# Vehicle Plot Delivery Task

There are several components involved in delivering plots into the system:

* Sender
* Receiver
* Query Service.

**Sender** is a console application that generates random plots for a given vehicle. The messages being sent contains the following information (**VehicleId**: int, **Latitude**: double, **Longitude**:double, **TimeStamp**:datetime, **EventCode**: IgnitionOn|IgnitionOff|Movement). Plots are sent to the receiver via queue. When running sender you can configure a vehicleId and intervals in which plots are sent (e.g. running “Sender.exe -v 10 -interval 500” will start sender which sends random plot each 500ms for vehicle with id = 10). Make sure the journeys created are correct (e.g. Movement is not sent after IgnitionOff, or IgnitionOn/Off is not sent twice). Any number of concurrently running senders is supported.

**Receiver** is a process that picks up plots from the queue and puts them in the distributed cache for the query service to consume. The caching window is configured in Receiver to xx seconds. Any number of concurrently running receivers is supported.

**Query Service** is a service that queries a history of plots for a given vehicle within a given timeframe. Additionally, for each plot when vehicle is moving return a time where vehicle was started (start of the journey). When vehicle is not moving return a time where vehicle stopped (end of the journey). Assume all queries are sent within a caching window so the data can be retrieved directly from the cache.

**Example:**

Plots sent by Sender over time:

{"VId": "v1","Lat": "la0","Lon": "lo0","EventCode": "IgnitionOff", "TimeStamp": "t0" },

{"VId": "v1","Lat": "la1","Lon": "lo1","EventCode": "IgnitionOn", "TimeStamp": "t1" },

{"VId": "v1","Lat": "la2","Lon": "lo2","EventCode": "Movement", "TimeStamp": "t2" },

{"VId": "v1","Lat": "la3","Lon": "lo3","EventCode": "Movement","TimeStamp": "t3" },

{"VId": "v1","Lat": "la4","Lon": "lo4","EventCode": "IgnitionOff","TimeStamp": "t4" },

{"VId": "v1","Lat": "la5","Lon": "lo5","EventCode": "IgnitionOn","TimeStamp": "t5" }

Plots queried for this vehicle for timeframe between t1 and t5

Query Service returns:

[

{"VId": 1,"Lat": "la1","Lon": "lo1","EventCode": "IgnitionOn", "TimeStamp": "t1", "JourneyStart": "t1", "JourneyEnd": "null" },

{"VId": 1,"Lat": "la2","Lon": "lo2","EventCode": "Movement","TimeStamp": "t2","JourneyStart": "t1","JourneyEnd": "null"},

{"VId": 1,"Lat": "la3","Lon": "lo3","EventCode": "Movement","TimeStamp": "t3","JourneyStart": "t1","JourneyEnd": "null"},

{"VId": 1,"Lat": "la4","Lon": "lo4","EventCode": "IgnitionOff","TimeStamp": "t4","JourneyStart": "t1","JourneyEnd": "t4"},

{"VId": 1,"Lat": "la5","Lon": "lo4","EventCode": "IgnitionOn","TimeStamp": "t5","JourneyStart": "t5","JourneyEnd": "null"}

]

## Task

Implement plot delivery subsystem focusing on performance (maximize as much as possible throughput and minimize latency of each component). Measure and report the following metrics under load test (the test fixture could be a part of solution):

# of messages/sec Receiver can handle

# of queries/sec Query Service can handle

avg latency of query service request processing

The implementation decisions have to be taken based on the Goals specified below

## Goals (ordered by priority)

1. Performance. Target for highest throughput and lowest latency possible.
2. Code Quality. Focus on clean and SOLID implementation. Units tests is an advantage.
3. Maintainability. It should be easy and straightforward to configure & run the application with as little configuration as possible.

## Implementation Requirements

**Platform**: .NET 4.6.2 or .NET Core

**Language**: C#

**Cache**: Redis (or anything else you think is better, justification is required)

**Message Broker**: RabbitMQ (or anything else you think is better)

It is recommended to implement **Sender** as console application, **Receiver** as windows service and **Query Service** as a WCF or ASP.NET MVC Core endpoint hosted inside a windows service.

For any questions contact: vladimir.makayev@verizonconnect.com