Big Picture & Design concerns

Some initial collection of topics to cover, concerns and challenges, and how to structure the definition of the "big picture" *Vehicle Data reference architecture.*

Big Picture

> **WARNING**
>
> Big Picture diagram below is out of date. See CCS PoC section for up to date version

look at this diagram

- VSS2 is used for the description of the data content
- it is proposed to re-use the container concept proposed by CVIM to encapsulate the data (the sample code in the box comes from CVIM)
- both MQTT (currently used by various OEMs) and new options (like GraphQL, look at GENIVI AASIG work on the external data server) are used for the communication between the in-vehicle part and the OEM cloud
- both REST and WebSocket protocols are used for the communication between the OEM cloud, neutral server and 3rd party service

⭐ From this outline below we have started defining example reference architectures and potential technologies.
Several of the later items however can be further analyzed.

0) Initial assumed technical architecture ("block architecture")

What: Draw and list technical components that may be involved in the architecture. This includes both hardware and software components.

We can start with some of our inputs we may use their proposed assumptions (ExVE and other) as initial input. However:

⚠️ The order of definition is not fixed here and it is a challenge. The technical architecture is basically the intended end result but at the same time a rough architecture needs to be drawn to identify the parts we intend to talk about (data flows, protocols, actors, and so on).
Therefore, we likely need to iterate over all of these angles of attack.

1) Technologies used in each system part

What: List typical hardware and software components. This may be quite abstract at times (e.g. the exact hardware of course differs) but in that case list the assumed *type* of component.

--> HW / Operating System / Software stack
(likely not very strictly defined -- lots of variability and continuous change)

Example of system parts
- Vehicle ECUs
- Vehicle gateway ECUs
- OEM intermediate servers
- Neutral server
- "Developer" endpoints (Web services and/or end-user devices?)
- Final app. What are the developer final products, i.e. the outcome of 3rd party development? This could be for example web services, or apps on popular smartphones.

2) Data flow architecture

What: A block diagram containing interacting technical systems (and consequently interacting actors) focusing on the data flow links between them.

This is to identify primarily the links between technical systems that need a defined communication protocol, (and preceeding that of course quality requirements and similar input).

3) Data protocols

What: Based on the block diagram, and performance/quality/content requirements for each link, propose appropriate data protocol standards.

The choice may be affected also by the communicating software components

Vehicle buses ECUs -> "Vehicle Gateway" (example EE architecture)
brief and for completness

Vehicle -> OEM server
(Vehicle -> 3rd party servers)
- ?
OEM server <-> Neutral Server
Neutral Server <-> 3rd party developers

For each actor in data flow architecture:
- Identify needs/requirements from that actor

For each link in data flow architecture
- (Specify actors involved)
- Identify data set (limitations, + non-functional requirements)
- Identify data protocol

4) Data needs

What: Define the different categories of 3rd party use-cases and the different technical needs for consuming the data.

- **Personalised vehicle data**
  Personalised data is identifiable to a specific VIN and is needed for service offerings to individuals and fleets. It should be discussed how much of these use-cases should be supported by a technical solution:
  - Retrieval of the latest cached vehicle data from the OEM server on a request basis.
  - Allow 3rd party event subscriptions and send-out of notifications when new data is available.
  - Allow the retrieval of historical vehicle states, e.g. the data values of the last 12 hours.
  - Provide a socket type of streaming API with real-time data updates.
  - Allow the triggering of a vehicle status refresh by the 3rd party of an individual vehicle.
  - Allow data retrieval of a group of vehicles by a fleet owner instead of having to request data for each individual VIN.
- **Anonymised vehicle data**
  Big Data that is distributed to 3rd parties at different update rates. Has to be enriched and analysed in order to provide insights e.g. for traffic information.

Initial ideas

Protocol for Vehicle -> OEM server
1) (W3C Gen 2)
2) Other? MQTT, etc...
3) Are those proprietary protocols (i.e. *unspecified* in this project)?

Protocol for Vehicle -> 3rd party servers
- Confirm: Is this desired and accepted?
- This seems one primary aim for W3C Gen2 to cover

OEM server <-> Neutral Server
1) W3C Gen 2
2) Big-data flow frameworks à la Apache NiFi

Neutral Server <-> developers

I think there are missing standards remaining to do here?
... W3C Gen 2 is likely not enough (pending requirements)... I'm thinking of:
- How to add address individual vehicles, or subsets of a fleet.
- How to gather aggregate data, etc.

W3C protocol is kind of point-to-point (developer app requests to one specific service, in-vehicle or

- SensorIS specifications are covering some of this, as well as the definition of how to *request* measurements to be done on the car.

W3C Gen 2 covers how to fetch a particular value "on demand", which sort of presumes it will deliver the (single) latest known/cached value.
SensorIS covers the definition of a "data-gathering" request to be sent, then processed by car (for possibly some extended time), then reported back.