W3C Automotive BG: F2F in Tokyo

GENIVI Vehicle Web API

Justin(JongSeon) Park
LG Electronics

29-May-13

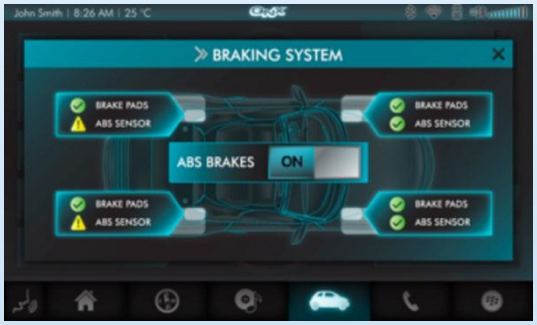
Agenda

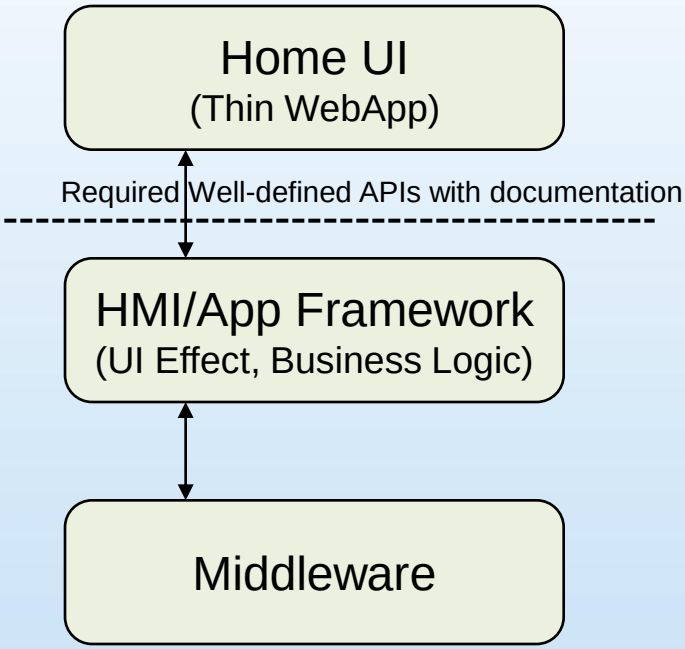
- Use Cases, Characteristics of Vehicle Data
- Selecting Data Types to Standardize
- Considerations on OBD-II
- Introduction of GENIVI Web Vehicle APIs
- Summary of Decision Points
- Q&A

Use Cases for Web API for Vehicle Data

Categorized into three types of WebApps which access vehicle data

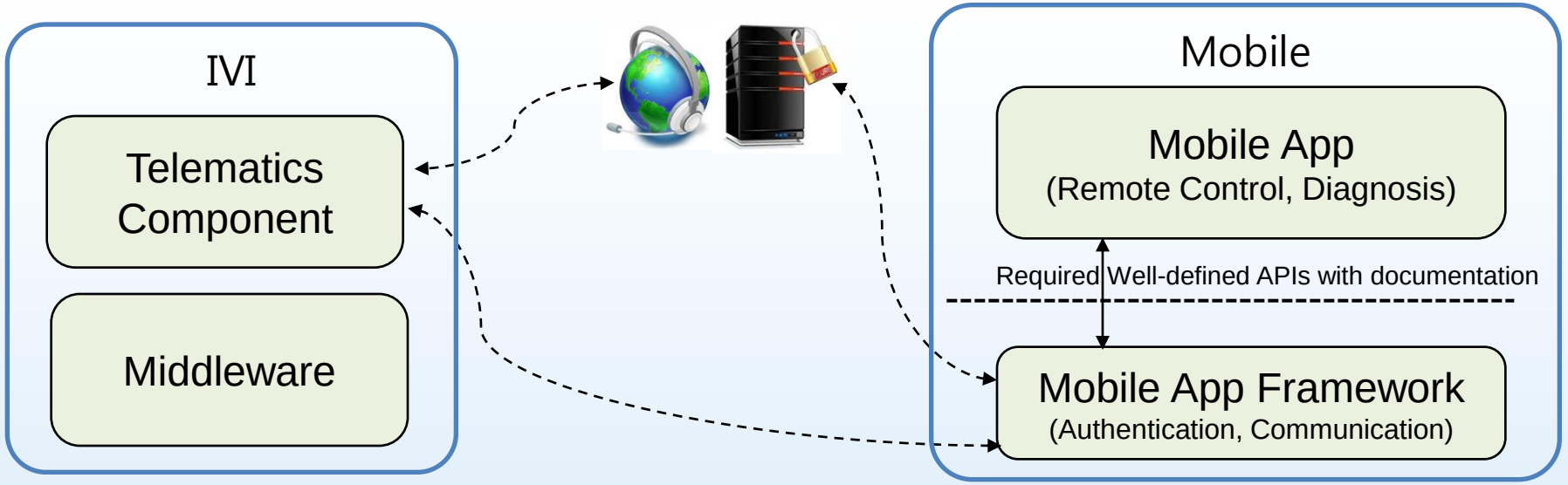
- Home (Main, HMI, Dashboard) - Installed(Build-in), OEM-provided
  - Major module that access various Vehicle Data
  - Includes all built-in functions – Controlling HVAC, Showing Vehicle Status
  - Needs almost all vehicle data for both reading/writing





Use Cases for Web API for Vehicle Data

- Telematics App for mobile phone - Downloadable, OEM-provided

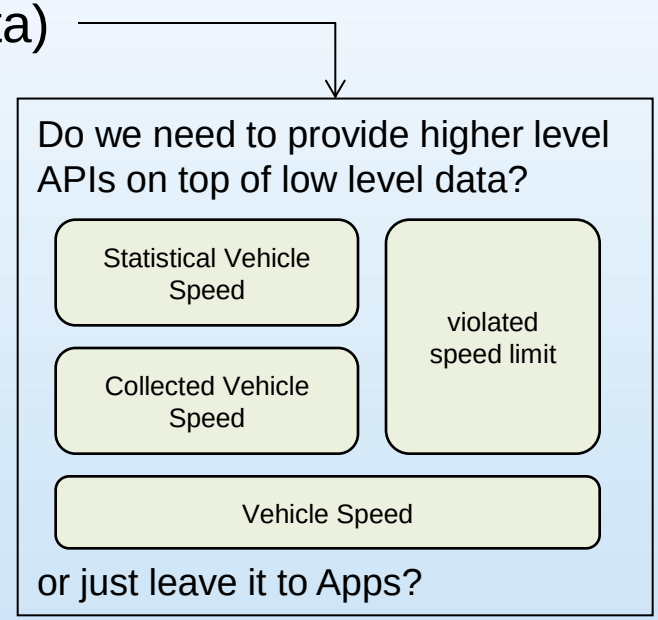


- Market App – Downloadable
  - Most Apps need to know whether vehicle is moving (regulations)
  - Insurance App (Pay-as-you-drive), Any creative Apps in future
  - It's not certain that OEMs will allow Market Apps to access vehicle data
  - Are we needed/able to suggest/predict all possible Apps per each data types?

How to Make Standard Vehicle Web APIs?

We have to understand and consider characteristics of vehicle data

- Data Characteristic
  - So many kinds of vehicle data and data types
  - A few Persistent Data - Car Type, VIN*, Model, WMI**, etc.
  - Most data are Transient; status at a moment
  - Only the latest value is meaningful (except GPS data)
- Vehicle Network Characteristic (usually CAN)
  - Real data exist somewhere else not in IVI
  - Data is broadcasted rather than query
- OEM Variations
  - Unit, Accuracy, Frequency, etc.
  - Policy - Which data are supported, Permissions



VIN* – Vehicle Identification Number (ISO 3779)
WMI** – World Manufacturer Identifier

How to Select Data Types to Support?

Considerations on set the scope of Standardization

- From Use cases
  - Market Apps : only a few types is enough
  - OEM-provided Apps : almost all data is candidates

- Two Approaches
  - Select only common data types through broad consensus
    - Hard to define the scope of common due to the variety of OEM
    - Risk to cover very small percentage of data types needed
    - Still might fail to prevent fragmentation → Only for compatibility of Market Apps?
  - Select all possible data types
    - Required much work
    - But it's easier to subtract than to add
    - Still have an issue that only a part of data types are support depending on models

Considerations on W3C Meeting

What is the reasonable criteria for selecting data to standardize?

- Two Approaches
  - Use case based - Typical way
  - Implementation based - Actually, each OEMs had already defined which data is used for IVI. In fact, the other data is filtered out. So it also makes sense to select all data which is used in IVI

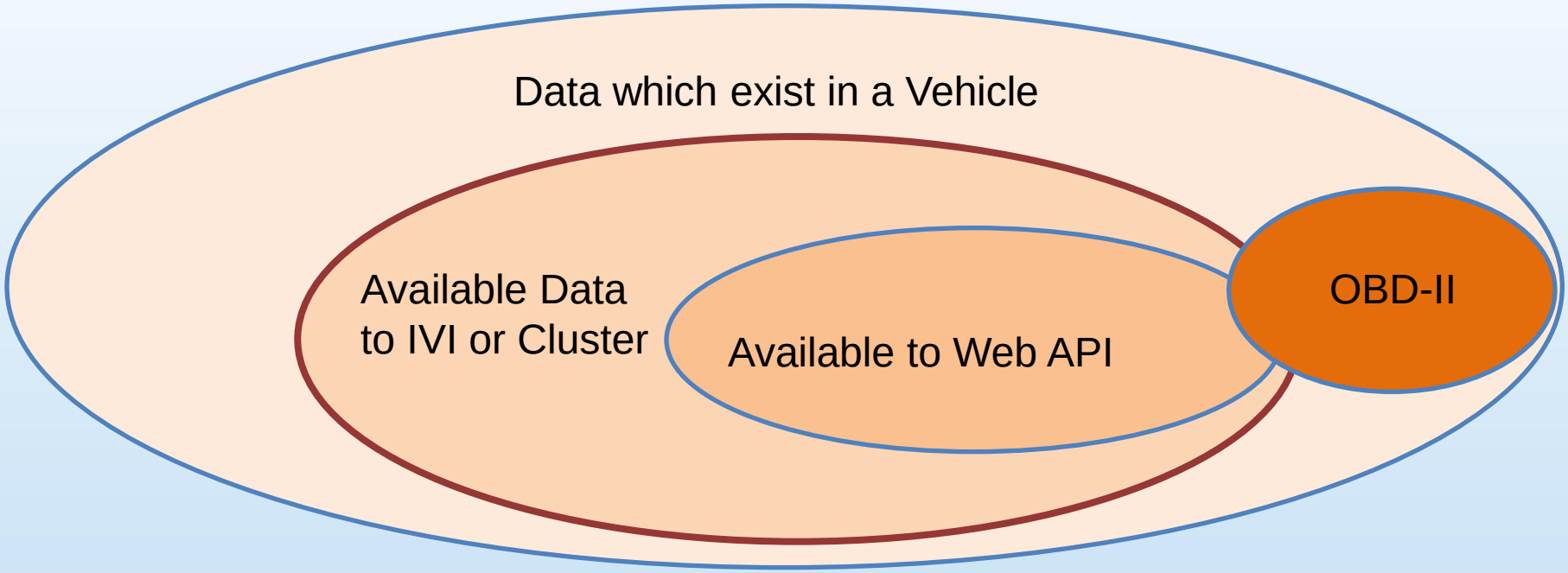
- How about HMI as a use case?
  - It requires almost all data
  - It requires not only get function but also set function
  - No evidence that OEM will use Web-based HMI (concerns regarding performance)

- How about Downloadable App?
  - It's also no evidence that OEM will allow APIs for open market (security concerns)
  - But, it can make the problem simple to limit the boundary
  - Is it possible to find use-cases for various vehicle data? Is it sharable?

Available Vehicle Data According to the Level

What kind of vehicle data do we need to take account?

- ❑ Among whole vehicle data, a part of vehicle data is available to IVI or Cluster
- ❑ All (or a part of) those data will be exported to WebApp
- ➔ We need to focus on data which IVI system actually use



Considerations on the previous W3C Meeting

OBD-II data as a starting point?

- Limited to diagnostic purpose. Too specialized and Focused on engine not IVI
  - OBD-II temperatures : Engine coolant temperature, Intake air temperature, Catalyst Temperature, Engine oil temperature, Engine coolant temperature, Turbocharger temperature, Charge air cooler temperature, Exhaust Gas temperature, Manifold surface temperature, Ambient air temperature

  Vs. GENIVI temperatures : HVAC Fan Target Temperature, interior/exterior temperature
  - OBD-II pressures : Fuel pressure, Intake manifold absolute pressure, Turbocharger compressor inlet pressure, Exhaust pressure

  Vs. GENIVI pressure : Tire Pressure per each tires
  - Missing all the other parts - safety, HVAC, etc.

- GENIVI Web API referred it and adopted valuable data for IVI systems

Data Types According To OBD-II

Among about 180 PIDs in OBD-II, 9 data types is in common

Function	Property	Supported PI				Remarks
		GENIVI Compliance 4.0 (Foton	Tizen IVI 2.0 (AMB) Web API	Webinos (27 Feb 2013)	OBDII	
VIN	vin	O	O		O	
Fuel Type	fuelType	O	O	O	O	
Speedometer	speedometer	O	O	O	O	0-255 km/h
Engine Speed	engineSpeed	O	O		O	rpm
Powertrain Torque		△			O	
Accelerator Pedal Position		△			O	
Throttle Position		△			O	
	Accumulated engine run time	△			O	
	Distance traveled with malfunction indicator lamp (MIL) on				O	
	Distance traveled since codes cleared				O	
	Time run with MIL on				O	minutes
	Time since trouble codes cleared				O	minutes
Fuel	fuel level	O	O		O	
Vehicle time since restart		△			O	
Engine fuel rate					O	L/h
	engineOilTemp	O	O		O	
	engineCoolantTemp	O			O	
Malfunction indicator lamp	malfunctionIndicatorLamp	O			O	
	Auxiliary input status				O	A0 == Power Take Off (PTO) status (1 == active)
Exterior Temperature	exteriorTemp	O	O		O	
Hybrid battery pack remaining life					O	

http://en.wikipedia.org/wiki/OBD-II_PIDs#Standard_PIDs

Data Types According To OBD-II

Intersection of 3 APIs contains 31 types, but only 2 is common with OBD-II

Function	Property	GENIVI Compliance 4.0 (Foton	Tizen IVI 2.0 (AMB) Web API	Webinos (27 Feb 2013)	OBD-II	Remarks
Fuel Type	fuelType	O	O	O	O	
Transmission Gear Type	transmissionGearType	O	O	O		
Speedometer	speedometer	O	O	O	O	0-255 km/h
Transmission Gear Status	transmissionGearStatus	O	O	O		
Lights Status	lightsStatusHead	O	O	O		
	lightsStatusHighBeam	O	O	O		
	lightsStatusTurnRight	O	O	O		
	lightsStatusTurnLeft	O	O	O		
	lightsStatusHazard	O	O	O		
	lightsStatusParking	O	O	O		
Odometer	odometer	O	O	O		
Transmission Oil	transmissionOilLifeLevel	O	O	O		
Tire Pressure	tirePressureFrontLeft	O	O	O		
	tirePressureFrontRight	O	O	O		
	tirePressureRearLeft	O	O	O		
	tirePressureRearRight	O	O	O		
Door Open Status	doorOpenStatusDriver	O	O	O		
	doorOpenStatusPassenger	O	O	O		
	doorOpenStatusRearLeft	O	O	O		
	doorOpenStatusRearRight	O	O	O		
	obstacleDistanceFrontLeft	O	O	O		
	obstacleDistanceFrontRight	O	O	O		
	obstacleDistanceRearLeft	O	O	O		
	obstacleDistanceRearRight	O	O	O		
Windshield Wiper	windshieldWiper	O	O	O		
	hvacFanTargetTemp	O	O	O		
Air-Conditioning	airConditioning	O	O	O		
Window	windowDriver	O	O	O		
	windowPassenger	O	O	O		
	windowRearLeft	O	O	O		
	windowRearRight	O	O	O		

How about AUTOSAR?

How to Overcome OEM Variations?

APIs must be very flexible to absorb variety

- Define as many data types as possible to prevent fragment
  - Need to gather OEM requirements as much as possible
- Allow OEMs much freedom to maintain their policy
  - A few mandatory data types
  - Most of data types need to be optional
- Consider flexibility of interface
  - Minimum number of common methods to support various data types
  - Less structured interfaces to absorb changes depending on OEMs

Considerations on Data Variation

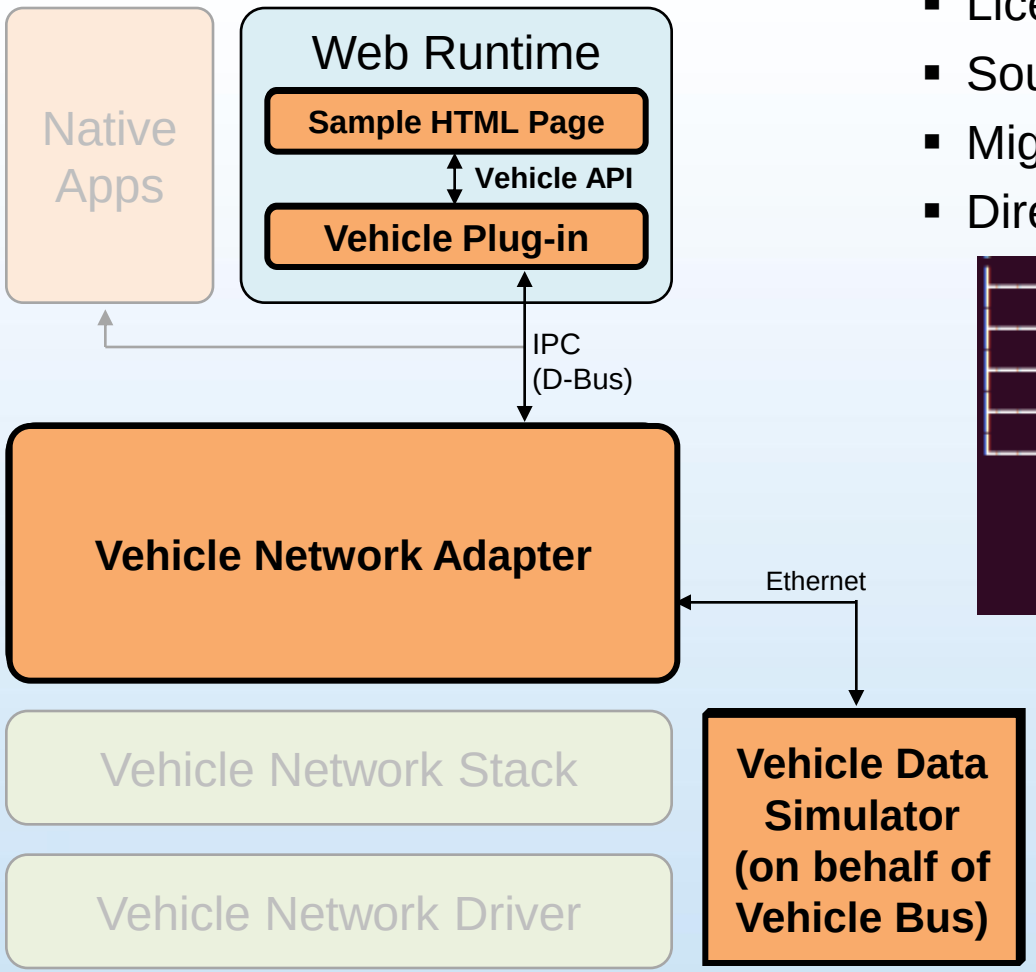
How handle variations?

- Even though a certain data type is selected with a reasonable use-case, some cars may not support it → Variation must be considered!

- Standardize some data as optional
  - How to judge a certain data is common or rare?
  - Problem is many data should be optional in cars
  - Even though W3C defines a certain data as mandatory, there's no guarantee that a certain car will support it.

- In GENIVI and Tizen, 'getSupportedTypes' API is provided.

Composition of GENIVI Reference Implementation



- License is MPL (Mozilla Public License) v2.0
- Source code is available in the GENIVI git
- Migration to OSS is almost done
- Directory Structure

```
├── bin
├── doc
├── html
├── script
└── src
    ├── VehicleDataSimulator
    ├── VehicleNetworkAdapter
    └── plugin
```

GENIVI Reference Implementation (2/3)

How to use it?

- Download
  - Available to the public

$ git clone ssh://git-genivi@git.projects.genivi.org/vehicle-web-api.git

- Build and Install
  - Script files are provided

$ ./script/build-all.sh

- Run
  - Need to execute 3 Apps separately

$ ./bin/VehicleNetworkAdapter &

$ ./bin/VehicleDataSimulator

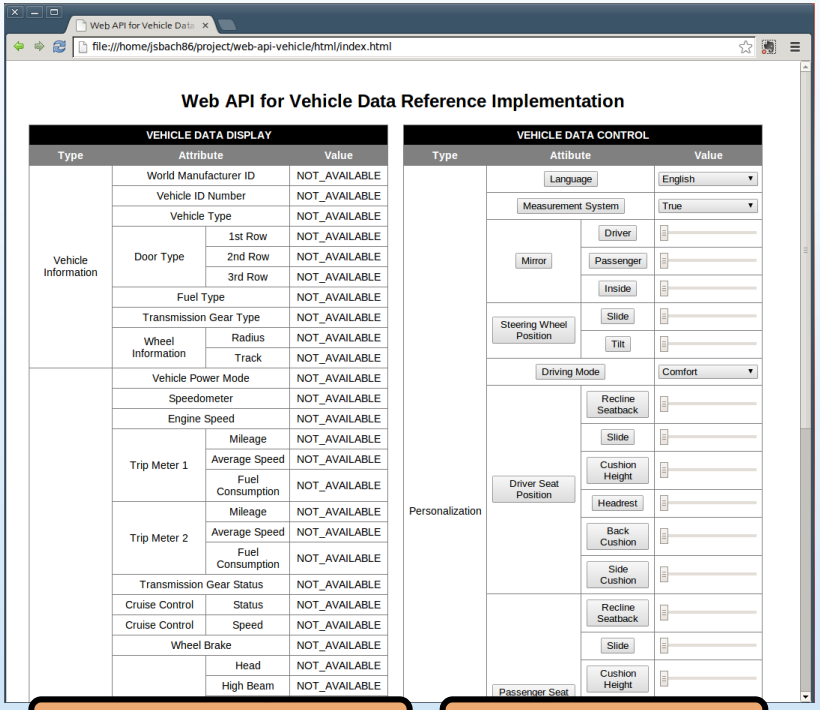
$ google-chrome ./html/index.html (Need to open html on browser)

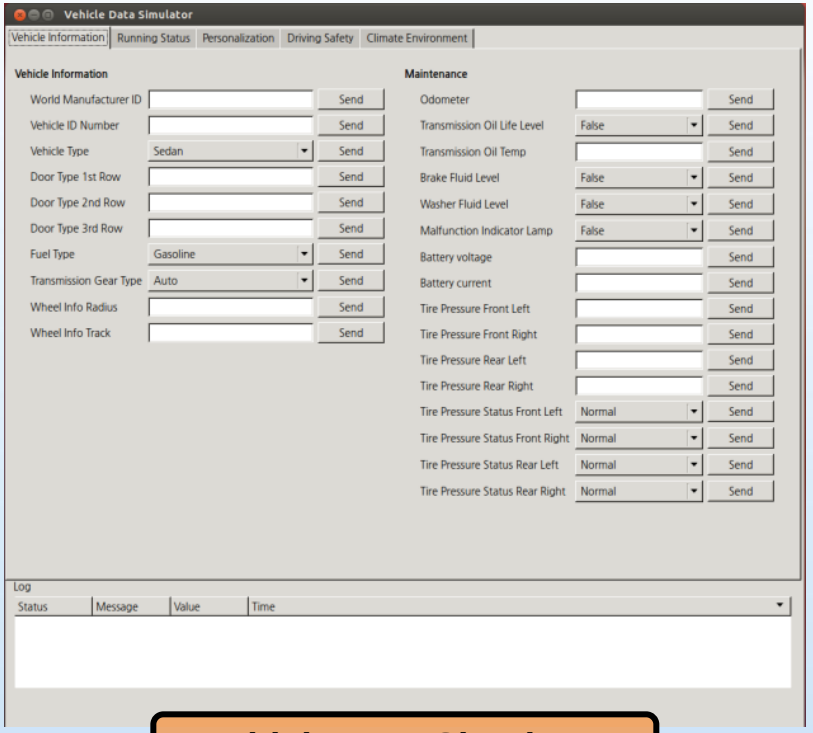
VIN* – Vehicle Identification Number (ISO 3779)
WMI** – World Manufacturer Identifier

GENIVI Reference Implementation (3/3)

Screenshot from run-time

- Made as simple as possible rather than looking nice
  - To help understanding easily from the source code
  - To let developers test a certain feature





Sample HTML Page | **Vehicle Plug-in** | D-Bus | **Vehicle Network Adapter (daemon)** | Ethernet | **Vehicle Data Simulator**

Summary of Decision Points

Things which must be determined to go next step

- Scope of Work
- HMI as a Use-Case?
- OBD-II as starting point?
- Only ‘Get’ or ‘Set’ also?
- How to select data types to be standardized?
- Do we need to define data as 'mandatory' and 'optional’?
- Do we need to provide 'getSupportedTypes' API?
- Anything else

Thank you for your attention

Any Questions?