*FIRST DRAFT, DO NOT CIRCULATE...*

**Cloud & Connected Services  
“Gap analysis document” – VSS chapters only**

**Motivation & problem space**

The Vehicle Signal Specification proposal has been active since at least early 2016, where it outlined he intention to produce standard data descriptions among different car brands.

Its original focused on the fact that the data signals on CAN tended to be completely proprietary and different among every car vendor.

This may have led to the misunderstanding that it would stand or fall on the condition of car OEMs being able to use the exact same CAN network standards but this is not the case.

- As we will see later, a standard like VSS is not only for the low-level network but also for other data exchange across cars, and cloud-based application.

- APIs and agreements are needed on other levels of systems than the low-level CAN bus

- Translations/mappings are possible from one level to another - thus if CAN networks are not speaking VSS signals directly, they can still be mapped into those.

- In-car networks are changing in parallell, which in any case may replace CAN with Ethernet/TCP-IP style communication, or other types of vehicle-buses that \*anyhow\* require a different encoding of data in signals since they are not based on the specific size limit imposed by CAN frames. In such cases, having a specification of standard data can help.

The proprietary nature of CAN buses is still a reality today, and a challenge to change since it incurs significant cost and involves large parts of the OEM electrical engineering departments that rely on the signal standards.

Of course, more standardization may over time reduce development cost and concerns, but in the shorter time it might incur an expense.

Unfortunately the (perceived) focus on CAN signals has led to some fairly vague concerns hampering the uptake of VSS in the beginning, which is thankfully changing now.

It is possible that additional non-technical (business strategy) concerns were hampering the project because there was unclarity to what extent companies would release control over what they have (or take on a major effort to adjust what they have).

Concerns among OEMs included the current situation of CAN buses being generally accessible (through ODB2-port or similar) and that the idea that the proprietary nature of signals was preventing unauthorized access to OEM-only features. Some such features might have a minor to medium impact (boosting engine power) or even more serious (unlocking cars and start-prevention systems with the intention of theft). Keeping data definitions secret has of course been very ineffective as a prevention, since the knowledge of many OEM-proprietary signals exist among proponents of both legal and illegal activities and tooling, and the solutions lie in proper security architectures. Nonetheless, the main point here is the understanding that VSS outlines a standard for data that OEMs can agree upon, as well as a methodology and tooling that are equally applicable on extension data trees that can be proprietary.

\* VSS can standardize harmless signal data without requiring all car signals to be standardized.  
\* The ability to write data / affect functions is separate from the ability to read information and controlled by access-control mechanisms that can apply in the implementation of the signal database.

It was also clear early on that regardless of the possibility to make CAN and similar networks adhere to a standard, standardizing the data descriptions using VSS or similar would still open up for programming standards on other levels, both within the car (application API) and outside (big data exchange, cloud/web applications or other)

It is essentially /required/ to have some kind of data API if data is to be used by applications. Therefore, this would be useful, perhaps even necessary, to define even if there is a translation required from the CAN level to this API level. If the desire is there to create a 3rd party application development ecosystem, then those developers would likely require some standardization of those APIs so that they are not different everywhere. This makes a standard like VSS useful as a definition of a

shared API, regardless if the low-level data is fully equal across adopters.

The usage of VSS as an underlying signal description database in the protocol work by W3C Automotive Working group have also proven in practice that the idea behind the hierarchical data description of VSS goes beyond the low-level signalling buses like CAN.

**Licensing**

The licensing of VSS is unique in that it started from the beginning with a well-known free and open-source license. Since the signal specification definition was mostly perceived as a document, a permissive Creative-Commons Attribution (CC-BY) license was used whereas some of the software had other licenses. This was later unified and the project now continues under one single license, namely the Mozilla Public License, v2 (MPLv2). Using this style of open-source ensures maximum usability of the specification by all companies and this can be an advantage for companies

betting on the VSS, or any derivative thereof.

**Data model & data characteristics**

The hierarchical organization tree-format is the basis of the VSS, like many other data model descriptions.

In addition to this, companies have researched into data ontologies, which are descriptions and organization of data that adds additional metadata including relationships between parts, or relationships between a description of data, and a description of the (sensor) source of that data.

These aspects cannot be encoded in a plain tree but are efficiently added to the tree-like structure of VSS. This is very interesting work that leads to the type of more complete thinking of data relationships that is necessary for the future's efficient data handling.

The most well known data ontology work to us is an extension of VSS. While the work has been ongoing for a while it has recently been referred to by name: VSSo.

For the reason that tree/category/hierarchy is the natural way to chunk up many data items into a working organization - all projects seem to do it in some fashion. Seemingly competing initiatives, are quite naturally also organizing data in a tree-like fashion and the choice of project to use really only falls back on style and desires, but also on the ability to not get stuck within one ecosystem. The latter depends on licensing. Open source licenses cannot take away the usage rights that have been given, and if a project is not extending in the direction that the user wants, there is explicit possibility to take what is there and simply make your own derivative in another direction. This is a future-safe way to handle the choice of a project license.

The VSS project continues as an open source project that encourages additions and change-requests which makes the future open towards any "derivations", renamed databases, or similar, possible while keeping the investment already put into VSS.

**Contents**

The VSS is both a concrete database and a set of standards and tools for how to write

and extend the database. Thus, looking at content only does not give the

full picture. However, the current VSS (open to modification by change requests) has already encoded a number of typical data items in a proposed tree structure. The top level includes:

\* ADAS

\* Body

\* Car

\* Drivetrain

\* OBD

\* Vehicle

\* Cabin

\* Chassis

\* Media

\* Private

This in turn includes sub-trees for examples like:0

\* Cabin, Infotainment, InteriorLights,

\* SingleDoor, SingleHVACStation, SingleShade, ...

\* ExteriorLights, ExteriorMirror...

\* Chassis, Wheel ...

\* BatteryManagement, ElectricMotor, Enginea, FuelCell, FuelSystem, Transmission, ...

etc.

Each data item in the VSS includes name, purpose, data type, unit, and other such metadata: