TDE
DIGITAL CREATIVES IN SPORTS

Development API

Improving the Grand Prix experience for F1 viewers at home

S8 Graduation FHICT

4 Sept 2023 - 16 Jan 2024

By Jordi Franssen

31 October 2023

www.tde.nl

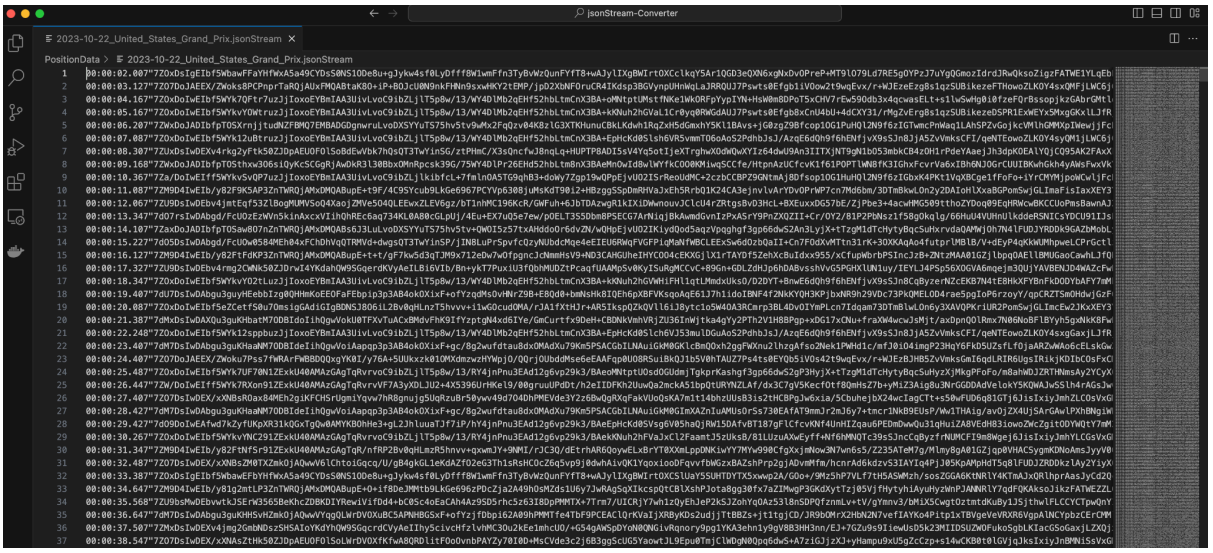
Introduction

In the available data analysis, I have already made a simple script that records the endpoint of an api. Later, I elaborated upon this to work with the websockets from the FIA API. Now, I've done a few attempts to collect data for development during races, but failed as websockets are still quite complex. Now I discovered that it's also possible to download entire datastreams of a Formula 1 race. Therefore, it's no longer necessary to collect data from a live API. However, my tool needs to work with live data. Therefore, I built a development API that utilizes these downloadable datastreams and converts them into live data so it resembles the live data API that will be used by the finished product.

In this document, you'll read about my approach and how I made the tool to set up a development API.

Livetiming.formula1.com/static

This is the URL a colleague shared that houses static F1 data. This URL also contains endpoints to dynamic, live data during racing events. This is an example. Entire datastreams can be downloaded by replacing .json with .jsonStream in the URL. Below is a screenshot of what this data looks like.



These files are massive, and it's not practical to send these to the client. This data is also encrypted. Therefore, the tool needs to decrypt this data and break it up into chunks. Fortunately, each line has a timestamp of when this data was present in the live-timing API. However, these timestamps are quite random and don't resemble a consistent interval. These timestamps are accurate to the millisecond, so it was quite challenging to accurately send this data to the API. I solved this by logging each timestamp and having the API use the correct chunk of data for each timestamp.

info@tde.nl
(+31) 040 782 00 01
www.tde.nl

S8 Graduation FHICT
s8-graduation.jordifranssen.com
Jordi Franssen

Rewriting the REST-API to SSE

The development API is now written so clients can poll data using a REST-API. I built it this way as I thought it would be a good method to use for my MPV and because it was relatively easy to make. However, in the meantime, I conducted research and discovered that SSE is the best method to get the racing car positions from the server to the client. As I've never heard of this method before I decided to rewrite the REST-API part of my development api to Server Side Events (SSE).

I also tried to make the API accurate on the millisecond. Therefore, each millisecond a check was done to confirm if data was available on that moment. This task turned out to be too heavy for the CPU, so I rounded the timestamps to 200ms, similar to the update frequency of the F1 API. Therefore a check for available data is only done five times per second.

Future recommendations

The development API can only make position data available from every race found in the static API. The rest of the data is only available from the 2023 Vegas GP as this is the race that was recorded live. It is possible to download all data from the static API and reconstruct a server that exactly mimics the live API, but this takes a lot of time. And since I only need the development API for development and testing, the 2023 Vegas GP will do the job just fine, but to test and display every race in the static API, the development API will need to be elaborated upon.

Result and conclusion

I ended up with a tool that only needs the URL from the datastream to start in the CLI. It downloads the jsonStream file, decrypts it and breaks it up in chunks, then saves these chunks in a folder. When this is all finished, it starts an API that presents the chunks accurately on the millisecond as I logged the timestamp for each data chunk.

The repository can be found here:
https://github.com/Lampenkap007/datastreamConverter

Summary

The tool I'm building is using live data from a Formula 1 race. These races only happen on the weekend and the last race of this season is in the last weekend of November already. For the sake of convenience, it's important to be able to develop and test the tool whenever I want. Therefore, I built a development API. This API is able to process the positioning data from the Formula 1 static API from every race and all the raw data from the live API from the 2023 Vegas GP. The data from this race will be sufficient to build the tool, but to test the tool for other races, the API will have to be elaborated upon in order to process all the data from other races.

Learning Outcome Clarification

- Learning Outcome 1: Professional Duties
- Learning Outcome 2: Situation-Orientation
- Learning Outcome 4: Investigative Problem Solving

This deliverable is a professional duty on a bachelor level in the activities of Realize and Advise as I realized a development API to use during development and advised on future elaborations of the API. Therefore, Learning Outcome 1: Professional Duties applies

This deliverable is relevant for one or more persons and creates value as this API is necessary to test and build the tool I'm going to make. Therefore, Learning Outcome 2: Situation-Orientation applies.

I identified a problem, which is the inability to develop and test when live data is not available. I found an effective approach and arrived at appropriate solutions by making a development API so I can test and build whenever I want. Therefore, Learning Outcome 4: Investigative Problem Solving applies.