GENIVI®

TECHNOLOGY BRIEF

DECEMBER 2019
Category: Cloud & Connected Services

Vehicle Data Models – Overview & Gap Analysis

Introduction

With the proliferation of connected cars making them the predominant form of automotive transportation in a few years, a number of parties are working at breaking down the barriers to adoption for mobility services based on automotive data. At the core of such initiatives lie several vehicle data models.

Models describing data produced by vehicles are highly heterogeneous and non-interoperable to the extent that many OEMs use proprietary specifications to define thousands of vehicle signals, different unit systems, modelling patterns and formats. The competition between proprietary solutions and overlapping collaborative efforts results in a fragmented ecosystem of vehicle data models. This fragmented ecosystem is a major motivation for the work of the *GENIVI Cloud and Connected Services (CCS) Project*.

The main goal of this project is to join forces and harmonize activities when designing and implementing the full data-oriented connected vehicle architecture (in-vehicle and back-end). In our charter, we present a list of benefits that could be enabled by this project:

- Enable easy interoperability of building blocks, flexibility and choice
- Develop common solutions and software
- Enable access to all data we want to exchange
- Control access to data
- Enable user privacy and data security
- Clarify responsibilities
- Agree on terminology for improving the shared work around data: names, roles, responsibilities
- Facilitate business opportunities and contractual agreements.

Aligning the data model(s) is crucial for our industry and required to take the next planned step for this project, which is to propose a reference architecture for the vehicle data oriented environment.

Analyzed Specifications

We investigate the topic of Vehicle Data Models in this gap analysis of major initiatives on Cloud and Connected Services that have significant support and a current development:

- The Vehicle Signal Specification (GENIVI)
- SENSORIS
- The Extended Vehicle (ISO)
- Android™ Automotive Vehicle Interface (Vehicle HAL)
- The Common Vehicle Information Model (AutoMat) even though it is no longer actively developed.

Of course, many more initiatives exist. Some of the projects were examined but not included in this gap analysis such as the Connected Car Consortium, SAREF Automotive, Open Group Data Exchange and ViWi. ViWi's main focus is the proposed extensible protocol, but when that proposal was made to the W3C, only some example data model descriptions were also shown and we have no definitive list/source to know all of them. Most of those data areas are covered in more detail in the VSS with the exception of multimedia metadata information (e.g., artist, track, playlist, etc.), which is not our main focus in the CCS project. Other projects were not included because they either did not have yet enough delivered documentation, no updates for years, or little support from the community.

Analysis Criteria

We analyzed the previously mentioned projects in regard to a set of criteria: motivation and problem space, data model and data characteristics, contents of the specification, stakeholders, metadata and policies.

In the *Motivation and Problem Space* criterion, we look into the scope of the specification, its license and if it is applicable to different domains. In the D*ata Model and Data Characteristics* criterion, we look into how data is encoded and which technologies are required. In the C*ontent* criterion, we look into the general specification, its tooling and related implementation, if applicable. In the S*takeholder* criterion, we look into the current list of stakeholders and if a roadmap is known. Finally, in the *Metadata and Policies* criterion, we look into how interoperable the data are, how their semantics are chosen and what policies are included for future users.

The remainder of this document is a presentation of each selected project in regard to our set of criteria.

Motivation and Problem Space

Vehicle Signal Specification (VSS)

Overview

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 driving 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 signals, but this is not the case. As we will see later, data standards like VSS are 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 common to increase third-party developer opportunities and other synergistic effects like:

- Agreement on data definitions and APIs are also needed on other levels of the system, not just the low-level CAN bus
- Translations and mappings are not only *possible* from one level to another, but in every system they are *necessary,* if a single data model is not yet used throughout the entire family of systems. Thus, if CAN networks are not based on VSS signals directly, they can still be mapped into those. Since the VSS signal collection and CAN databases are formally described, it should even be possible to use semi-automated tooling in order to make some of those translations. The potential for third-party application development APIs appears only once translation of data is performed to a common standard.
- In-car network technologies have evolved over time. CAN remains a popular choice for lower bandwidth and local buses but Ethernet/TCP-IP style communication, FlexRay, CAN-XL (in development) and other types of vehicle-buses are also used. The usage of those will drive a different encoding of data since they often do not have the size and usage limits imposed by data encoding into CAN frames. In such cases, having a standard specification of data like VSS can help to quickly achieve this migration

to new data encodings that are more optimized for the newer network types. Without it, it is more challenging since that work essentially starts with a blank sheet of paper.

The proprietary nature of CAN buses is still a reality today and it is a challenge to change this 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 integration concerns also for CAN, but in the immediate timeframe, expect to incur an expense.

The perception that benefiting from VSS requires standardization of all signals on all CAN networks has likely slowed the acceptance of VSS as a good starting point. This is changing now as more and more see that the approach is also well designed for the broader data exchange challenge. It is also possible that additional non-technical and business strategy concerns were hampering the project. For example, it may have been unclear to what extent OEMs would release control over the actual data if there is a common standard. This too is a largely misunderstood aspect of any data access standards and it remains a discussion worth having.

Some of the concerns among OEMs included the current situation of CAN buses being physically 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 perhaps more serious (unlocking cars and bypassing start-prevention systems). Keeping data definitions secret has of course been a very ineffective prevention, since the knowledge of many OEM-proprietary signals exist among proponents of both legal and illegal activities and tooling. The solutions lie instead in proper security architectures. Nonetheless, the main point here is the understanding that VSS outlines a standard for vehicle data that OEMs can agree upon, as well as a methodology and tooling, whereas methodology and tooling is still also applicable also 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 that are applied in the implementations for the signal database
- Finally, we should recognize that agreement on VSS (or any 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 brings a whole set of methods, middleware-implementations, and related tooling that can still be shared. It might be that it is in that agreement on standard formats, methods and tools that even the greatest value can be found.
- Finally, the acceptance of VSS as an underlying signal description database in the web-protocol work by the **W3C Automotive Working Group** have proven that the hierarchical and extensible data description of VSS goes far beyond the low-level signaling buses like CAN.

Scope

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 are 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 (i.e., on-board, cloud and 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 are in themselves a reusable part. They open up for extensions (common or proprietary) to be added and, therefore, 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 activities.

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)

Translating data descriptions using a shared standard at some level of the system would open up for programming standards above it, both within the car (application API) and outside (big data exchange, cloud/web applications or other). It is fundamentally necessary to have some kind of known data API if data is to be used by applications. Therefore, a standard like VSS would be necessary *somewhere,* even if there is a translation required from, for example, the CAN level to this API level. If the desire is there to create a third party application development ecosystem, then those developers require some standardization of those APIs so that they are consistent. A standard like VSS in effect defines a shared API, regardless of whether the low-level data is fully equal across adopters.

The protocol work by the W3C Automotive Working group is one initiative that aims to enable such a common standard for third party developers and other VSS-based initiatives are underway.

Licensing

The licensing of VSS appears unique among similar projects 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 and related tools.

SENSORIS

Overview

Sensor Interface Specification *Innovation Platform*, SENSORIS, is a 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 that, defining an appropriate interface for exchanging information between the in-vehicle sensors and a dedicated cloud as well as between clouds would:

- 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

SENSORIS' 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 and 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

** We interpret this as meaning that the low-level communication mechanism is out of scope, rather than the data definitions that may be exchanged between v2v/v2i/i2v.*

Licensing

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.

The following clarification of the license is provided in SENSORIS specifications as follows:

What is allowed:

- 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
- Join the consortium and contribute

What is not allowed:

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

Extended Vehicle (ExVe) ISO Standard

Overview

There were three main factors for introducing the ExVe ISO standard:

1. Increasing demand from third parties to access vehicle data and functionality. The workaround to date has been installing additional hardware into the car, which has limited scale but more importantly raises the question of 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 and common methods.

Scope

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

A large part of the standard focuses on authorization, in other words how user consent should be obtained and maintained. There are several categories of vehicle data: personalized (identifiable to a specific vehicle with a VIN), pseudonymized, and anonymized. If we consider a vehicle data point as a resource, in each of these data categories the resource controller depends on the nature of the data and has specific authorization 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.

Licensing

The documentation is provided by the ISO body 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**

Common Vehicle Information Model (CVIM)

Overview

The AutoMat project is focused on Big Data and how to enable an open ecosystem driven by vehicle data. The project provides an end-to-end framework starting from data capturing in the vehicle to data consumption by third parties through marketplaces.

Scope

CVIM is focusing on enabling Automotive Big Data Marketplaces for Innovative Cross-sectorial Vehicle Data Services.

Data harmonization is seen as a key factor for enabling AutoMat; therefore, CVIM was developed to form a common understanding of data and information formats.

Out of Scope

- Signal vocabulary
- Data type specification/restriction
- CVIM also does not define any minimum datasets that an OEM should deliver

Licensing

Android Automotive Vehicle Interface (Vehicle HAL)

Overview

It is important to consider the complete solution for vehicle data exchange. It starts where data is produced (i.e., typically the in-car software and electrical architecture). The cloud connectivity standards should therefore adapt to existing technology used in the vehicle electrical architectures, as well as to influence those technology choices for the same purpose effect (i.e., to achieve more commonality).

GENIVI runs other projects that deal with this “big picture” thinking and some of them relate to the Cloud and Connected Services. This relationship includes minimizing differences among execution environments, and reducing efforts to connect the standards for cloud data exchange as discussed here, with the actual in-car technical systems that also consume or produce such data. A typical OEM vehicle electrical architecture runs a number of different operating systems and platforms that need to cooperate with data and need to use common solutions where possible. Android Automotive is an increasingly popular choice for the Infotainment system and less considered in other types of ECUs.

Android Automotive Current Approach

The current versions of Android publish a Vehicle Hardware Abstraction Layer (Vehicle HAL, VHAL) concept that intends to expose certain vehicle-specific functions and properties to the Android system and application.

One primary feature is the list of vehicle properties that are required to be implemented. They are defined as static list of properties, with unique numerical IDs.

Example from source code:

```
public static final int INVALID = 0;
public static final int INFO_VIN = 286261504;
public static final int INFO_MAKE = 286261505;
public static final int INFO_MODEL_YEAR = 289407235;
public static final int INFO_FUEL_CAPACITY = 291504388;
…

public static final int PERF_ODOMETER = 291504644;
public static final int PERF_VEHICLE_SPEED = 291504647;
public static final int ENGINE_OIL_LEVEL = 289407747;
etc.
```

There are also particular services for particular car subsystems. E.g. HVAC car-lib/src/android/car/hardware/hvac/CarHvacManager.java with its own properties:

```
public static final int ID_ZONED_SEAT_TEMP = 0x1540050b;
public static final int ID_ZONED_AC_ON = 0x15200505;
```

GENIVI AA-SIG

The Android Automotive Special Interest Group (AA-SIG) in GENIVI is an example of a project that deals with improving and aligning the technologies used in the in-car systems. It discusses extensions that enable access to a very wide set of vehicle data, ideally described by the same standards as the cloud-connection, while working to align this with Android certification requirements and maintained compatibility with existing Android methods.

Scope

Android Automotive Current Approach

The scope of the standard Vehicle Properties in Android is varied across a few different areas but seems to be primarily a selection of only the most commonly used or needed data properties for typical in-car Android applications. As expected, it is also changing between versions – i.e., adapted over time but still focused on what the Android system needs. Naturally, since this is what is required of every VHAL implementation, the data exchange can be assumed to also be designed for the needs of Google-specific apps and automotive related Google services.

GENIVI AA-SIG

The AA-SIG work currently focuses on the definition of technical solutions for connecting a VSS-based data description to application usage in Android for Automotive.

This includes solutions that expose data to the same official Vehicle Properties as defined by Android, using libraries/servers that provide and translate data that arrived into the system in according to VSS data definition / data type and unit. At the least, basing the implementation of the VHAL, which every product must perform, on a data standard below it and providing software design for the solution would facilitate the implementation of Compatibility Test Suite (CTS) compliant HALs among many vendors. While everything below the HAL is by design considered a unique implementation and not part of the Android common source code (this is the typical usage of an *abstraction layer*) creating standards in how to achieve it could even enable sharing implementation among vendors.

In the bigger picture, this work can provide solutions that provides the entire VSS defined data model to Android applications (as a major extension to what Android Vehicle Properties define today), just as this level of data connectivity may be available in other systems/platforms and in cloud-connectivity scenarios.

Licensing

Android Automotive Current Approach

The information about VHAL is part of the Android project and provided there, in source code and documentation. The Android Open-Source Project (AOSP) code is under a variety of open-source licenses, whereas development of new functionality and future versions tend to be discussed between Google and partners only, and this is the same for the automotive functionality. Rather than risking an incorrect summary of the licenses here, we refer you to the corresponding web sites (e.g., https://source.android.com).

GENIVI AA-SIG

For the AA-SIG facilitated by GENIVI, the discussions occur in a public GENIVI project, open for any potential Android Automotive adopters to join. The results are documented in slides and minutes on public Wiki pages. The proposed designs and solutions may be supported by code development, which would then likely be licensed using GENIVI default open-source license choice (MPLv2). The results are also expected to affect mainline Android development later on -- development that is usually done by Google (taking into account input received from car companies) and then released into the Android Open-Source Project (AOSP) when a new release of Android is published. Adjustments of the software license may then occur, if required.

Data Models and Characteristics

Vehicle Signal Specification (VSS)

Data Representation

Like many other data model descriptions, the hierarchical organization tree-format is the basis of the VSS. In addition to this, companies have researched data ontologies, which are descriptions and organization of data that add additional metadata, including in particular the relationships between parts. There are many potential relationships but one example is a description of data and a description of the source (sensors) of that data.

These aspects cannot be encoded in the original tree because of being different in different cars, because of the one-to-many or many-to-one relationships, or because it can change over time. The information is however efficiently added as additions to the tree-like structure of VSS. This is very interesting work that leads to the type of more complete data relationships that are necessary for the future's efficient data handling.

The only 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.

VSS, like many others, organize data. In VSS there are no limits placed on the depth of the tree hierarchy. Recent extensions deal with the possibility to define instances of nodes efficiently, since the duplication of identical but distinct data items is quite common (think for example about an identical sensor at each wheel).

A recently discussed proposal named *VSS Layers* is designed to enable augmenting or overriding metadata to cover a wide variety of categorization of data items. See metadata and policies chapter for more details.

Technologies

The source format is written in the markup language YAML, which is a popular plain-text format that is both machine-readable (with many processing software choices available) and considered the most human-readable of similar formats (i.e., compared to XML, JSON, etc.).

This makes it a perfect source format for such a specification. The usage of YAML invites 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

Data Representation

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 that 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

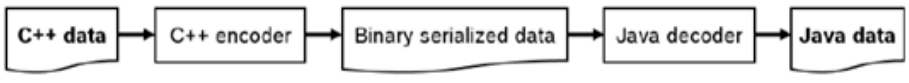
The second set of identifiers is used for cross-referencing events within a message. The **event** identifier uniquely identifies an event within a message and is only required if a reference to the event is needed beyond chronological time stamp relation. Event identifiers are of type integer and begin 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, that is, 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.

Technologies

For SENSORIS version 3 of the **Google protobuf 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 an 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

Data Representation

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

Technologies

The standard does, however, define requirements on the web service interface that is provided to third 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 third parties similar no matter what OEM web service is being consumed.

Common Vehicle Information Model (CVIM)

Data Representation

CVIM holds a specification for three different layers that each have a JSON representation:

- **Signal**: Primitive data structures within the vehicle that contain information, such as vehicle odometer reading, speed or VIN. Can be interpreted of data typically sent through the CAN-bus.
- **Measurement Channel**: This layer describes how many signals was measured and in which frequency. It also can include GPS coordinates as meta information. A measurement channel encapsulates many signals.
- **Data Package**: The data package is an overall container that encapsulates many measurement channels. It contains important meta information such as timestamps, data ownership, CVIM version and a cryptographic signature generated by the vehicle when putting together the package.

Measurement channels can output time series, histograms, geo-histograms.

Technologies

Those messages are defined in JSON-schema and made for JSON data, with several formats:

- (u)int8/16/32/64
- Double
- Date(-time)
- Uuid
- Email
- Version

Android Automotive Vehicle Interface (Vehicle HAL)

Data Representation

Android Automotive Current Approach

There are currently 80-100 Vehicle Properties that are defined as part of the Android API whereas hundreds or thousands of signals are expected to be feasible in a full VSS tree. The discussions in the Android Automotive SIG deal with how VSS defined signals can be mapped to those properties.

GENIVI AA-SIG

Group consensus has evolved to use the VSS is the primary database for vehicle signals in the discussion of how to access vehicle data from Android systems in the vehicle (including how to also support the standard Android Automotive defined vehicle properties from a common implementation).

Therefore, please refer to the chapter on VSS for Data Representation details.

Technologies

Android Automotive Current Approach

Android is a very code-driven project and the technology used are typically to simply define the programming API, or in this case the available Vehicle Properties in the source itself, and to complement this with documentation as needed. There seems to be no particular separate text or database model defining the vehicle properties - they are simply provided as part of the Android API, in source and docs.

It is however possible that some of the standard Android development such methods and tooling, such as the Android IDL (AIDL) are involved to some extent when defining these vehicle centric APIs.

GENIVI AA-SIG

Technologies include a mix of those defined for VSS (refer to the chapter) and those defined in the standard Android software architecture. (For more details, see Tools and related implementations).

Contents

Vehicle Signal Specification (VSS)

Specification

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*

*VSS Layers is intended to cover data categorization (of many different kinds), access-control rules, local laws, and other metadata.

Tools and Related Implementations

There are tools to convert the VSS source format (YAML) into other formats as needed, such as JSON, CSV (spreadsheet file), These are available in the tools directory in the VSS source code repository

- Tools directory of the VSS repository [VSS repository]
- W3C protocol “Generation 2” implementation (Melco and Geotab and Volvo ...)
- W3C protocol (VISS, version 1) implementation, with recent addition of experimental REST interface, in Eclipse KUKSA project. Ref: [Email: “New REST API support in w3c-visserver-api“] (https://lists.w3.org/Archives/Public/public-automotive/2019Nov/0019.html).

SENSORIS

Specification

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.

Tools and Related Implementations

The Here SDK named Open Location Platform (OLP) covers ways to access information related to SENSORIS defined information.

Here OLP includes the neutral server concept:

- Access to vehicle sensor data - the Marketplace can now act as a secure, neutral, GDPR-compliant hub for data consumers to gain access to vehicle data from participating automotive manufacturers
- Consent management - ensures privacy for owners and drivers of vehicles, by allowing them to grant and revoke consent for specific third-parties to access data generated by their vehicles
- ISO 20078 "ExVe" compliant interface - simplifies platform integration for data providers and consumers

Extended Vehicle (ExVe) ISO Standard

Specification

The contents consist purely of specification documents. These include detailed descriptions along with visual diagrams and logical workflow diagrams of how an 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 authorization process, resources and data access.

Tools and Related Implementations

No tooling is provided to support the implementation of the specification and no source code or references are provided to support the implementation of the specification. The offering of tools and implementation is left to the market (e.g., Here OLP above)

Common Vehicle Information Model (CVIM)

Specification

The specification consists in several documents defining: the CVIM open specification, a cyber security framework, a business model approach and prototype reports.

Tools and Related Implementations

A code repository for CVIM developers is provided on GitHub https://github.com/automat-project/SDK that implements the specification in the Swagger (OpenAPI) format. Together with OpenAPI/Swagger tools, it's possible to use this repository to generate both server and client side code for a REST interface. In order to test the API requests a marketplace reference implementation (also delivered within the AutoMat project) should be used.

The SDK structure follows the OpenAPI/Swagger format, which is widely used for REST interface descriptions.

Android Automotive Vehicle Interface (Vehicle HAL)

Specification

Android Automotive Current Approach

Vehicle data is defined as part of the Vehicle Hardware Abstraction Layer defined in recent Android versions. The Vehicle Properties are documented in the public documentation for Android.

GENIVI AA-SIG

Discussion of improved data access methods in Android is currently a work in progress. There is therefore no formal specification, but the ideas are presented in public slide-decks and minutes from the AA-SIG meetings.

Tools and Related Implementations

Android Automotive Current Approach

There is no known tooling to implement the Vehicle HAL itself, i.e., to implement the mapping between the car electrical platform and automotive signal buses to the vehicle properties inside Android.

Application developers get access to the properties through existing and documented APIs so it needs no additional tooling than the normal way to do Android application development.

Extensions to Vehicle properties need to be described in the HIDL interface description language so that existing tooling can be used to make this part of the vehicle HAL.

GENIVI AA-SIG

Refer to the VSS chapter for references to general VSS-related tooling.

Overall, the AA-SIG takes a “big-picture” approach and also discusses the diversity of operating systems and finding standards that match the whole car system while also being appropriate solutions for the Android-based systems. Because of this, a large set of existing and potentially new technologies are discussed in the areas of bindings to existing data exchange protocols in the car and beyond (SOME/IP and W3C-web protocols, and others), and how to balance a data-model based programming with traditional APIs often defined in Franca IDL. From this wide perspective, various code generation tools and bindings that exist (Common-API C++) are discussed, and potentially new ways to automatically create the bindings of such common car interfaces into the standard Android development methods that includes AIDL/HIDL descriptions of interfaces, to support the application development that uses Java and generally fit into the Android standard software architecture:

Examples:

- VSS to Franca observable Attributes translator (exists, proof of concept)
- Franca IDL to AUTOSAR-XML (The [FARACON] tool)
  -> (Together these in theory makes up VSS -> AUTOSAR-XML, which enables SOME/IP access)
- Direct VSS to observable properties in AUTOSAR-XML translation instead (future possibility)
- Discussion about how various remote ECU services can be mapped into Android Service layer, in other words how to produce a familiar Android Application development API, including access control tied to the standard Android permissions system.

Stakeholders

Vehicle Signal Specification (VSS)

Participants

The VSS is being increasingly used as the basis for a number of data-oriented collaboration projects. Since the first version of the associated **W3C** web protocols, a.k.a. the VISS (v1) specification and continuing with ongoing “Generation 2” work, the VSS is the primary vehicle data representation format, as well as the intended list of standard signals for a compliant implementation.

Participation in implementing the resulting W3C protocol standards could be one way to measure, although it does not always translate to active participation in growing the VSS itself, it suggests a future desire to do so.

While not all implementations might be known, the official list of implementation of the first version of web protocol is provided in W3C wiki infrastructure at VISS implementations

Another initiative that is currently in its build phase and led from the W3C automotive business group is the intention to collect historical data measurements from a few OEMs, anonymize the data where needed, describe it using VSS definitions of those data items, and make the data set available on a joint server for open research into methods of querying and using the data in interesting ways.

The Android Automotive Special Interest Group (AA-SIG) in the GENIVI Alliance simultaneously discusses how to bring a full set of vehicle data into Android implementations. However, in doing so the group must also take on the discussion about how to achieve standards based on VSS in the entire vehicle network. Android is but one implementation technology, primarily for IVI, whereas the majority of other ECUs in the system also need to exchange data. Participants include Tier 1 suppliers, technology vendors and at least two major car OEMs.

Current Status

VSS is a mature but developing standard available on a GENIVI public code repository https://github.com/genivi/vehicle_signal_specification.

Roadmap

Like many open and collaborative projects, the roadmap is influenced by what companies decide to do with it and in this several spin-off projects have been started.

- Vehicle Data Ontology work has been shown and is the basis for a doctoral thesis and there is continued interest for this direction. This work is presented in the Abstracting and Interacting with Vehicles in the Web of Things paper. The collection of VSS described sample data into a server to try out data based applications and graph query technologies, as discussed by the W3C business group
- Additional implementations of VISS (a.k.a. v1) and Generation 2 of W3C-specified protocols
- The build-out of the flexible **VSS Layers proposal,** which has the potential to define access control lists, technical “contracts” between systems, policies for which data is privacy-sensitive and more.

The VSS project itself intends to continue to promote an open-licensed and common single data definition standard including agreement on the standard signals that shall be expected on all platforms, and the potential for adopters to extend the database to cover future yet unknown needs.

- Continued collaboration with W3C web-protocols
- Promotion across the entire in-vehicle and cloud

SENSORIS

Participants

SENSORIS is a membership and business-driven innovation platform. Managed by ERTICO – ITS Europe. SENSORIS represents a group of 28 key players from the global vehicle industry, map and data providers, sensors manufacturers and telecom operators. There are 4 WGs each led by HERE, Elekrobit, Daimler and Continental.

The complete list of member companies can be found at https://sensor-is.org/members/.

Current Status

SENSORIS has released version 1.0.0 of the specification that is available at https://sensor-is.org/wp-content/uploads/sites/21/2019/07/sensoris-specification-v1.0.0-public-1.zip.

Roadmap

Expect updates to the Specification.

Extended Vehicle (ExVe) ISO Standard

Participants

The standardization has been driven by OEMs through the European Automobile Manufacturers Association (ACEA) and the German Association of the Automotive Industry (VDA).

Current Status

The first version of the standardization was published as an ISO standard in Q1 2019.

Roadmap

The first OEMs who have launched Extended Vehicle compatible interfaces are BMW and Mercedes-Benz.

Common Vehicle Information Model (CVIM)

Participants

AutoMat Project - Consortium members:

- Volkswagen AG (VW, Coordinator), Germany
- Renault SAS (RSA), France
- Centro Ricerche Fiat SCPA (CRF), Italy
- Institut für angewandte Systemtechnik Bremen GmbH (ATB), Germany
- ERPC European Research Programme Consulting Gmbh (ERPC), Germany
- Technische Universität Dortmund (TUDO), Germany
- ATOS Spain SA (ATOS), Spain
- Institut Mines-Telecom (IMT), France
- Trialog (TRIALOG), France
- Here Global B.V (HERE), Netherlands,
- Meteologix AG (METEOLX), Switzerland

Current Status

The project is finished and has delivered final results.

Roadmap

The project has been discontinued.

Android Automotive Vehicle Interface (Vehicle HAL)

Participants

Several car OEMs have announced their intention to put Android Automotive in production within the next years. OEM adopters provide input individually in conversations with Google, which presumably affects development of future versions, including the Vehicle HAL and Vehicle Properties.

Current Status

According to Android Release 10.0.

Roadmap

Expecting additional growth and changes. There is no public roadmap. As noted, we expect Android adopters discuss future changes with the Android development team at Google.

Metadata and Policies

This section describes the metadata used and their openness as described in the 5-star open data.

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 (VSS)

Non-formal metadata is within VSS with a given set of properties (i.e., label, comment, min, max, etc.) that are 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 automotive standards if there is a wide consensus on a unit. Otherwise, the default usage is to use SI-units. Most of the generic preferred units are explicitly defined in the documentation.

The VSS Layers concept is proposed to allow multiple separate similarly formatted files to encode different aspects of the data in a particular usage. This makes it possible to keep metadata that changes over time, or are different in particular systems, or varying in the markets where the product is sold.

VSS Layers could cover:

- Data categories
- Access-control rules
- Laws governing usage of certain data
- Data criticality for safety or other concerns
- Precision, Quality, Reliability and other qualitative attributes

It is likely that the extensions that VSS Layers enable also bring additional policies for the usage of those extensions, both some that are agreed in the standard/collaborative specification and some that are local within companies.

SENSORIS

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

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.

Common Vehicle Information Model (CVIM)

The signal specification includes metadata entries related to signals themselves:

- Name
- Comment
- Type (numeric, enumeration, information)
- Unit
- Min
- Max
- Items

Signals additionally include metadata entries related to signal usage:

- Rate
- Resolution

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.

Android Automotive Vehicle Interface (Vehicle HAL)

As a source code centered project, Android puts less emphasis on the formality of each set of metadata and mostly describes what is possible to do from the programming perspective. The vehicle properties however have some structure defined, as shown in the documentation. Properties can be tied to a zone so that they can be addressed as instances (e.g., left and right instance of the same concept). Properties also have a defined computational data type (string, integer, float and so on) and as general documentation of course the significance (corresponding physical unit) of the provided value is also described, such as if speed is defined in km/h or miles/h.

Comparison of Metadata and Policies

A summary of metadata specifications and policies for the previously described initiative is presented below.

Specification	Structure	Semantics	Policies
Vehicle Signal Specification (VSS)	Open structured source format (YAML) with transformations to other formats.	Non-formal fixed set of properties	Reuse SI and automotive unit standards Extensions are separate from standard/required data items
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	
Android Automotive Vehicle HAL	Flat list of properties. Web documentation of API for VHAL. Implementation in open source code	Non-formal explanation in document web page	Organic growth, defined by Google based on input from Android adopters

Summarizing Table

The table below compares data models according to four key criteria. CVIM data model is excluded because the hosting project is discontinued.

Criterion	Stakeholders	Motivation and problem space	Data model	Metadata
VSS/VISS	GENIVI and W3C, contributions from JLR, Kuksa, BMW, Volvo, Geotab...	Develop service specifications for exposing vehicle data and other information around vehicle centric functions. Not define or mandate implementation details including vehicle, network or sensor protocols	Vehicle Signal Specification (VSS) as the per default model Alternative data models possible	Vehicle Signal Specification (VSS): Extension mechanism Modeling best practices for signals and attributes
SENSORIS	ADAS, service providers, OEMs, navigation suppliers, telecoms...	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	Data messages in categories (which you can create) Identifies of submitter, session, message, vehicle fleet, vehicle, and driver Developed in Google Protobuf library	Units explicitly defined (e.g., “deg_c” for Celsius degrees) Policy for category extension to be compatible
ExVe	ISO, European OEMs contributing	Increasing demand from 3rd parties to access vehicle data and functionality OEMs already equipped vehicles with telematics units and IT-infrastructure to handle connectivity Need to define a design and requirements to ensure that security, safety and data privacy (best practices, common methods)	For 3rd parties to implement RESTful with JSON or XML schema with requirements on several aspects: URI definition, error handling, Naming, interaction pattern	Policies: requirements for 3rd parties on data modeling good practices (e.g., URI use)
Android Automotive VHAL	Several car OEMs have announced their intention to put	To provide the data needs for Google Automotive	Independent and Android-specific definition of data, as	Only basic type and explanation. (Un)availability flag

	Android Automotive in production within the next few years.	Services, and for third-party car-specific apps in Android Automotive based Infotainment systems.	part of the specified Vehicle HAL	for each item. A small selection of Zones can define their own instances that can be addressed in interaction. E.g., Zone Wheels -> address Right or Left Wheel or both. Coarse-grained connection to app-permission system.

Conclusions

In this document, we explored five projects in regard to a set of analysis criteria: motivation and problem space, data model and data characteristics, contents of the specification, stakeholders, metadata and policies.

We found commonalities among these projects that lead to decisions of either joining efforts on common needs or drawing the line between specification that have different scopes. For instance, we see no need for competition between the VSS model and other projects' data models in that these initiatives have modular data models (ExVe, SENSORIS, CVIM) or are still under development and have a need for a data model (Android HAL).

We see additionally that a shared data model across the industry, exemplified by VSS, would be appropriate for aftermarket because of common standards so that innovative applications can be developed.

The contributing members and stakeholders of the projects presented in this document show that there is a global interest in common specifications.

In the GENIVI Cloud and Connected Services project, and other projects, we are working with the assumption that the VSS is the common data model. However, the work done in the architectural design track of the project is done in a way that is not strictly tied to a single data model.

References

- [5-star open data] https://5stardata.info/
- [Abstracting and Interacting with Vehicles in the Web of Things] https://fr.slideshare.net/BenjaminKlotz2
- [Android Automotive Vehicle Properties] https://source.android.com/devices/automotive/properties
- [Common Vehicle Model Specification] https://automat-project.eu/content/open-cvim-specification
- [FARACON] Franca-to-AUTOSAR-XML translation https://github.com/GENIVI/franca_ara_tools
- [HERE OpenLocation Platform] https://developer.here.com/blog/open-location-platform-2.8-release
- [SENSORIS specification] https://sensor-is.org/wp-content/uploads/sites/21/2019/07/sensoris-specification-v1.0.0-public-1.zip
- [ISO 20078 Road vehicles — Extended vehicle (ExVe) web services] https://www.iso.org/standard/66978.html
- [Vehicle Properties in Android Automotive] https://source.android.com/devices/automotive/properties
- [W3C VISS implementations] https://www.w3.org/auto/wg/wiki/VISS_implementations

Authors

- Kevin Valdek, High Mobility
- Benjamin Klotz, Eurecom
- Narasimha Swamy Gururaja, Bosch
- Gunnar Andersson, GENIVI
- Participants of the GENIVI Cloud and Connected Services project