# Case Study : Vehicle Data Architecture for Connected Car Services at Hyundai Motor Company

**Hyundai Motor Company**
**Seung-jae Lee ( seungjae.lee@hyundai.com )**

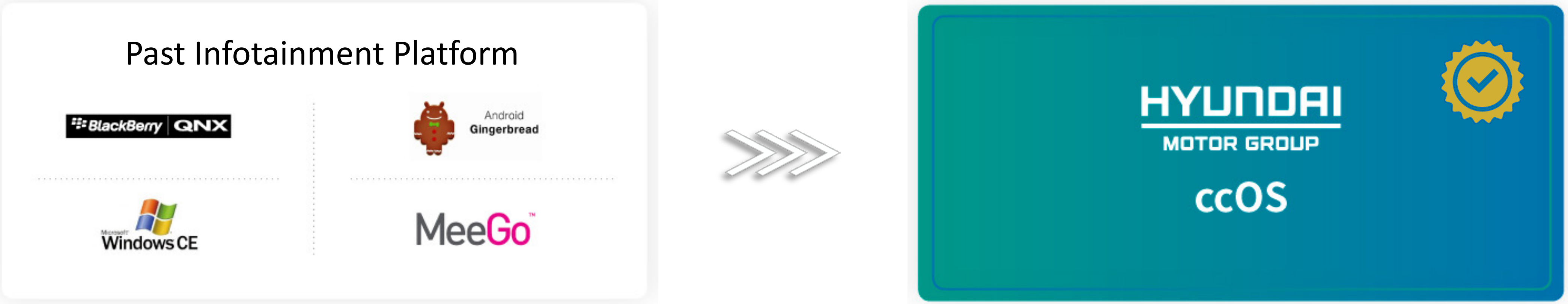
October 2022

# Agenda

1. Introduction
   - ccOS Overview
   - Vehicle Service and ccos::*HVehicle* API
2. ccOS VSM (Vehicle Signal Model)
   - Mission of VSM
   - Key Feature
   - VSM Details
   - VSM Case Study #1 / #2
3. VSM for CCS
   - Introduction to CCS
   - CCS VSM
   - Key Feature
4. Vehicle Data Architecture
5. Q&A

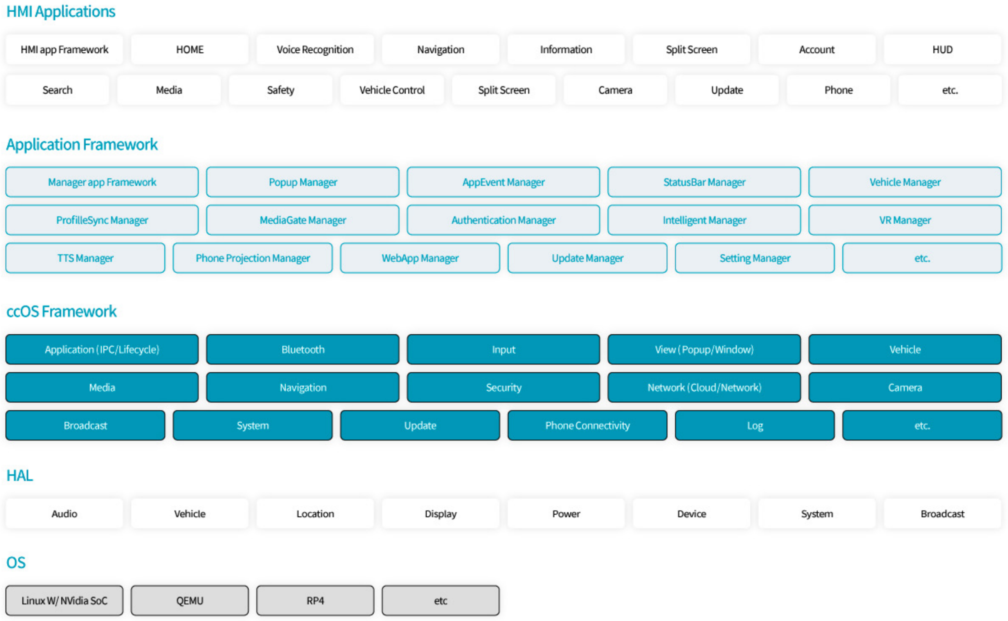
# ccOS Overview

- Connected Car Strategy
  - We producing 8 million vehicles per year, and going to connect all cars by 2025
- What is the ccOS?
  - We started developing *ccOS(Connected Car OS)* in 2016 to build a connected car service ecosystem and prepare for the future SDV environment
  - The ccOS based infotainment system was introduced market through the Genesis G80/GV80


Past Infotainment Platform

# ccOS Overview

- ccOS Architecture Design Attributes
  - Layered Architecture
    - to enhance S/W component reusability and hardware portability
    - support for ARM-based processors and x86-based QEMU environments with HAL layer
  - Performance Considerations
    - C++ native software components
  - Security Enhancement
    - Linux kernel security module
    - Strictly managing network resource



HMI Applications

HMI app Framework | HOME | Voice Recognition | Navigation | Information | Split Screen | Account | HUD

Search | Media | Safety | Vehicle Control | Split Screen | Camera | Update | Phone | etc.

Application Framework

Manager app Framework | Popup Manager | AppEvent Manager | StatusBar Manager | Vehicle Manager

ProfilleSync Manager | MediaGate Manager | Authentication Manager | Intelligent Manager | VR Manager

TTS Manager | Phone Projection Manager | WebApp Manager | Update Manager | Setting Manager | etc.

ccOS Framework

Application (IPC/Lifecycle) | Bluetooth | Input | View ( Popup/Window) | Vehicle

Media | Navigation | Security | Network (Cloud/Network) | Camera

Broadcast | System | Update | Phone Connectivity | Log | etc.

HAL

Audio | Vehicle | Location | Display | Power | Device | System | Broadcast

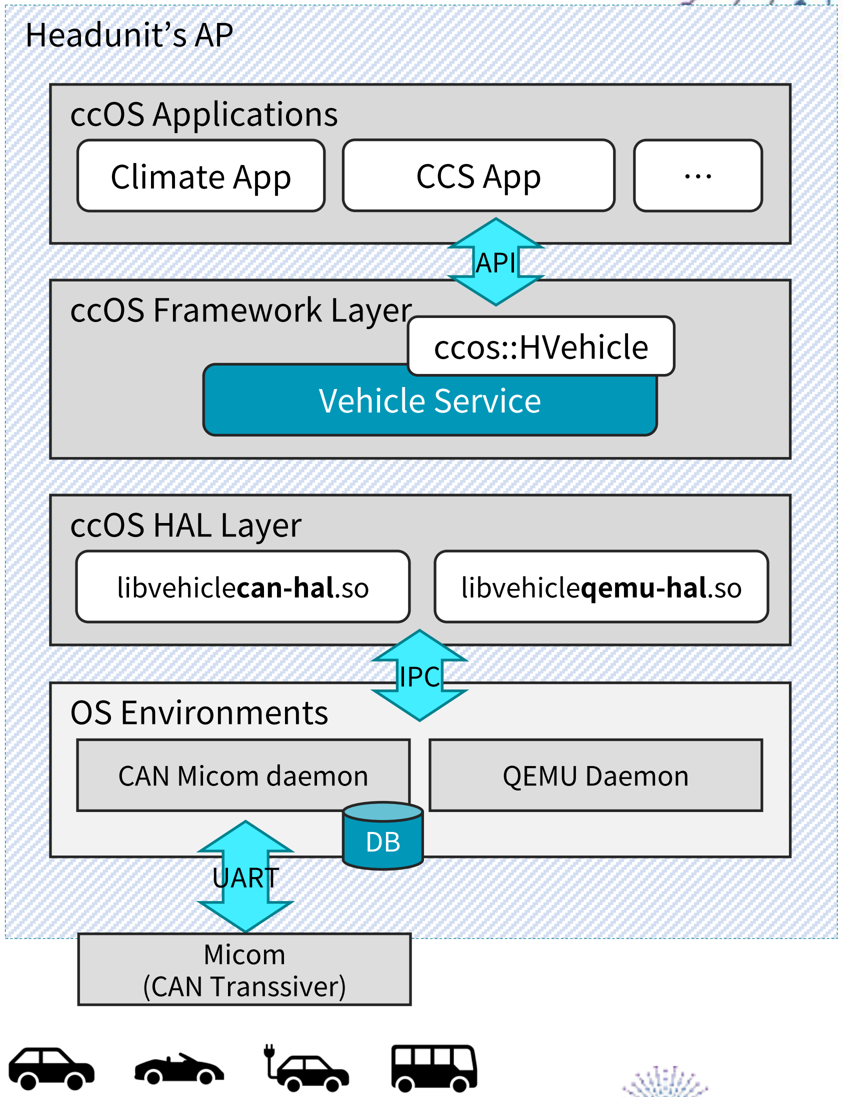
OS

Linux W/ NVidia SoC | QEMU | RP4 | etc

ccOS Component Architecture

# Vehicle Service and ccos::HVehicle API

- Vehicle Service
  - Provides abstracted IVN signals to the ccOS Apps as a generalized methodology
  - It covers that need to be handled by the headunit and signals that need to be interacted with the server



# Vehicle Service and ccos::HVehicle API

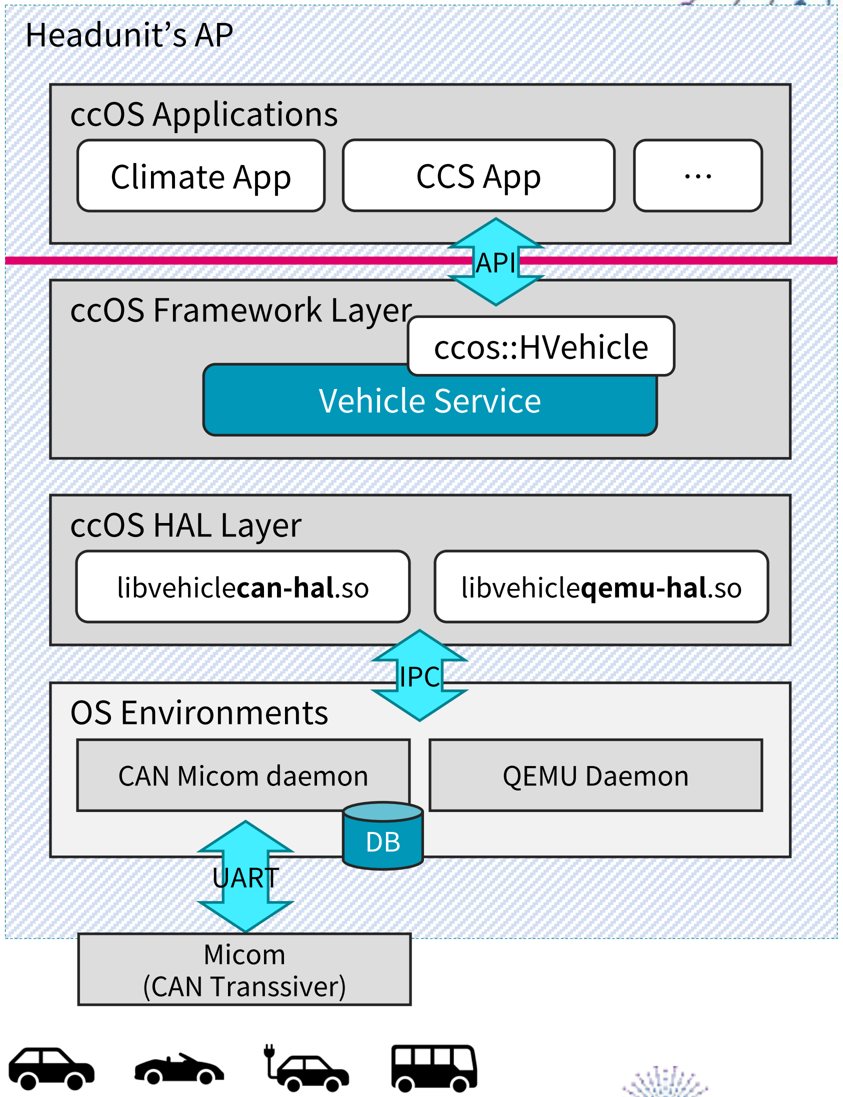
- ccOS::HVehicle API
  - Our legacy vehicle APIs
    - Define all vehicle signals as C++ classes statically…

```
cabin::isDoorOpened(HDoorPosition::FRONT_LEFT, Value);
```

    - From a semantic perspective, all signals on the vehicle are defined as classes
    - Difficult to automate to build a test suite, which requires a lot of effort.
    - Build time dependency
  - New vehicle APIs
    - Separation of API for behavior and signal data model

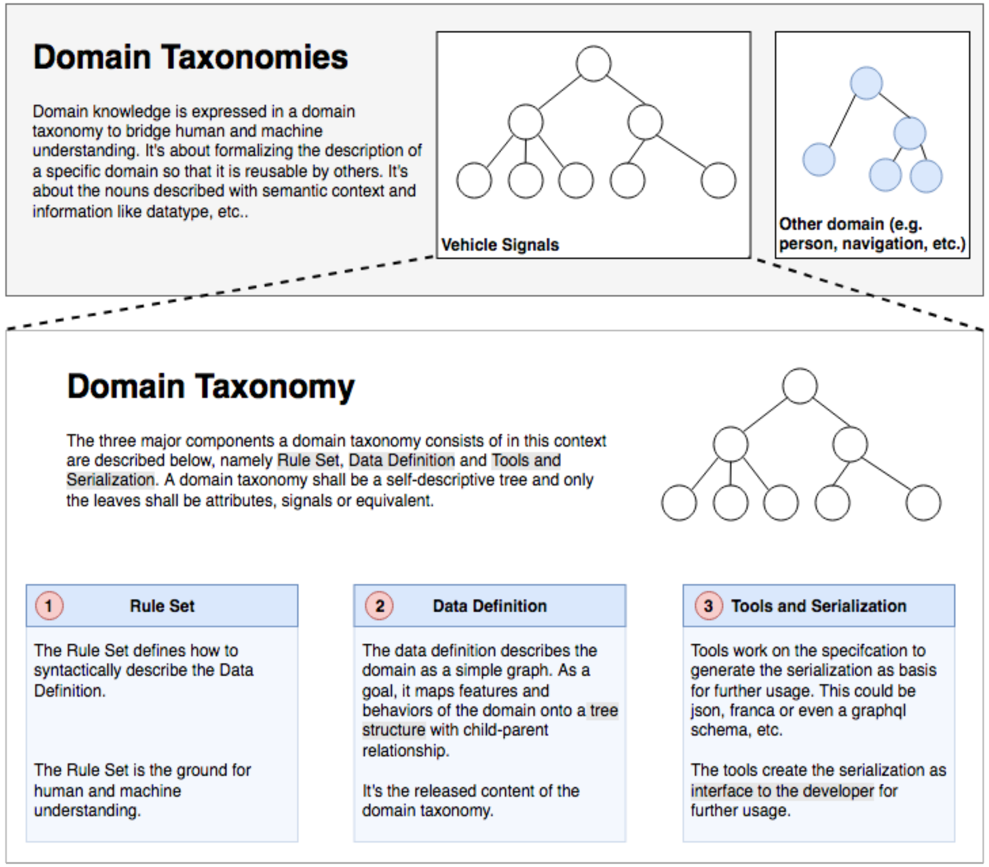
```
getSignal(“Vehicle.Cabin.Door.Row1.Driver.Open”, Value);
```

    - Using the Code Generator by VSM(*.vsm) definition
    - Runtime dependency
    - Easier to create connectivity with servers based on defined data models



# ccOS VSM

- Vehicle Signal Model (VSM)
  - Main mission to standardize vehicle monitoring and control as an interface
  - Same starting points from COVESA Vehicle Signal Specification’s domain taxonomies
  - VSM provides a standard interface for vehicle integration of ccOS App
  - Over 2800+ signals that for vehicle integration have already been defined



## Domain Taxonomies

Domain knowledge is expressed in a domain taxonomy to bridge human and machine understanding. It's about formalizing the description of a specific domain so that it is reusable by others. It's about the nouns described with semantic context and information like datatype, etc..

## Domain Taxonomy

The three major components a domain taxonomy consists of in this context are described below, namely Rule Set, Data Definition and Tools and Serialization. A domain taxonomy shall be a self-descriptive tree and only the leaves shall be attributes, signals or equivalent.

**1 Rule Set**

The Rule Set defines how to syntactically describe the Data Definition.

The Rule Set is the ground for human and machine understanding.

**2 Data Definition**

The data definition describes the domain as a simple graph. As a goal, it maps features and behaviors of the domain onto a tree structure with child-parent relationship.

It's the released content of the domain taxonomy.

**3 Tools and Serialization**

Tools work on the specifcation to generate the serialization as basis for further usage. This could be json, franca or even a graphql schema, etc.

The tools create the serialization as interface to the developer for further usage.

< Reference: Taxonomies::Vehicle Signal Specification (covesa.github.io) >

# ccOS VSM

- VSM Key Feature
  - VSM's Rule Set and Data Definition method follows VSS
    ***e.g., YAML syntax, using root node name with "Vehicle.*"***
  - Rule Set
    - Node Type
      : branch, sensor, actuator, attribute, ***getproperty, setproperty***
    - instance, aggregate concept is not considered
    - Node Name Rules
      : Defined the node name in terms of classification of control and sensor's target (★)
      : Use the same node name with sensor/actuator, getproperty/setproperty
  - Binding IVN signal relationship to VSM Node
    - Regular Relationship Support (1:1 Case)
    - Multiple Relationship Support (N:1 Case, 1:M Case)

| Perspective | Brake | TurnSignal |
|---|---|---|
| by Target (★) | Vehicle.Chassis.Brake | Vehicle.Body.Lights.Rear.Left.TurnSignal |
| by Driver | Vehicle.Cabin.Brake | Vehicle.Cabin.StreeringWheel.TurnSignal |

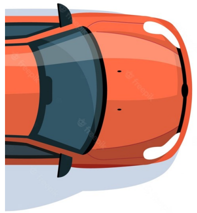
# VSM Details

- IVN Signal - VSM Node Binding Rules
  - Supporting Regular Relationship (1:1 Case)
  - Supporting Multiple Relationship (N:1 Case, 1:M Case)
    - Multi IVN signals combines to One VSM node (N:1 Case)
    - One IVN signal updates to Multiple VSM node (1:M Case)

| Options | Defined Rule | Case Study |
|---|---|---|
| 1:1 Case | IVN and VSM has 1:1 correspondence | For checking the heating status of the handle<br>□ IVN: StrWhlHtrSwSta (Rx)<br>□ VSM: Vehicle.Cabin.SteeringWheel.Heat.State (Sensor) |
| N:1 Case | Can be expressed as one node by combined expression<br>(exclusive condition or If-else) | Sender ECU could be different, but it combine to single VSM<br>□ IVN: StrWhlHtrSwSta (Rx), StrWhlHtrSwSta_v2 (Rx)<br>□ VSM : Vehicle.Cabin.SteeringWheel.Heat.State (Sensor) |
| 1:M Case | One IVN signal is represented in various VSM node, it may be defined as Alias Node. | □ IVN: TrnSigLmpLtBlnkngSta, TrnSigLmpRtBlnkngSta (Rx)<br>□ VSM: Vehicle.Body.Lights.Front.Left.TurnSignal.Blink (Sensor)<br>Vehicle.Body.Lights.Front.Right.TurnSignal.Blink (Sensor)<br>Vehicle.Body.Lights.Rear.Left.TurnSignal.Blink (Sensor)<br>Vehicle.Body.Lights.Rear.Right.TurnSignal.Blink (Sensor) |

# VSM Case Study #1

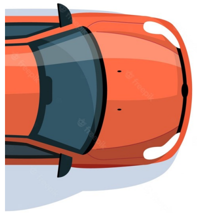
- Case Study : Headlamp system warning and lamp open-circuit warning
  - IVN signals requirement and use-case based scene analysis
    - Each headlamp has an open circuit warning signal or system level warning signal
    - Front headlamp has an up-light, down-light and turn-light on the left and right sides
    - Three different IVN signal types (Type A/B/C)
    - To design a leaf node with Vehicle.Body.Lights.Front.Left. as a parent
    - Need to inform the customer of the detailed trouble shooting when error occurred



Type A (Lamp system)
- lamp state
**- lamp open warning**
- Hi/Low lamp

< Type A >

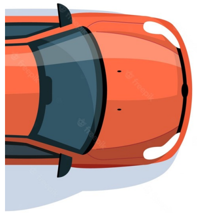
- Vehicle.Body.Lights.**Front.Left**.HighBeam.Warning
- Vehicle.Body.Lights.**Front.Left**.LowBeam.Warning
- Vehicle.Body.Lights.**Front.Left**.HighBeam.LampState
- Vehicle.Body.Lights.**Front.Left**.LowBeam.LampState



Type B (LED system)
- lamp state
**- headlamp circuit warning**
- Hi/Low lamp

< Type B >

- Vehicle.Body.Lights.**Front.Left**.HeadLamp.HighWarning
- Vehicle.Body.Lights.**Front.Left**.HeadLamp.LowWarning
- Vehicle.Body.Lights.**Front.Left**.HighBeam.LampState
- Vehicle.Body.Lights.**Front.Left**.LowBeam.LampState



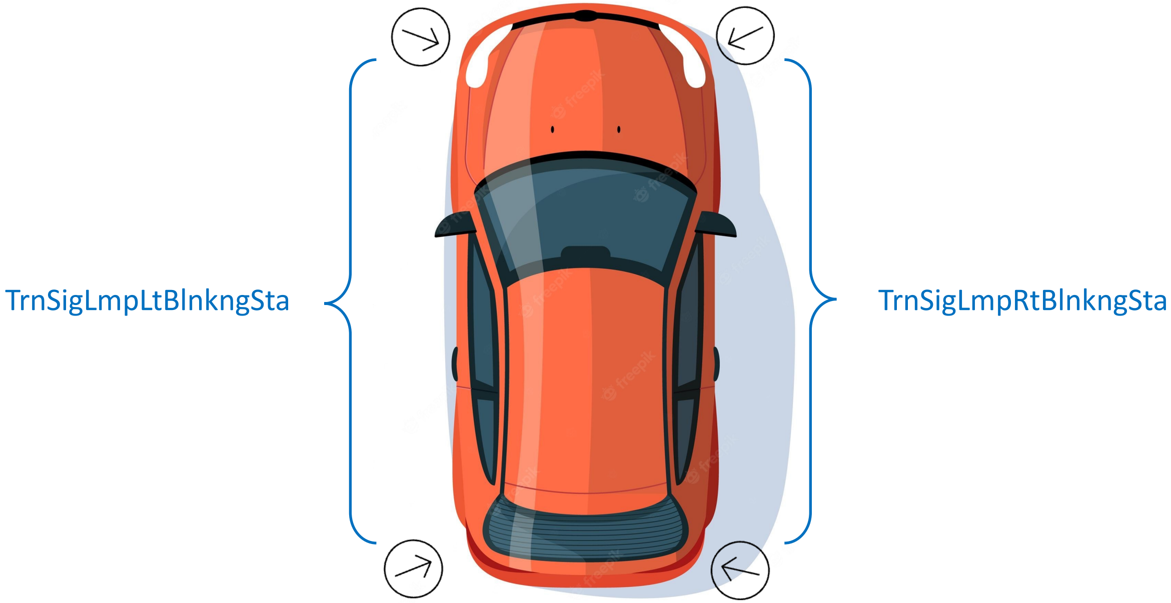
Type C (Bi-lamp system)
- lamp state
**- headlamp system warning**
**- single bi-directional lamp**

< Type C >

- Vehicle.Body.Lights.**Front.Left**.HeadLamp.BiWarning
- Vehicle.Body.Lights.**Front.Left**.HighBeam.LampState
- Vehicle.Body.Lights.**Front.Left**.LowBeam.LampState

# VSM Case Study #2

- Case Study : Turn light signal
  - IVN signals and scene analysis
    - There are 4 turn-light signal lamps on vehicle
    - The left and right blink independently
    - Turn-light signal can be operated from side to side
      ( TrnSigLmpLtBlnkngSta / TrnSigLmpRtBlnkngSta )
    - Signal value is always on while turn-light signal is flashing
    - Hazard lamp blinks 4 turn-light signal together, with individual signal, not sharing turn-light signal
  - Review neighbor VSM node
    - Vehicle.Body.Lights.Hazard.State
    - Vehicle.Body.Lights.**Front.Left**.HighBeam.Warning
    - Vehicle.Body.Lights.**Front.Left**.HighBeam.LampState
    - Vehicle.Body.Lights.**Front.Left**.LowBeam.Warning
    - Vehicle.Body.Lights.**Front.Left**.LowBeam.LampState
    - Vehicle.Body.Lights.**Front.Left**.TurnSignal.Warning
    - Vehicle.Body.Lights.**Front.Left.TurnSignal.Blink** ← ***New Sensor Node***
    - Vehicle.Body.Lights.**Rear.Left.TurnSignal.Blink** ← ***& Alias Node Here!***



< ***Alias Node*** *for* TrnSigLmpLtBlnkngSta >

Vehicle.Body.Lights.**Front.Left**.TurnSignal.Blink
Vehicle.Body.Lights.**Rear.Left**.TurnSignal.Blink

< ***Alias Node*** *for* TrnSigLmpRtBlnkngSta >

Vehicle.Body.Lights.**Front.Right**.TurnSignal.Blink
Vehicle.Body.Lights.**Rear.Right**.TurnSignal.Blink

# Introduction to CCS

- Connected Car Service
  - Our connected car service was launched in 2003 with MOZEN service
  - Support for safety security features, route searching, concierge services, media streaming, etc.
  - Next Generation CCS includes near-real-time vehicle status transmission

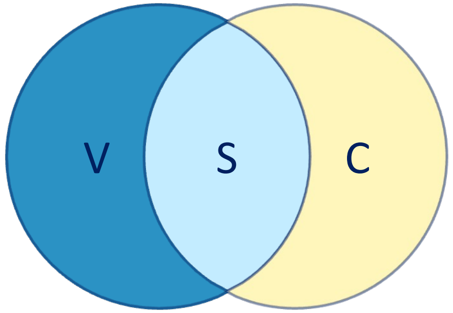
Cloud

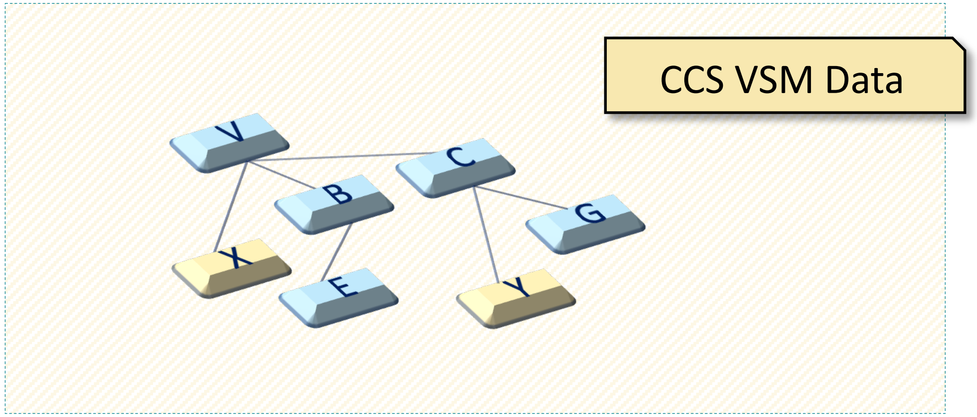
# CCS VSM

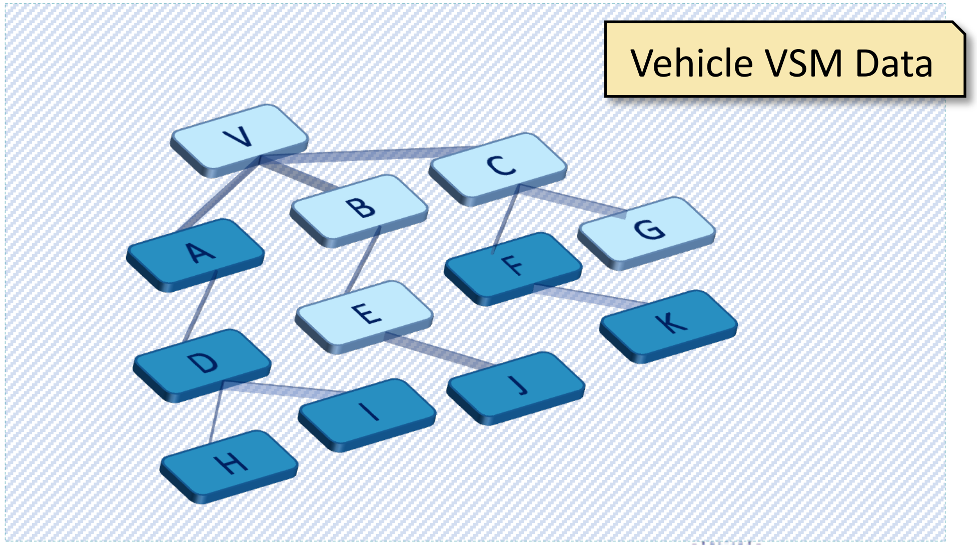
- Main mission to standardize vehicle monitoring and control as an interface
- Extracting data model data commonly used from the perspective of implementing connected car services
- Abstract if the state model is complex, includes sematic abstraction of transmitting value
- CCS VSM provides a standard data model for vehicle integration with connected car
- Over 300+ signals that for vehicle integration have already been defined for Smartphone Application

# CCS VSM

- CCS VSM Key Feature
  - CCS VSM Rule Set and Data Definition is the subset of Vehicle VSM
  - Rule Set
    - Node Type
      : branch, sensor, *Rule based Reporting Policy (In Progress...)*
  - Binding IVN signal relationship to CCS VSM Node
    - Refining the valid value and apply it to CCS VSM node
  - Legend
    - 🟦 : Vehicle VSM Domain Only
    - ⬜ : Shared VSM Node
    - 🟨 : CCS VSM Domain Only







# VSM Case Study #3

- Case Study : Headlamp State ***to VSS?***
  - Conversion from Vehicle VSM node to CCS VSM

< Type A >

- Vehicle.Body.Lights.**Front.Left**.HighBeam.Warning
- Vehicle.Body.Lights.**Front.Left**.LowBeam.Warning
- Vehicle.Body.Lights.**Front.Left**.HighBeam.LampState
- Vehicle.Body.Lights.**Front.Left**.LowBeam.LampState

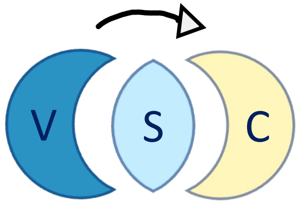
< Type B >

- Vehicle.Body.Lights.**Front.Left**.HeadLamp.HighWarning
- Vehicle.Body.Lights.**Front.Left**.HeadLamp.LowWarning
- Vehicle.Body.Lights.**Front.Left**.HighBeam.LampState
- Vehicle.Body.Lights.**Front.Left**.LowBeam.LampState

< Type C >

- Vehicle.Body.Lights.**Front.Left**.HeadLamp.BiWarning
- Vehicle.Body.Lights.**Front.Left**.HighBeam.LampState
- Vehicle.Body.Lights.**Front.Left**.LowBeam.LampState

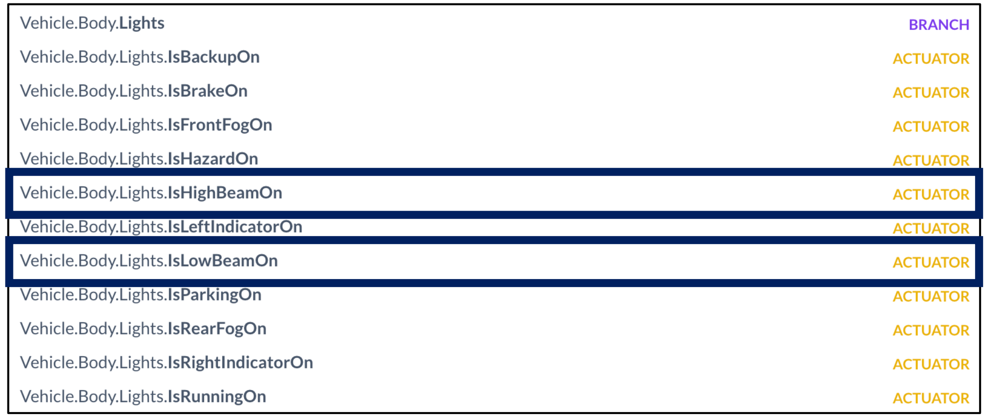
  - VSS Neighboring node



< CCS VSM >

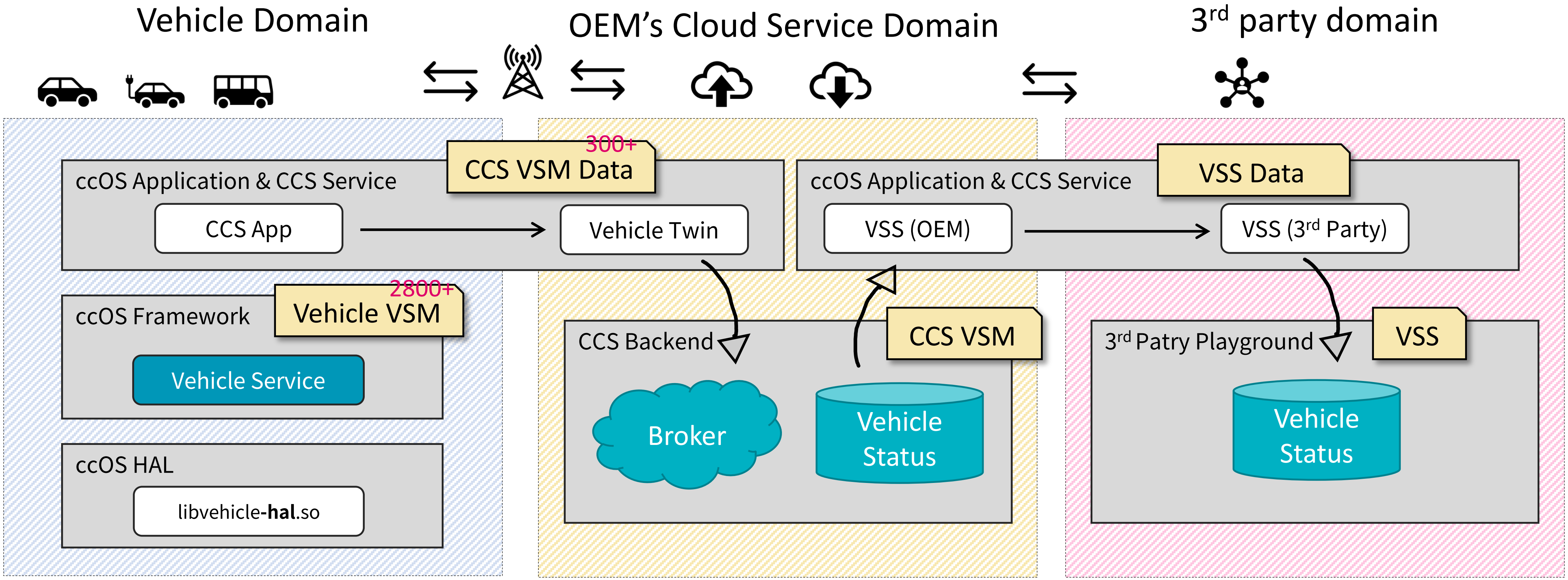
Vehicle.Body.Lights.**Front.Left**.HighBeam.LampState ||
Vehicle.Body.Lights.**Front.Right**.HighBeam.LampState

Vehicle.Body.Lights.**Front.Left**.LowBeam.LampState ||
Vehicle.Body.Lights.**Front.Right**.LowBeam.LampState



| | |
|---|---|
| Vehicle.Body.**Lights** | BRANCH |
| Vehicle.Body.Lights.**IsBackupOn** | ACTUATOR |
| Vehicle.Body.Lights.**IsBrakeOn** | ACTUATOR |
| Vehicle.Body.Lights.**IsFrontFogOn** | ACTUATOR |
| Vehicle.Body.Lights.**IsHazardOn** | ACTUATOR |
| Vehicle.Body.Lights.**IsHighBeamOn** | ACTUATOR |
| Vehicle.Body.Lights.**IsLeftIndicatorOn** | ACTUATOR |
| Vehicle.Body.Lights.**IsLowBeamOn** | ACTUATOR |
| Vehicle.Body.Lights.**IsParkingOn** | ACTUATOR |
| Vehicle.Body.Lights.**IsRearFogOn** | ACTUATOR |
| Vehicle.Body.Lights.**IsRightIndicatorOn** | ACTUATOR |
| Vehicle.Body.Lights.**IsRunningOn** | ACTUATOR |

# HMC’s Vehicle Data Architecture



Vehicle Data Architecture

COVESA

Accelerating the future of connected vehicles

# Thank you :-)

seungjae.lee@hyundai.com