

University POLITEHNICA of Bucharest, Romania
Faculty of Automatic Control and Computer Science



WackyRacers

Server Manual

Document version: 1.1

Team UPB (board):

Cristian Andreica
Project manager
Alexandru Isaila
Team Leader
Bogdan Ivascu
Team Leader
Ciprian Mardare
Team Leader

Mentors:

Adrian Cătu
Coach & Advisor



Version 1.1 Release notes:

- [fixed] server crashed when connecting an extra AI after the race has started
- [fixed] each has starts with a null command (0,0,start_angle), even if the AI sends a command before the first 0.1 seconds elapsed
- [fixed] <https://koala.cs.pub.ro/redmine/issues/754>
- [fixed] <https://koala.cs.pub.ro/redmine/issues/756>

Document changes:

- [updated] *Send Driving Info* message, changed ACC and BRK values from *int* to *double*
- [added] Chapter 2.3 – Server Command Queue
- [added] Chapter 2.4 – Car Disqualify Policy



1. Overview

The purpose of this document is to present the Server module for the Formula 1 simulation game. It will be used as a manual for installing and deploying the Server, connecting with the Viewer, developing the AIs and finally, connecting with the AIs.

This document is intended for both the evaluators of this project as for the development teams and will be presented on 18.12.2010 for approval.

1.1 Server Characteristics

Event Based Implementation

- request-response behavior

Highly Scalable

- multithreaded implementation
- the Viewer and AIs can be connected in parallel, no constraints

Network Configurable

- binds to hostname and port set by user

Multiple Races allowed

- no need to restart
- restart policy on errors

AI controller

- user controlled car for testing server

Console Input

- user can ask the server for information
- ability to force race start if some AIs fail

Message Queue drop policy

- the AI can specify if he wants to drop the message queue

Map compression

- the map is compressed for speeding communication with AIs

High-Precision distance

- line-cross algorithm for calculating distance traveled on track

Flexible AI protocol

- messages have headers for maintaining compatibility with legacy versions



Debug-Mode

- if wanted, all the messages exchanged can be viewed on console

Wrong Way notification

- the AI is informed if he's going the right way

1.2 Step-by-step install & run

The server module the next files:

/server.jar – server binaries

/server.bat – server executable

/aiController.jar – aiController binaries

/high_speed_car.bat – aiController executable

/default_car.bat – aiController executable

/best_handling_car.bat – aiController executable

In order to configure the server, edit the *server.bat* file. The parameters are:

- *hostname* to bind to (egs. *localhost*)
- *port* to listen for viewer connections (egs. *4444*)
- *port* to listen for AI connections (egs. *4445*)
- *debug* to show communication on console (egs. *false*)

Double-click on *server.bat* to start the server.

AI controller can be used for playing a car from keyboard, by using the arrows. Before starting the controller, you should configure it properly.

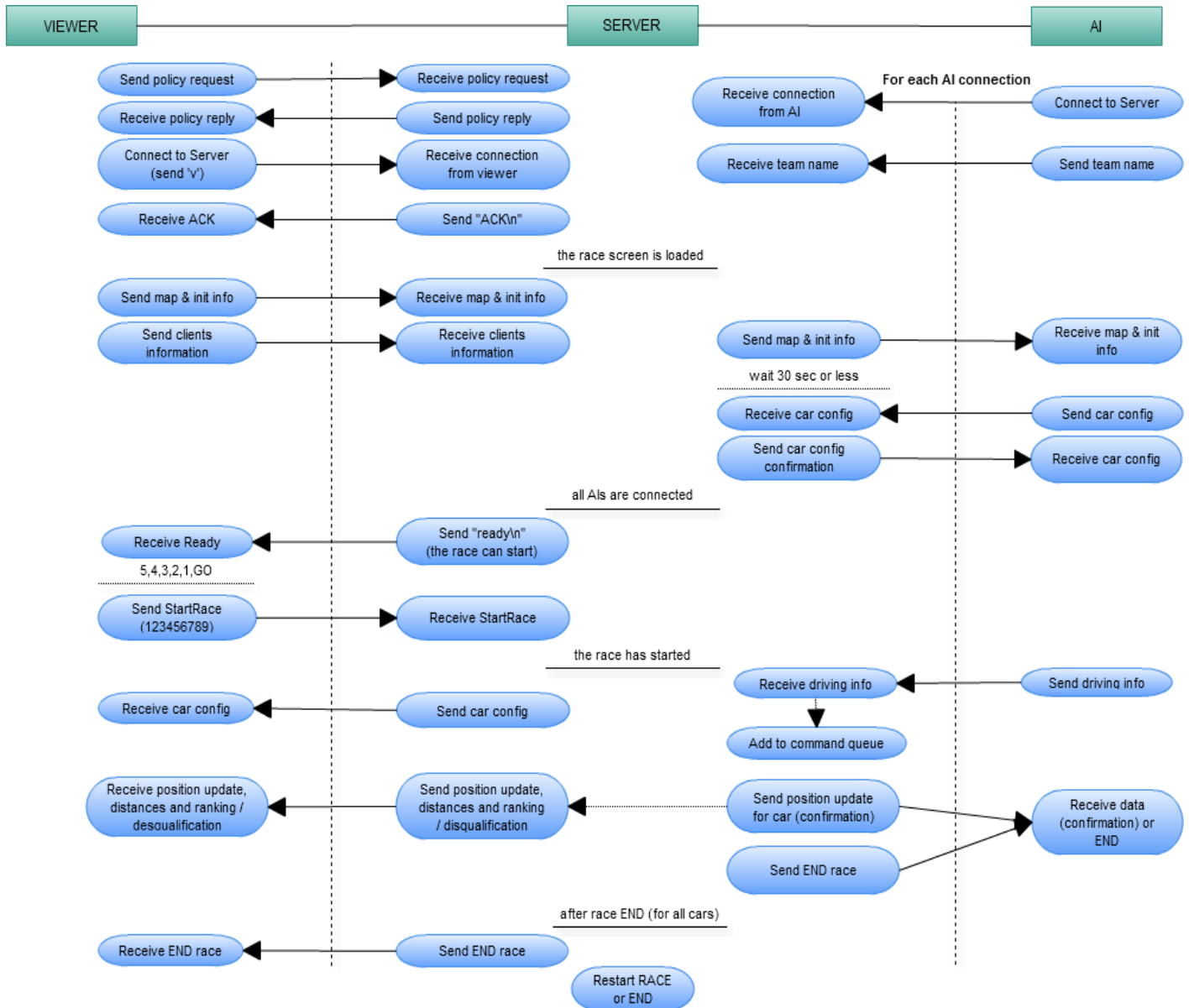
The parameters for AI controller are:

- *hostname* to connect to (egs. *localhost*)
- *port* to connect to server (egs. *4444*)
- *team_name* for the current race, as stated by the Viewer (egs. *WackyRacers*)



2. Development Manual

2.1 AI – Protocol Diagram





2.2 Connect AI with Server – Step By Step

All data should be sent/received in big endian format. Strings should be sent/received byte with byte, in UTF-8 encoding. One integer has 4 bytes. One double has 8 bytes.

Each message from the AI-Server network protocol has a 1 integer header. This header represents the message type and the Server expects different data, based upon this value.

The messages should be sent / received exactly in the order displayed on the diagram.

Headers

- [1] Team name
- [2] Map dimensions
- [3] Initial info
- [4] Car configuration (and confirmation)
- [5] Send driving info
- [6] Receive position confirmation
- [7] End Race

Send team name

- header (1 integer)
- name length (1 integer)
- case-sensitive name ($N * 1$ byte)

The AI needs to correctly send the team name in order to connect to a race, because the Server makes the associations AI – car based upon the team name received from Viewer.

Receive map

- header (1 integer)
- width in pixels (1 integer)
- height in pixels (1 integer)
- map (width*height + padding integers)

The encoding for the map: 32 values inside one integer. If the map is not 32 multiple, then 0 padding is used.



Receive initial info

- header (1 integer)
- start point X (1 integer)
- start point Y (1 integer)
- width in meters (1 integer)
- height in meters (1 integer)
- direction (1 integer)
- number of laps (1 integer)
- maximum lap time (1 integer)
- initial car angle (1 double)

The direction is 1 for clockwise, 0 for counter-clockwise. The maximum lap time is represented in milliseconds. The initial car angle is a value in radians in $[0:2\pi]$.

Send car configuration

- header (1 integer)
- max acceleration in m/s^2 (1 integer)
- max brake in m/s^2 (1 integer)
- max top speed in m/s (1 integer)
- steering speed in rad/s (1 double)

The initial steering speed is a value in radians in $[0:2\pi]$. These values should respect the formula $10\text{ACC} + 10\text{BRK} + \text{SPD} + 200\text{STR} \leq 400$. This message has a confirmation. If the formula is not respected, the Server responds with the default values ($\text{ACC} = 10$, $\text{BRK} = 10$, $\text{SPD} = 100$, $\text{STR} = 0.5$).

Receive car configuration confirmation

- header (1 integer)
- max acceleration in m/s^2 (1 integer)
- max brake in m/s^2 (1 integer)
- max top speed in m/s (1 integer)
- steering speed in rad/s (1 double)

The header has the same value as the previous message, which is 4.

Send driving info

- header (1 integer)
- acceleration percentage (1 **double**)
- brake percentage (1 **double**)
- wanted angle (1 double)
- drop queue (1 integer)



The value for drop queue should be 1, if and only if the AI wants to drop all the messages he has sent but haven't been applied yet. Else, this value should be 0.

Receive position confirmation

- header (1 integer)
- current X in meters (1 double)
- current Y in meters (1 double)
- current speed in m/s (1 double)
- current angle in radians (1 double)
- current direction (1 integer)

The current direction is 1, if the AI is driving correctly in the race. If he's driving reverse, -1 is sent.

Receive end race

- header (1 integer)

2.3 Server Command Queue

The Server maintains a command queue for each car. This queue is where all the car commands received from AI end up before actually being processed.

At each 0.1 seconds, the Server extracts a command for each car. If no command is available at the next extraction, the last command is reapplied for the same car. If, by chance, the AI doesn't send the first command, then a *null* command is applied (ACC=0%, BRK=0%, STR=initial_angle), until a command is received.

There are cases when the AI receives a command confirmation which he does not expect, most probably because bad calculus. The AI can drop the queue, telling the server which is the next command to be applied. Use the *drop queue* parameter in *Send Driving Info* message to drop the queue, before adding the current message.

2.4 Car Disqualify Policy

The cars can get disqualified for the race if their current lap time exceeds the maximum lap time for the race. This is the normal behavior.

However, there are other cases when a car can get disqualified. If an error appears when sending/receiving messages with the AI, that AI(car) gets automatically disqualified for that race. This policy is the safest to apply, as it does not affect in any way the other players/cars.



If the AI sends all the commands for that race, as he could have calculated them before receiving confirmations from Server, and the application exits, the Server receives an exception on his socket, thus being forced to disqualify that car.

As a general rule: let the AI read messages from Server until the *End Race* message.