Vehicle Data Models - Overview & Gap Analysis deliverable

Augmenting the VSS Vehicle Data Access Requirements

Electric Vehicle (EV) Data Set

Since the beginning of 2020, we started looking at how to augment the VSS Data Model with data set covering the Electric Vehicle (EV) (look at the minutes of CCS calls of 27 January and 10 February).

The following docs were pointed out

- Geotab presentation on Commercial EV Management
- W3C VSS vspec: vehicle_signal_specification/blob/master/spec/Drivetrain/BatteryManagement.vspec repository (almost empty for the time being)
- High-Mobility auto-api: vehicle_signal_specification/blob/master/spec/Drivetrain/BatteryManagement.vspec repository
- Geotab EV data access requirements doc is here

From the discussions in the calls, it turned out that the data related to battery performance are highly sensitive and it is likely that OEMs are not goint to share these data although one can consider the data could be described in a proprietary branch of the VSS tree.

For the time being, it is proposed to populate the EV branch of the VSS tree with a set of data used by Geotab in commercial EV management where the data are retrieved from the OBD-II port, knowing other (OEM proprietary) battery data are rather available on the vehicle networks like CAN. The purpose of addition this EV data set is just to tell we are future-proof with the handling of EVs.

Jira ticket TBD

5GAA - V2X data

5GAA related papers for cloud & connected services

Gunnar Andersson please add a conclusion on whether there are some relevant inputs for augmenting the data set in those whitepapers

The text below corresponds to the preparation of the deliverable

---

WARNING

This deliverable has been uploaded to Google Docs, @contributors: please edit the google docs (and not this wiki page) ! Thanks

Analysis of data and signal specifications (use link)

This is a draft with placeholder texts to serve as a container for a project deliverable.

Introduction

- Introduction to the Cloud & Connected Services GENIVI project
- Charter summary
- Scope
- Deliverables

Analysed specifications

- Listing the open source projects that are analysed in this document:
  - Vehicle Signal Specification (VSS)
  - SensorIS
  - Extended Vehicle (ExVe) ISO standard
  - (Connected Vehicle Information Model (CVIM) from AutoMat) - **need to decide how to present this based on the project state**
  - Android Auto vehicle interface **(should there be a Jira task to survey this project separately first?)**
  - **Any other IOT projects?**
- Listing of projects that were examined but not included in the GAP analysis - together with the reasons: **(we need to decide how to handle these two projects, this is just a suggestion)**
  - Connected Vehicle Information Model (CVIM) from AutoMat?
  - Connected Car Consortium

- SAREF Automotive
- the Open Group data exchange standards

Analysis criteria

- What parts of the projects were analysed and to what extent?
  - Motivation & problem space
    - VSS
      - Scope
      - Licenses
      - Applicable to different domains
    - Sensoris
    - CVIM
  - Data model & data characteristics
    - VSS
      - Encoding
      - Technologies
    - Sensoris
      - Encoding
      - Technologies
    - CVIM
    - ...
  - Contents
    - Specification
    - Tooling
    - Source code structure
  - Stakeholders
    - Current status
    - Roadmap
  - Metadata & policies (Unknown User (benjamin_klotz) , Benjamin has a special interest for metadata & policies of all existing projects)

GAP analysis

following above list of analysis criteria

Vehicle Signal Specification (Gunnar Andersson)

Motivation & problem space

Updated text (taken from previous separate document)

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

The first consideration, but as we shall see not the only, was that data signals on CAN networks tended to be completely proprietary and different among every car vendor. The discussion of this may have led to the misunderstanding that VSS would stand or fall only 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, for cloud-based application, and for any place where development APIs need to be made common to increase third-party developer opportunities:

- APIs and agreements are needed on other levels of systems than the low-level CAN bus
- Translations/mappings are not only possible from one level to another, but in every system they are necessary if a standard is not yet accepted throughout. Thus, if CAN networks are not speaking VSS signals directly, they can still be mapped into those (possibly using semi-automated tooling) in order to achieve a standardization on for example third-party application development APIs
- In-car networks have also changed over time, which in any case may replace CAN with Ethernet/TCP-IP style communication, or other types of vehicle-buses. This will drive a different encoding of data since these networks do not have the size and usage limits imposed by the CAN standard. In such cases, having a specification of standard data can help to quickly achieve this change to data encoding that is more optimized for newer network types, rather than starting with a blank paper.

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 would rely on the used signal databases. Of course, more standardization may over time reduce development cost and concerns, but in the shorter time it might incur an expense.

Unfortunately the perception that this only covers, or is fully requiring, standardization of all signals on all CAN networks, has likely slowed the acceptance of VSS as a general idea in the beginning. This is thankfully changing now, as more and more see the full picture

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. Unauthorized access to such features might have a minor to medium impact (boosting engine power) or even more serious (unlocking cars and bypassing start-prevention systems). 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. 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 kept proprietary.

- VSS can standardize "harmless" and common signal data, without requiring *all* car signals to be standardized.
- The ability to write data and affect functions is separate from the ability to read information, and this shall always be controlled by access-control mechanisms thatare applied in the implementations for the signal database
- Finally, we must recognize that agreement on VSS (or other standard) brings with it not only the signal data definitions, but an agreement on the **format** to describe data, including future extensions (common or proprietary), and it furthermore brings a whole set of methods, middleware-implementations, and related tooling that can be shared. It might be that it is in that agreement on standard formats, methods and tools, that even the largest value can be found.
- The acceptance of VSS as an underlying signal description database in the protocol work by the **W3C Automotive Working Group** have proven in practice that the idea behind the hierarchical data description of VSS goes beyond the low-level signalling buses like CAN.

New text...

Domains

In its current form the VSS covers a good selection of varying vehicle data organized in separate sub-trees (see Content chapter). Because of its history, it is relatively focused on the type of data that would either be exchanged between ECUs on a CAN or similar network (i.e. data that useful for several subsystems in the car, as opposed to limited use within one ECU only). There is also some focus on the type of data that external applications (on-board, cloud, web, ...) might find attractive in developing new applications that extend the standard function of the car itself, which matches among other things the needs of the W3C Automotive Working Group specifications of web-protocol access to vehicle data.

The flexible and highly extensible format of VSS opens opportunities. As previously mentioned, the agreement on standard description format, methods and tools, open up for extensions (common or proprietary) to be added and as such there should be no limit to usage of the VSS principles in any car domain. Of course, the wider the usage, the more commonality car companies can expect to achieve across their entire engineering. As noted above, the VSS is also focused on formats, methods and tooling, which is in itself a reusable part.

It might happen that some domains require a different set of data description (more complex data types), or a focus that is more on streaming-data than on the concept of a "signal". While flexible, it is still possible that car companies choose a different method for particularly unique engineering areas, but this would then only complement the VSS standard.

Standard Development Application Programming Interfaces (APIs)

It should also be mentioned that regardless of the feasibility of making 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 necessary to have some kind of data API if data is to be used by applications. Therefore, a standard like that of VSS 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 among the compared standards 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 licensing ensures high usability of the specification by all types of companies and this can be an advantage for companies that bet on the VSS, or any derivative thereof, as it protects against unexpected privatization of the specification.

Stakeholders

Do we write this chapter for each of the standards?

SensorIS (Unknown User (gururaja.n))

Overview

Sensor Interface Specification *Innovation Platform*, SENSORIS, is an open group of significant actors from the global vehicle industry, map and data providers, sensors manufacturers and telecom operators who joined forces, under the form of this *Innovation Platform*, driven by the common vision ~~and belief~~ that, defining an appropriate interface for exchanging information between the in-vehicle sensors and a dedicated cloud as well as between clouds

- enable broad access, delivery and processing of vehicle sensor data
- enable easy exchange of vehicle sensor data between all players
- enable enriched location based services
- drive global growth in this field

Scope

Scope is to deliver and maintain technical specifications defining the format and content of sensor and campaign data:

In Scope

- vehicle-to-cloud data upload format* (vehicle-based data only)
- cloud-to-cloud data exchange format (vehicle-based data and other data needed for mobility services)
- cloud-to-vehicle 'campaign' request format (request for specific data at specific locations and times only)
- conformance to data authorization/authentication process
- conformance to data privacy regulations
- conformance to approved security regulation

*An initial list data item definitions (data model) for some subject areas has also been published by Sensoris, as described in the Data model & data characteristics chapter.

Out of scope

SENSORIS will not:

- define infrastructure or architecture
- establish commercial agreement frameworks for data exchange
- define data exchange for v2v, v2i, i2v (cooperative data) exchange
- define cloud-to-vehicle services

Licenses

SENSORIS specifications will be handled through a dual license model. Every release will be first released internally to the members of SENSORIS under a SENSORIS license. As SENSORIS is committed to be open to the public, all schemas and documentation will be published after a 12 month retention period to the public under the Creative Commons Attribution-NoDerivatives 4.0 International license. Please review the referred license for the exact description of the rights and obligations related to the use of the SENSORIS schemas and documentation.

The published package under this license also contains HTML-documentation as well as the schema description in Google Protocol Buffers (protobuf) language.

What is allowed to do:

- Download the schema from the website.
- Compile a protobuf library using the openly available protobuf compiler.
- Use the protobuf library to implement any software that encodes or decodes SENSORIS messages.
- Define proprietary (non-standardized) Any-extensions.
- Use SENSORIS and your proprietary extensions even commercially between non-member to non-member business cases.
- You can join the consortium and contribute.

This clarification part is not necessary in the GAP analysis.

What is not allowed to do:

- Change the protobuf schema and distribute it to third parties under any name (SENSORIS or other) to exchange data.

Extended Vehicle (ExVe) ISO standard Unknown User (kevinval)

The were three main factors to introduce the ExVe ISO standard:

1. Increasing demand from 3rd parties to access vehicle data and functionality. The workaround to date has been installing additional hardware into the car, which has on one hand has limited scale but more importantly raises the question how security and safety is being handled.
2. OEMs have already equipped vehicles with telematics units and built up IT-infrastructure to handle data connectivity between vehicles and backend systems. To some extent OEMs have offered external parties to integrate with vehicle data using web services, this has been done however without any guidelines or requirements on the system design.
3. As there is active data sharing already, either through external hardware or individual OEM web services, there is a need to define a design and requirements to ensure that security, safety and data privacy is handled with best practices common methods.

The scope of the ISO standard is web services only, which means data offered by OEM backends. It assumes that OEMs have vehicle data available in their backend and applies no requirements on how the data collection is done.

A large part of the standard focuses on authorisation, in other words how user consent should be obtained and maintained. There are several categories of vehicle data; personalised (identifiable to a specific vehicle with a VIN), pseudonymised and anonymised. If we consider a vehicle data point as a resource, in each of these data categories the resource owner depends on the nature of the data and has specific authorisation requirements. All of the data categories are considered in this standard.

Apart from data retrieval the standard also provides requirements and methods for handling modification to the vehicle state through functions.

Licenses

The documentation is provided by the ISO standardisation under a commercial license. The license can be obtained from www.iso.org for the following documents:

- ISO 20078-1 Road Vehicles – Extended Vehicle (ExVe) web services – **Part 1: Content**
- ISO 20078-2 Road Vehicles – Extended Vehicle (ExVe) web services – **Part 2: Access**
- ISO 20078-3 Road Vehicles – Extended Vehicle (ExVe) web services – **Part 3: Security**
- ISO/TR 20078-4 Road Vehicles – Extended Vehicle (ExVe) web services – **Part 4: Control**

CVIM (Unknown User (benjamin_klotz))

Please see draft in this document.

Includes: **Motivation, Scope, Contents, Stakeholders...**

Data model & data characteristics

Vehicle Signal Specification (Gunnar Andersson)

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. Other, seemingly competing initiatives, are quite naturally also organizing data in a tree-like fashion and the differences are in the details, available tooling, license and ecosystem. A smart choice should be done based on style and desire, 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 a specification like this.

The source format is written in YAML which is both machine-readable (with many processing software choices available) and also the most human-readable of similar formats (compare XML, JSON, ...)
This makes it a perfect source format for such a specification. The usage of YAML also invites for future extensions which add additional metadata to each data definition. The VSS project continues as an open source project that encourages additions and change-requests. Its licensing also makes the future open towards any derivations, renamed databases, or similar outlook, possible while keeping the investment already put into VSS.

SensorIS (Unknown User (gururaja.n))

Sensoris has data elements grouped into categories. Any number of categories can be created based on the requirement.

Data Message Content

Data messages contain vehicle sensor data. Data messages communicated from one vehicle of a vehicle fleet to its vehicle cloud contain sensor data from the one vehicle

Identifiers : Several identifiers are used in a SENSORIS message which affect privacy. They allow for identification of a **submitter, session, message, vehicle fleet, vehicle,** and **driver**. All identifiers are optional and are a powerful and fine-grained control instrument for ensuring privacy aspects in SENSORIS session_id, message_id, last_message_of_session, vehicle_fleet_id, vehicle_id, driver_id

Identifiers and referencing

The second set of identifiers is used for cross-referencing events within a message

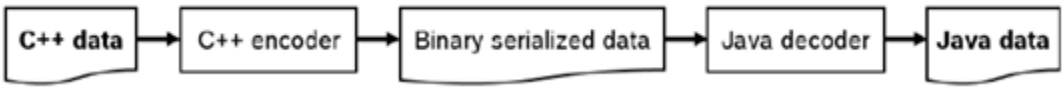
The **event** identifier uniquely identifies an event within a message and is only required if a reference to the event is needed beyond chronological time stamp relation. Event identifiers are of type integer and begin with value 0 and are incremented by 1. The **event relation** protobuf message type enables binary relations between events within a single data message.The **event group** protobuf message type enables smart grouping of events based on the same relative spatial reference system

Some event protobuf message types contain an object identifier which enables referencing between individual events over time

Message Encoding

SENSORIS **job request, job status,** and **data messages** are communicated between the three actor roles vehicle fleet, vehicle cloud, and service cloud. The SENSORIS messages have to be encoded for over-the-air and over-the-wire communication channels, i.e. they have to be serialized by the sender prior to communication and then have to be deserialized by the receiver. Encoding shall support evolution of the data format, i.e. adding new data types or fields shall be backward compatible so that the new data format can be read by both new code and code generated for previous versions of the data format

For SENSORIS version 3 of the **Google Protocolbuf library** is used which adds a streamlined approach for proprietary extensions. The communication from a vehicle of a vehicle fleet to its vehicle cloud is used as example, On the vehicle the obtained sensor data is filled into the C++ data access classes. The class instances are then serialized into a byte array by the also auto-generated C++ encoder. The serialized data is transferred over-the-air to the vehicle cloud. There the auto-generated Java decoder deserializes the byte array into Java class instances having the same schema and sensor data as the C++ class instances on the vehicle. The **protobuf Any5** message type fulfills the requirement for proprietary extensions.



Extended Vehicle (ExVe) ISO standard Unknown User (kevinval)

The ExVe ISO standard does not introduce a data model. In this aspect any data model analysed in this document would fit as a data model in the implementation of a ExVe compatible web service. No ExVe compatible interface that has been introduced to the market today uses any standardized data model.

The standard does however define requirements on the web service interface that is provided to 3rd parties. It has to be RESTful and use the JSON or XML schema. Furthermore the standard includes requirements on several aspects: URI definition, error handling, naming and interaction patterns. All of these are aimed to make the implementation for 3rd parties similar no matter what OEM web service is being consumed.

CVIM (Unknown User (benjamin_klotz))

See (copy from) attached document

Contents

Vehicle Signal Specification (Gunnar Andersson)

Contents (VSS)

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:

- 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 related metadata

SensorIS (Unknown User (gururaja.n))

In the first public version 1.0.0 of the Sensoris Specification the following categories are defined.

Category envelope is the mandatory first field of each category. It follows the google.protobuf.Any format with type_url and value in bytes

Field	Type	Description
envelope	EventGroup.Envelope	Envelope.
localization_category	sensoris.protobuf.categories.localization.LocalizationCategory	Localization category.
object_detection_category	sensoris.protobuf.categories.objectdetection.ObjectDetectionCategory	Object detection category.
weather_category	sensoris.protobuf.categories.weather.WeatherCategory	Weather category.
driving_behavior_category	sensoris.protobuf.categories.drivingbehavior.DrivingBehaviorCategory	Driving behavior category.
intersection_attribution_category	sensoris.protobuf.categories.intersectionattribution.IntersectionAttributionCategory	Intersection attribution category.
road_attribution_category	sensoris.protobuf.categories.roadattribution.RoadAttributionCategory	Road attribution category.
traffic_regulation_category	sensoris.protobuf.categories.trafficregulation.TrafficRegulationCategory	Traffic regulation category.
traffic_events_category	sensoris.protobuf.categories.trafficevents.TrafficEventsCategory	Traffic events category.
traffic_maneuver_category	sensoris.protobuf.categories.trafficmaneuver.TrafficManeuverCategory	Traffic maneuver category.
brake_category	sensoris.protobuf.categories.brake.BrakeCategory	Brake category.
powertrain_category	sensoris.protobuf.categories.powertrain.PowertrainCategory	Powertrain category.
map_category	sensoris.protobuf.categories.map.MapCategory	Map category.

e.g. Localization category has,

Field	Type	Description
envelope	sensoris.protobuf.types.base.CategoryEnvelope	Envelope.
vehicle_position_and_orientation	repeated VehiclePositionAndOrientation	Vehicle position and rotation.
vehicle_odometry	repeated VehicleOdometry	Vehicle odometry.
vehicle_speed	repeated VehicleSpeed	Vehicle speed.
vehicle_acceleration	repeated VehicleAcceleration	Vehicle acceleration.
vehicle_rotation_rate	repeated VehicleRotationRate	Vehicle rotation rate.

Extended Vehicle (ExVe) ISO standard Unknown User (kevinval)

The contents consist purely of specification documents. These include detailed descriptions along with visual diagrams and logical workflow diagrams of how a ExVe web service should be defined. This also covers the system design on an architectural level, meaning how certain logic should be decoupled in different components to ensure a secure implementation of the authorisation process, resources and data access.

CVIM (Unknown User (benjamin_klotz))

Please see draft in this document

Table comparison (if applicable)

Think about if this still makes sense in the final structure

Stakeholders

Vehicle Signal Specification

Definition contributions from JLR, BMW, Volvo Cars (through W3C), Geotab. Implementation in Kuksa project (Bosch – any other Tier 1s/2s?)

SensorIS

Contributions by Here, BMW, Daimler, Volvo, Audi, Full member list https://sensor-is.org/members/. Bosch has the implementation of the spec.

Extended Vehicle (ExVe) ISO standard Unknown User (kevinval)

There is good support for the ExVe standard which is demonstrated by market launch of ExVe compatible web services by several European OEMs already. European OEMs agreed to adopt the ExVe concept in 2016 which was followed by the finalisation and publication of the ISO standard in Q1, 2019.

Possible to clarify more?

CVIM

Mention original contributing companies. No known extension / application of results however.

Metadata & policies

This section describes the metadata used and its openness.

Metadata can be available (pdf documentation), structured, on open formats. It can additionally use ad hoc semantics (e.g. a fixed list of properties), reuse standard semantics (e.g. format properties, units) or be fully linked (e.g. reuse URIs as identifiers between connected entities).

Vehicle Signal Specification (Gunnar Andersson) (see the documentation)

Non-formal metadata is within VSS with a given set of properties (label, comment, min, max…), provided in various open formats. VSS comes with an extension mechanism to create or update branches, signals or attributes.

There are policies in VSS in regard to its units: they should follow first automotive standards, or at least be SI. They are explicitly defined in the documentation.

SensorIS (Unknown User (gururaja.n))

SensorIS uses different serialization open formats for its data, but the current release only has its metadata in protobuf. Units are standard are explicitly defined (e.g. deg_c for Celsius degrees)

There is a policy for extensions (new properties of signals).

Extended Vehicle (ExVe) ISO standard Unknown User (kevinval)

This specification does not define a data model.

The specification has requirement to data formats (JSON, XML Schema), uses URIs and some best practices that candidate data models should follow.

CVIM (Unknown User (benjamin_klotz))

Measurement data comes in packages through data channels. The packages are provided in JSON (open format). Non-formal metadata is supposed to be provided in JSON too but so far only examples are available in the pdf documentation. Data package metadata is composed of a fixed list of properties.

There are no policies to create new properties or standard units.

Table comparison (if applicable) (Unknown User (benjamin_klotz) To Be Confirmed)

Specification	Structure	Semantics	Policies
VSS	Multiple open structured formats	Non-formal fixed set of properties	• Reuse unit standards • Extensions
SensorIS	Open structured format (protobuf)	Non-formal fixed set of properties	Extensions

ExVe	Available documentation (pdf)		Requirements on candidate data models
CVIM	Available documentation (pdf)	Non-formal fixed set of properties	

Conclusions

- We fulfilled purpose of document. (which is ... comparison, finding commonality and drawing conclusions about which standards need to be considered)
- (VSS (or any common approach) is...) appropriate for aftermarket because of common standards so that innovative applications can be made.
- Conclusions about adoption.
- We have taken steps toward a common approach and ability to converge.

References

TBA.

Project Team

TBA.