codec.EnvironmentInterface`.

### 8.2.3 Experiments Functions

All experiments **can call** the static methods in `org.rlcommunity.rlglue.codec.RLGlue`.

# 9 Changes and 2.x Backward Compatibility

There were many API/Interface changes from RL-Glue 2.x to RL-Glue 3.x. For those that are at the level of the API and project organization, please refer to the the RL-Glue project documentation.

## 9.1 RL-Glue Split and Package/Jar Naming Changes

With the RL-Glue 3.x project, we've decided to keep the codecs separate from the main RL-Glue project. This is so the codecs can be more agile while the main RL-Glue project stays very stable.

Since things were changing, we took an opportunity to fix some of our long-term gripes with the Java codec. The name of the JAR file has changed from `RL-Glue.jar` to `JavaRLGlueCodec.jar`.

The JAR file is now distributed in:
`rl-glue-ext/projects/codecs/java/products/JavaRLGlueCodec.jar`

In the previous incarnation of the Java codec, the classes were in a very shallow heirarchy: `rlglue.RLGlue`, `RLGlue.types`, etc. For the updated release, we've moved to a richer package description that is more in line with other Java projects in the Reinforcement Learning community. The new package hierarchy is:
`org.rlcommunity.rlglue`

We've also moved a few of the class/interfaces around. The most notable change is that instead of `Agent` and `Environment` interfaces, we now have `AgentInterface` and `EnvironmentInterface`. Also, instead of these interfaces being in their own package, they are now in: `org.rlcommunity.rlglue.codec`.

Updating existing code might seem like a lot of work, but it's easier than it seems. The main changes from the user end are: 1) Using the right jar (JavaRLGlueCodec.jar instead of RL-Glue.jar) (Not necessary if you are using RL-Viz)

2) Change classes that implement `Agent` and `Environment` to `AgentInterface` and `EnvironmentInterface`

3) Change package imports. Find and replace:

```
rlglue.RLGlue                    ==>    org.rlcommunity.rlglue.codec.RLGlue
rlglue.types.                    ==>    org.rlcommunity.rlglue.codec.types.
rlglue.agent.Agent               ==>    org.rlcommunity.rlglue.codec.AgentInterface
rlglue.environment.Environment   ==>    org.rlcommunity.rlglue.codec.EnvironmentInterface
```

Those three things should cover most of it. If someone makes a very strong case, we can probably create a codec that is completely compatible with the old naming conventions. However, I assure you that I uploaded several tens of thousands of lines of code for the RL-Viz, RL-Library, and BT-AgentLib projects in only about 30 minutes total.

**POSSIBLE CONTRIBUTION**: If we've missed anything, or there is an easier way, please let us know!

## 9.2 Agent/Environment Loading

Historically, there have been a few different approaches for loading agents and environments.

For all these examples, lets pretend that the `JavaRLGlue.jar` path and JAR are in an environment variable called CODECJAR. We could actually do that (with a bash shell) like:

```
export CODECJAR=/path/to/JavaRLGlueCodec.jar
```

The first (original) method was to call the main() method of the AgentLoader/EnvironmentLoader class, passing it the name of the Agent/Env class you wanted to load. Hopefully you remembered to put it in your classpath. This approach inevitably lead to lots of typing to load an agent, as frustrating run-time failures when paths weren't set right. Here is an example of how that looks:

```
java -classpath $CODECJAR:/to/Agent/classes org.rlcommunity.rlglue.codec.util.AgentLoader myAgent
```

The second method was to bypass the AgentLoader and go through `RLGlueCore`, which is contains the default `main` method in the JAR. This method is a bit better, but it is still long and depends on run-time checking in a way that we (I?) really didn't like. Here is how it looks:

```
java -classpath /to/Agent/classes -jar $CODECJAR myAgent
```

The **new and improved** method is to put a `main` method inside your agent/environment that calls the applicable loader so that the agent/env can load itself. So, in your agent (for example) you'd have code like:

```
import org.rlcommunity.rlglue.codec.util.AgentLoader;
/* rest of agent here */
//This would be inside MyAgent.java
public static void main(String[] args){
        AgentLoader theLoader=new AgentLoader(new MyAgent());
        theLoader.run();
}
```

So, for the price of that tiny bit of code inside your agent, you can now run the agent class directly:

```
java -classpath $CODECJAR:/to/Agent/classes myAgent
```

We feel that this is a useful step forward, and will be encouraging this approach.

# 10 Frequently Asked Questions

We're waiting to hear your questions!

# 11 Credits and Acknowledgements

Andrew Butcher originally wrote the RL-Glue network library and first version of this codec. Thanks Andrew.

Brian Tanner has since grabbed the torch and has continued to develop the codec.

## 11.1 Contributing

If you would like to become a member of this project and contribute updates/changes to the code, please send a message to rl-glue@googlegroups.com.

# Document Information

```
Revision Number: $Rev: 239 $
Last Updated By: $Author: brian@tannerpages.com $
Last Updated   : $Date: 2008-09-28 22:55:51 -0600 (Sun, 28 Sep 2008) $
$URL: https://rl-glue-ext.googlecode.com/svn/trunk/projects/codecs/Java/docs/JavaCodec.tex $
```