Train Simulator External Interface API

Introduction

The External Interface API (EI hereafter) enables external applications and hardware to interact with Train Simulator during runtime, both to find out what is happening and also to make changes.

The primary usage of this API is to enable external hardware vendors to be able to create hardware with gauges, indicator lights and panels as well as levers, buttons and switches to provide input.

It is also important to note that the full functionality that was provided originally by the interface is still present and can be used.

Access

For legacy compatibility reasons, the filename of the interface is “raildriver.dll”. Import this in to your application code and you will then be able to access a range of functions that will use the DLL file to communicate with the game at runtime.

Applications

Lights

Trains have a wide range of lights such as PZB and LZB indicators, Wheel Slip and Sanding indicators and a number of forms of cab signalling.

In addition, there is a capability to determine if the player train is inside a tunnel, this could be used to switch off the lights in the room when the player plunges in to a dark tunnel, adding greatly to the immersion. Further, with access to the current time of day, applications can dim or brighten room lights automatically for the time of day of the scenario.

Gauges

Most trains have a variety of fairly standard gauges such as speedometer, various types of brake-related gauges, fuel levels, ammeters, rpm counters and numerous gauges related to steam locomotives such as boiler pressure and steam chest pressure.

Panels

Part of the EI allows external hardware to access the current rotation and pitch of the player locomotive as well as its longitude and latitude in the world. This information could be used to plot a path via an external system such as Google Earth.

Levers

With regulators, reversers, brakes and numerous other levers, there are a number of opportunities for control of these levers with external hardware.

Buttons

There are some common buttons in many locomotives, particularly regional, such as the DSD foot pedal, AWS Acknowledge button, SIFA Acknowledge button and the three PZB buttons. These are all great candidates for being exposed via external hardware and allowing a much more responsive feel.

Switches

Lights and wipers are two common switches but there are many more.

API Function Reference

```
string GetLocoName()
```

This function returns the Provider, Product and Engine name of the engine being driven at the moment. This is in the form “PROVIDER.:.PRODUCT.:.ENGINENAME”. This allows application developers to accurately determine what locomotive is being driven and adapt accordingly.

```
string GetControllerList()
```

This function returns a list of every controller in the current locomotive. Each one is separated by two colons (::). This list is later used for indexing so the first entry in this list should be considered controller 0 and the next controller 1 etc. This is something like “Alerter::VirtualThrottle::Regulator::TrainBrake” etc. This would usually be the same list as seen in the Control State Dialog.

```
float GetControllerValue(int controllerId, int getType)
```

This function queries the value of a controller and returns it.

getType should be 0 to get the current value, 1 to get the minimum value of this controller and 2 to get the maximum value of this controller.

It is important to use the controllerId within the context of the values returned by GetControllerList as it cannot be assumed that a particular controller is always at the same ID on different loco’s, in fact it quite often won’t be.

```
void SetControllerValue(int controllerId, float value)
```

Set the new current value of the controller specified to the value provided.

In addition to the controllers that are returned via GetControllerList() there are some additional “virtual” controllers that are provided by the EI itself, these are:

Controller ID	**Purpose**
400	Latitude of Train
401	Longitude of Train
402	Fuel Level
403	Is in a Tunnel?
404	Gradient
405	Heading
406	Time of day hours
407	Time of day minutes
408	Time of day seconds

Implementation Guide

It is expected that applications will frequently call GetLocoName to determine if the locomotive has changed. If it does change, then the application should call GetControllerList and then determine if it needs to adjust how it's interacting with the current locomotive.

If the application is trying to be more generic in how it interacts with different types of locomotives, it may be important to call the GetControllerValue function on initialisation to request the Min and Max of each controller that the application might want to set – this would then allow the application to scale its values from 0 to 100% accordingly, rather than assuming that all controls are perhaps 0.0 to 1.0 (which maybe the case generally but will leave edge cases not working).

Operationally, the application would then use the GetControllerValue to retrieve values and SetControllerValue to set them. Note that note all controllers can have their values set – the same rules as for LUA Engine scripts apply (e.g. you can't set the boiler pressure). All controls can be read from.