In-vehicle access to standardized VSS Vehicle Signals

FOSDEM 2023
Sebastian Schildt
Is an OSS software project

Is an Eclipse Working group comprised of several automotive centered OSS software projects

**SDV promises**

- Faster updates
- More “app” less “firmware”
- 10x faster development
- Reusable software
- “Cloud native” in vehicles
- Happy Users, happy devs, happy corporations, blissful happiness for all

Software Defined Vehicle Mindset

Latest and greatest hype in the automotive industry
The SDV approach
What did we achieve?

The good
- I can easily deploy Wordpress in my vehicle
- Probably runs Doom

The bad
- Probably should get security and safety right

The ugly
- Without any access to a vehicle’s hardware, deploying Wordpress and running Doom is likely all I can do
Access to Vehicle Hardware

We have **sensors** (what is our current speed?) and **actuators** (e.g. open the trunk!)

- Accessible over Vehicle busses (e.g. CAN, Ethernet), originating in some embedded, often safety critical ECUs (µCs, AUTOSAR, …)
  - Challenge: Accessing those systems directly from our fancy IT stacks would be insane for safety reasons alone
- How to represent a Vehicle Speed (serialisation, identifiers, units) is not standardised. Varies from OEM to OEM, from model to model, model year, variant
  - Challenge: Semantics of Signals very much not standardised. Similar things are not represented in the same way
Challenge: No standardized signals
Solution: COVESAs Vehicle Signal Specification (VSS)

- A simple, flexible and protocol agnostic common approach for describing vehicle data
- Extensible data model & catalog with industry supported tooling.
- Enables improved interoperability and integration, saving time and cost.
Question: Where to best leverage VSS?

Deeply-Embedded Layer
- Small µCs
- CAN/LIN
- Very proprietary
- Very limited compute resources

Not a happy place for VSS

Backend
- Aggregating data of many vehicles
- Link data to other domains

Good for VSS. Systems already in production
Answer: Convert in a Vehicle Computer*

- Here you can afford the costs of abstraction
- This is the place, where the industry is working on decoupling hard- from software (SDV!)
- Here you save money & effort with more generic/portable software (SDV!)

* Something with a processor and a full blown (POSIX) OS
KUKSA.val Scope and Design Choices

- 100% Open Source Eclipse Project (Apache 2.0 license)
- “In-vehicle digital twin” based on VSS
- Lightweight (core written in RUST)
- Only providing “current” view (no historic data)
- Easy to use language-agnostic interface (GRPC)
- VSS Providers/Feeders to transform data to VSS
Sensors & Actuators in KUKSA.val

but we do want to be able to “actuate” simple things.

Subscribe

Set target:

KUKSA.val databroker

Subscribe target

Set

VSS Provider
Trunk feeder

VSS Provider
Trunk control service

E/E network (CAN, SOME/IP, LIN, DDS, etc.) or Autosar (adaptive) platform
(How)does this work?

Is this written in Powerpoint, or what?
VSS Provider: Trunk feeder

```python
def main():
    with VSSClient('127.0.0.1', 55555) as client:
        isOpen = exists("/tmp/trunkopen")  # access vehicle (bus) systems
        client.set_current_values({
            'Vehicle.Body.Trunk.Rear.IsOpen': Datapoint(isOpen),
        })
        print(f"Trunk feeder: trunk open {isOpen}")

if __name__ == '__main__':
    main()
```

If we detect the trunk is currently open,
Update the current value in KUKSA.val
VSS Provider: Trunk control service

If KUKSA.val pushes a new target state,

Interact with vehicle systems to let it happen

```python
from kuksa_client.grpc import VSSClient
from os import remove, exists

with VSSClient('127.0.0.1', 55555) as client:
        if desired_state == True:
            print(f"Trunk control service: OPEN SESAME!")
            with open('/tmp/trunkopen', 'w'): pass
        else:
            print(f"Trunk control service: CLOSING!")
            if exists("/tmp/trunkopen"): remove('/tmp/trunkopen')
```
Demo

https://youtu.be/fD6My8za4jY
Safety and Security Control points

Where Safety and Security come into play depends largely on your application.

It is safe to assume, they will play a role.

KUKSA.val architecture and separation of concerns gives you several control points.
Enabling SDV

**Challenge:** Letting any application access lower level vehicle systems is **insane.**
- ✓ KUKSA.val gives you a control point
- ✓ Architecture allows integration of safety controls on different levels depending on your requirements

**Challenge:** Semantics of Signals very much not standardised. Representation of similar signals in different vehicles are **not the same.**
- ✓ KUKSA.val leverages standard COVESA VSS signals enabling portable applications
## Stay in contact

<table>
<thead>
<tr>
<th></th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>Github</td>
<td><a href="https://github.com/eclipse/kuksa.val">https://github.com/eclipse/kuksa.val</a></td>
</tr>
<tr>
<td>Me</td>
<td><a href="http://sdv.expert">http://sdv.expert</a></td>
</tr>
<tr>
<td>Eclipse SDV</td>
<td><a href="https://sdv.eclipse.org">https://sdv.eclipse.org</a></td>
</tr>
<tr>
<td>COVESAS VSS</td>
<td><a href="https://covesa.github.io/vehicle_signal_specification/">https://covesa.github.io/vehicle_signal_specification/</a></td>
</tr>
<tr>
<td>Eclipse Velocitas</td>
<td><a href="https://websites.eclipseprojects.io/velocitas/">https://websites.eclipseprojects.io/velocitas/</a></td>
</tr>
</tbody>
</table>

---

**Thank you**