Vehicle Simulators as example data providers

Genivi Vehicle Simulator and OpenDS

Cloud & Connected Services workshop

GENIVI Virtual Technical Summit | 14 May 2020

Stefan Wysocki
stefan.wysocki@tietoevry.com
Overview

Original purpose of this investigation

• To check if the simulators are applicable as a presentation layer and „real“ data provider for Android Framework and Applications
• Invent some usecases and verify it against the ones agreed in face-to-face workshop
• Highlight the problems or identify blockers that need to be resolved together
• Kickoff VSS Feeder implementation!
Overview

In order to decode the token being sent by the application, app can first verify if JWT was issued by a valid agent and then proceed to decode it.

TPM - trusted platform module

TODO: Franca Service connection ( SomeIP ).
TODO: Check Signal to Service translation in Adaptive Autosar
TODO: Create a layer concept for the Franca to VSS leaf mapping.
TODO: Implement Feeder as a PoC
TODO: Agree on PoC Use Cases Services
TODO: Create PoC SomeIP simulation component to playback some agreed use cases
TODO: Select and implemented VSS data storage inside the Feeder
TODO: APP access for in-vehicle data (e.g. App manifest permissions layer concept )
TODO: APP authorization to access the Data Server (e.g. JWT)
TODO: Finalize the permission layer concept.
TODO: App request/response serialization
TODO: Resolve requested date for the APP from the VSS data structure.
TODO: Change/write data values for requested data leafs
TODO: Handle the subscription for APPs.
TODO: Write a generator which will handle permissions in the data server.
TODO: Implementing the APP permissions based on permission defined/proposed in VSS layers.
TODO: Implement access token request
TODO: Generate access token for the APP including the APP permissions
TODO: App request/response serialization for the Client (JSON to Java)
TODO: Implement selected Use Case
TODO: Request APP permissions from Package Manager

Demo running on:
1. Platform: Renesas --> AOSP P
   ⦁ Android App --> App layer
   ⦁ Android Apollo Plugin
2. Authentication Service --> Framework layer
   ⦁ Java Implementation
   1. NodeJS
   2. Apollo GraphQL Server (Data Server)
     ⦁ Schema generated out of VSS
     ⦁ Permissions generated out of VSS
     ⦁ JWT authentification.
     ⦁ implement Resolvers
     ⦁ implement Mutations
     ⦁ implement Subscriptions
3. Feeder (TBD)

Use Cases:
1. Battery status (high voltage).
2. Tire pressure
3. Air Conditioning
OpenDS
OpenDS

- [https://opends.dfki.de/](https://opends.dfki.de/)
- Configurable by set of XML
- Written in java (with native parts, sometimes closed-source), however did not try to compile it locally
- Officially 5.0 sourcecode distributed with executables in release bundle
- Unofficially multiple forks hosted on Github under the name openDS-oss (4.5)
- Based on opensource and free libraries
OpenDS

- Able to provide data through tcp/ip sockets:
  - Speed (km/h)
  - Engine state (on/off)
  - Engine rpm
  - Actual gear
  - Steering wheel angle
  - Gas/brake pedal state
  - Fuel consumption (l/100km)
  - Fuel - max amount
  - Fuel - current amount
  - Headlights (high/low/off)
  - GPS position (lon, lat, alt, orientation)
OpenDS

• To enable this data to be accessible configuration xml needs to be modified for:

```xml
<settingsControllerServer>
  <startServer>true</startServer>
  <port>5678</port>
</settingsControllerServer>
```
OpenDS

Workflow:

• Connect to socket
• Send xml message to establish connection
• Server starts sending the properties
DEMO
Genivi Vehicle Simulator
Genivi Vehicle Simulator

• https://github.com/GENIVI/genivi-vehicle-simulator

• Use Unity as an engine

• Written in C#, developed in Unity IDE

• Sends values as a map, every frame includes property name bound with value and timestamp
EMSSetSpeed, 0.0000, 33.99587
EngineSpeed, 3740.9430, 33.99587
GearPosActual, 4.00, 33.99587
GearPosTarget, 4.00, 33.99587
AcceleratorPedalPos, 0.0000, 33.99587
DeceleratorPedalPos, 0.0000, 33.99587
RollRate, 0.0000, 33.99587
SteeringWheelAngle, 0.0000, 33.99587
VehicleSpeed, 95.1472, 33.99587
VehicleSpeedOverGnd, 95.1472, 33.99587
WheelSpeedFrL, 44872.8800, 33.99587
WheelSpeedFrR, 45907.8800, 33.99587
WheelSpeedReL, 45003.8100, 33.99587
WheelSpeedReR, 45866.3700, 33.99587
YawRate, 0.0000, 33.99587
## Genivi Vehicle Simulator

### Properties comparison

<table>
<thead>
<tr>
<th>GVS Exclusive</th>
<th>Common</th>
<th>OpenDS Exclusive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target gear</td>
<td>Speed (km/h)</td>
<td>Engine state (on/off)</td>
</tr>
<tr>
<td>Wheel speed</td>
<td>Engine rpm</td>
<td>Fuel consumption (l/100km)</td>
</tr>
<tr>
<td></td>
<td>Actual gear</td>
<td>Fuel - max amount</td>
</tr>
<tr>
<td></td>
<td>Steering wheel angle</td>
<td>Fuel - current amount</td>
</tr>
<tr>
<td></td>
<td>Gas/brake pedal state</td>
<td>Headlights (high/low/off)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GPS position (lon, lat, alt, orientation)</td>
</tr>
</tbody>
</table>

- **Surface:** OpenDS Exclusive
- **Surface:** Common
- **Surface:** GVS Exclusive

**Public**
Reference and contacts

Weekly telcos of AASIG Vehicle Api subgroup

- Tuesdays – 17:00 CET (US friendly time) – Vehicle Data APIs / VHAL

AASIG Mailing list


Reference VSS feeder implementation

- [https://github.com/genivi/vss-feeder](https://github.com/genivi/vss-feeder)

Whole demo setup including GraphQL schema from VSS + GrapQL server

(work in progress, README.md to be updated for docker-compose)
