GENIVI Technical Summit 2019

Vehicle Hardware Abstraction Layer (HAL) Design

12 November 2019
Vehicle Hardware Abstraction Layer (VHAL) Design

AOSP Overview

- Android is an open source, Linux-based software stack created for a wide array of devices and form factors.
- Android's primary purpose is to create an open software platform.
- The objective is a shared product that each contributor can tailor and customize.
- Uncontrolled customization can lead to incompatible implementations. To prevent this, the Android Open Source Project (AOSP) maintains the Android Compatibility Program.
- Besides infotainment tasks, AOSP aims to handle vehicle-specific functions.

**Key Architecture Aspect:**

- AIDL: allows you to define the programming interface that both the client and service agree upon in order to communicate with each other using inter-process communication (IPC).
- Hardware Abstraction Layer (HAL) provides standard interfaces that expose device hardware capabilities to the higher-level.
- Binder IPC – is the backbone of Android communication system.
- TREBLE: OS Framework update without affecting lower layers and applications.
Vehicle Hardware Abstraction Layer (VHAL) Design

Vehicle HAL & Vehicle Properties

- Vehicle Hardware Abstraction Layer (VHAL) interface defines the properties OEMs can implement.
- Each property is uniquely identified by an int32 key and has a predefined type.
- Each Property is defined with following attributes:
  - Area Type
  - Zone: Each zoned property must use pre-defined area type. Each area type has a set of bit flags defined in an enum for the area type.
  - Area ID: Each zoned property may support one or more Area IDs
- Android provisions to define vendor specific properties with VENDOR as group.
Vehicle Hardware Abstraction Layer (VHAL) Design

Vehicle HAL & Vehicle Properties

• Every property value comes with a VehiclePropertyStatus value. This indicates the current status for the property:
  • AVAILABLE
  • UNAVAILABLE and
  • ERROR

• The VHAL uses the following interfaces
  • Getter API
  • Setter API
  • Subscribe
  • Callbacks
Vehicle Hardware Abstraction Layer (VHAL) Design

Problem statement

• Currently, Android defines minimal set of vehicle properties, interfaces and data types to access vehicle data. For OEMs, in order to access the vehicle data from other ECUs, custom vehicle properties and custom HALs needs to be implemented; which leads to inconsistent implementation vehicle data access.

• For accessing the vehicle property metadata, AOSP enforces the architecture to fulfill the interface requirements for the vehicle HAL.

• OEMs need a simpler, scalable and reusable approach to access aggregated vehicle data. This creates and enables App development infrastructure for better user experience.

• So, in order to address these concerns, is there a solution to define scalable, reusable vehicle data model and interface to access it?

• Does it makes sense to standardize data model and interface to access?
Do you see VSS as a data centric abstraction standard?

<table>
<thead>
<tr>
<th>Android Automotive</th>
<th>CE Device (Android / iOS)</th>
<th>Other ECUs (Linux / QNX / Autosar)</th>
<th>W3C Vehicle API (VSS)</th>
<th>OEM Specific</th>
</tr>
</thead>
</table>

- **Data centric model (understandable by 3rd party)**
- **Service driven API**

**Vehicle**

- If the answer is “No”:
  Alternatives? Missing features? Requirements to be considered?
- If the answer is “Yes”:
  Status of current VSS standard? GENIVI VSS standardization working model? APIs to consider?
Data centric API as a reusable Assert

How to expose car network the right way?

- Which requirements have to be fulfilled by the solution?
- What is our target architecture? (document advantages and disadvantages for each arch. proposal)
- Contribution by GENIVI?

<table>
<thead>
<tr>
<th>Android Automotive</th>
<th>CE Device (Android / iOS)</th>
<th>Other ECUs (Linux / QNX / Autosar)</th>
<th>W3C Vehicle API (VSS)</th>
<th>OEM Specific</th>
</tr>
</thead>
</table>

Simple car data access (3rd party developers)

Data centric model (understandable by 3rd party)

Vehicle

Service driven API
Architectural proposal I (via custom HAL)

Legend

- Green: Google
- Light Blue: Genivi

Genivi aware components will have access to complete set of VSS exposed by VISS server in Android fashioned way.

VISS server deployment is not limited to Android only. E.g., can live also on other partition in multi-OS solution with interface exposed. Any adaptations to server deployment are required from VISS HAL component only.

Provides properties defined and required by Android in Vehicle HIDL spec. This is subset of VSS set served by VISS server.

Provides complete set of properties served by VISS server (VSS set). Genivi aware components will have access to complete set of VSS exposed by VISS server in Android fashioned way.

CTS and VTS test suites are required for Android certification.

VISS server deployment is not limited to Android only. E.g., can live also on other partition in multi-OS solution with interface exposed. Any adaptations to server deployment are required from VISS HAL component only.
Architectural proposal II (via Vehicle HAL)

**Contributions:**
- Google
- Genivi
- OEM / Tier1

**VSS tree:**
- Vehicle.Body.Hood
- Vehicle.Body.Trunk
- Vehicle.Body.Windshield

**VSS tree is a base for the GraphQL schema:**
- create table trunk

**VSS tree is a base for the data structure:**
- e.g. create table trunk

**This component is responsible to read data from car network in an OEM specific way and feed this data to the VSS Database:**

**CTS and VTS test suites are required for Android certification:**

**VSS tree is a base for the android Vehicle HAL property extension:**
- e.g. Vehicle.Body.Hood.IsOpen

**Provides properties defined and required by Android in Vehicle HAL spec. and complete set of properties provided by VSS data tree**
Architectural proposal III (via GraphQL Server)

**Comments:**
- GraphQL could be any protocol
- Unix Socket = can be network socket

**VSS tree:**
- Vehicle.Body.Hood
- Vehicle.Body.Trunk
- Vehicle.Body.Windshield

This component is responsible to read data from car network in an OEM specific way and feed this data to the VSS Database.
Following architectural designees have been made based on the discussions during the Android Technical Summit in US.
Architectural proposal I (via Vehicle HAL)

Contribution

- Google
- Genivi
- OEM / Tier1

VSS tree:
- Vehicle.Body.Hood
- Vehicle.Body.Trunk
- Vehicle.Body.Windshield

This component is responsible to read data from car network in an OEM specific way and feed this data to the VSS database.

VSS tree is a base for the data structure.

e.g. create table trunk

VSS Feeder

VSS Database

Hal layer

VSS tree is a base for the android Vehicle HAL property extension.

e.g. Vehicle.Body.Hood.IsOpen

CTS and VTS test suites are required for Android certification.

VSS Feeder

VSS Database

CTI

VTS

Provides properties defined and required by Android in Vehicle HAL spec. and complete set of properties provided by VSS data tree.

VSS tree is a base for the android Vehicle HAL property extension.

e.g. Vehicle.Body.Hood.IsOpen
Architectural proposal II (via Data Server)

Defined by
- Google
- Genivi
- OEM / Tier1

Comments:
- One idea was proposed that a helper class in Android Framework layer implements the network protocol instead (App is calling a Framework API)...
  - but bundling this network binding code with the app was also considered OK by some.

**VSS tree is a base for the GraphQL schema**

VSS tree:
- Vehicle.Body.Hood
- Vehicle.Body.Trunk
- Vehicle.Body.Windshield

WebSocket

This component is responsible to read data from car network in an OEM specific way and feed this data to the VSS Database.
Architectural proposal III (via Global SomeIP Service)

Defined by
- Google
- Genivi
- OEM / Tier1

ECU 1..N

SomeIP Service A

SomeIP Service B

HAL layer

Service A

Service B

Framework layer

Java API

Manager A

Manager B

Generic SomeIP service

SomeIP

Manager A

Manager B

Java API

HIDL

Car AIDL

VehicleHAL

HIDL

Application layer

Head Unit (on-board)

App A

App B

App 1

App n

«App»

App 1

App n

«App»

VehicleHAL AIDL

Vehicle HAL

HIDL

CarAIDL

Car API

Car API

Manager A

Manager B

Java API

AIDL

AIDL

AIDL

AIDL

AIDL

AIDL

AIDL

CTS and VTS test suites are required for Android certification

Provides properties defined and required by Android in Vehicle HIDL spec. and complete set of properties provided by VSS data tree

VSS tree is a base for the android Vehicle HAL property extension.

e.g. Vehicle_Body_Hood_IsOpen
Questions

• For accessing vehicle data, what are the implications of bypassing Android architecture?
  • Impact on CTS, VTS?
  • TREBLE?
  • What if Google keeps adapting the properties.
  • What are the alternatives to VSS Data model and interfaces to access it?
    • How do we address the reuse from existing systems.
  • What is the process to define complex data types to be used in data model?
  • What are the concerns on Data access