

VSS – A deeper Dive

How to build things with VSS - An example

Sebastian Schildt, Erik Jägervall, Adnan Bekan
COVESA AMM, April 17th 2024



COVESA

Accelerating the future of connected vehicles

NOTE

NOTE

There are some command lines in this presentation. It seems when copying them, sometimes some whitespaces are converted not correctly so you get weird error messages like missing arguments. Somewhere "near" this presentation you should have found a `playbook.txt` file that contains safely copiable commands.

In the past...

... or earlier at this AMM



YOU

Where to use VSS?

What of VSS do I need to support?

What is VSS even?

Which software can I use?

I am confused



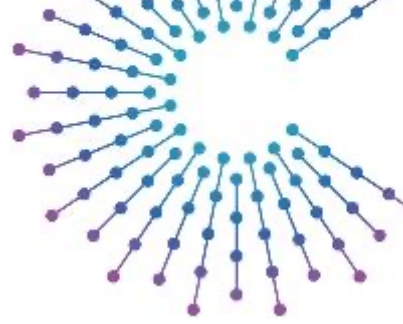
Wherever you want to!

Whatever you want to!

Amazing. It is not an API, Serialization or protocol though.

Many!

JOIN COVESA



This changes now

We show you an **example^{1,2}** of how to use VSS

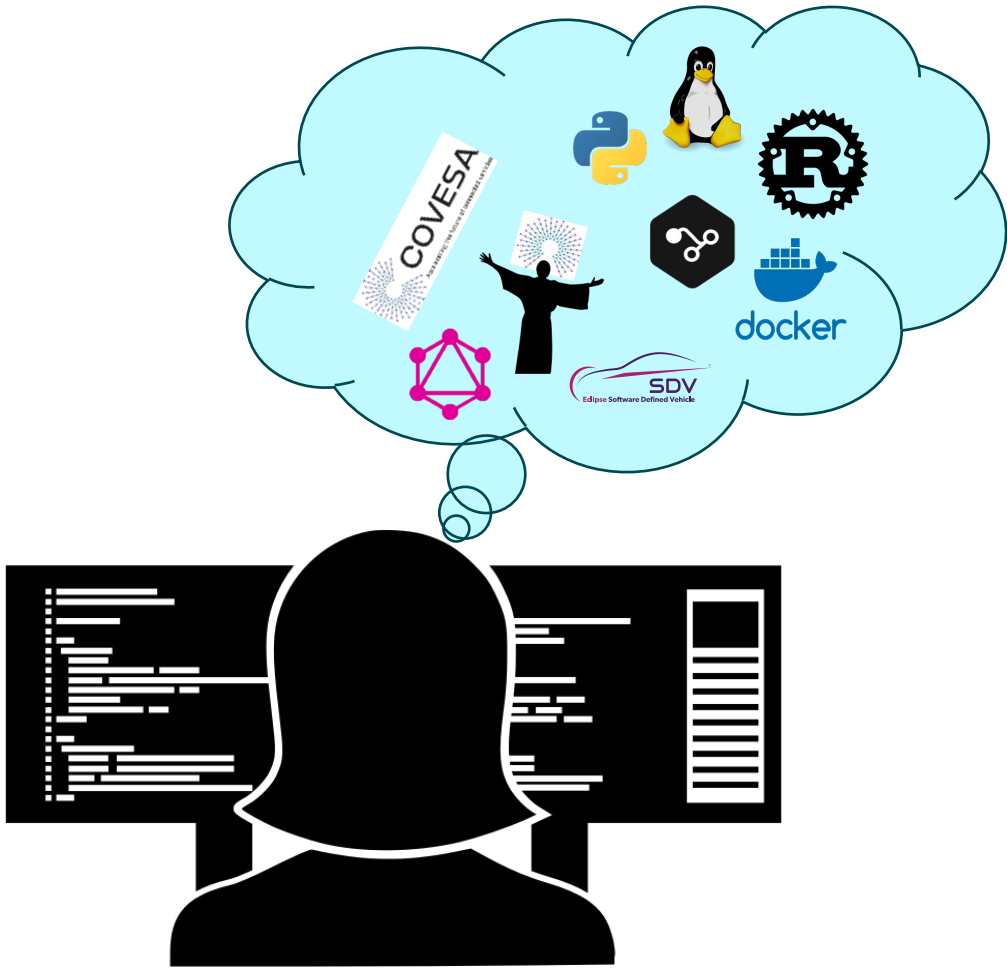
¹ We will...

- look at and modify VSS files
- apply the VSS overlay concept
- use VSS tools
- use real CAN Data
- write real code interacting with VSS

² We will not...

- look at every tool or framework supporting VSS
- show all domains (not too much cloud in here)

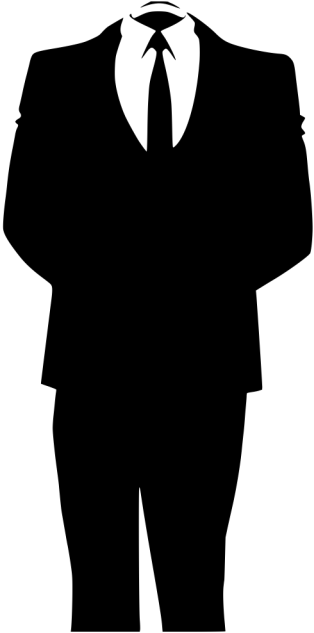
The idea



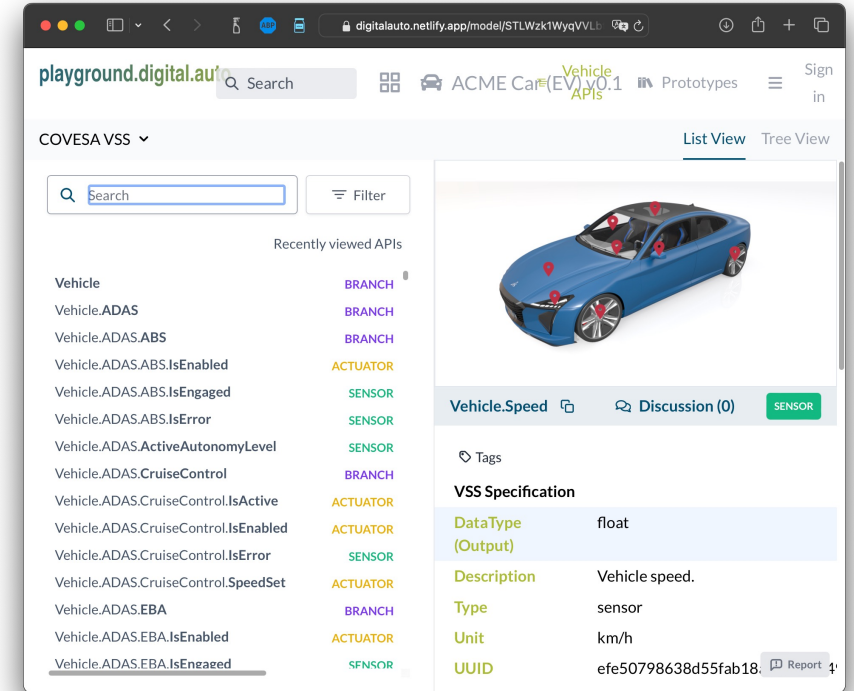
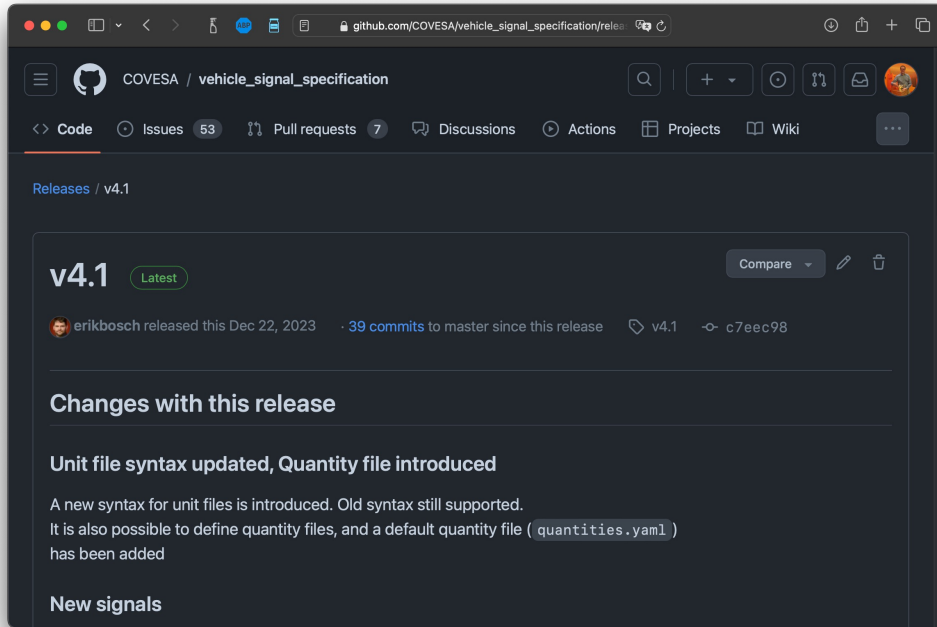
Vehicle Speed + Brake disc temperature =

A yellow speech bubble with a dark border. It contains a brown money bag with a dollar sign, five yellow stars, a 'CAR OF THE YEAR' award logo, and a 'JAPAN CAR OF THE YEAR' award logo.

A red speech bubble with a dark border. It contains a yellow stopwatch icon and a blue dollar sign with a red diagonal slash through it, indicating cost reduction.



VSS – We use VSS already, are we covered?



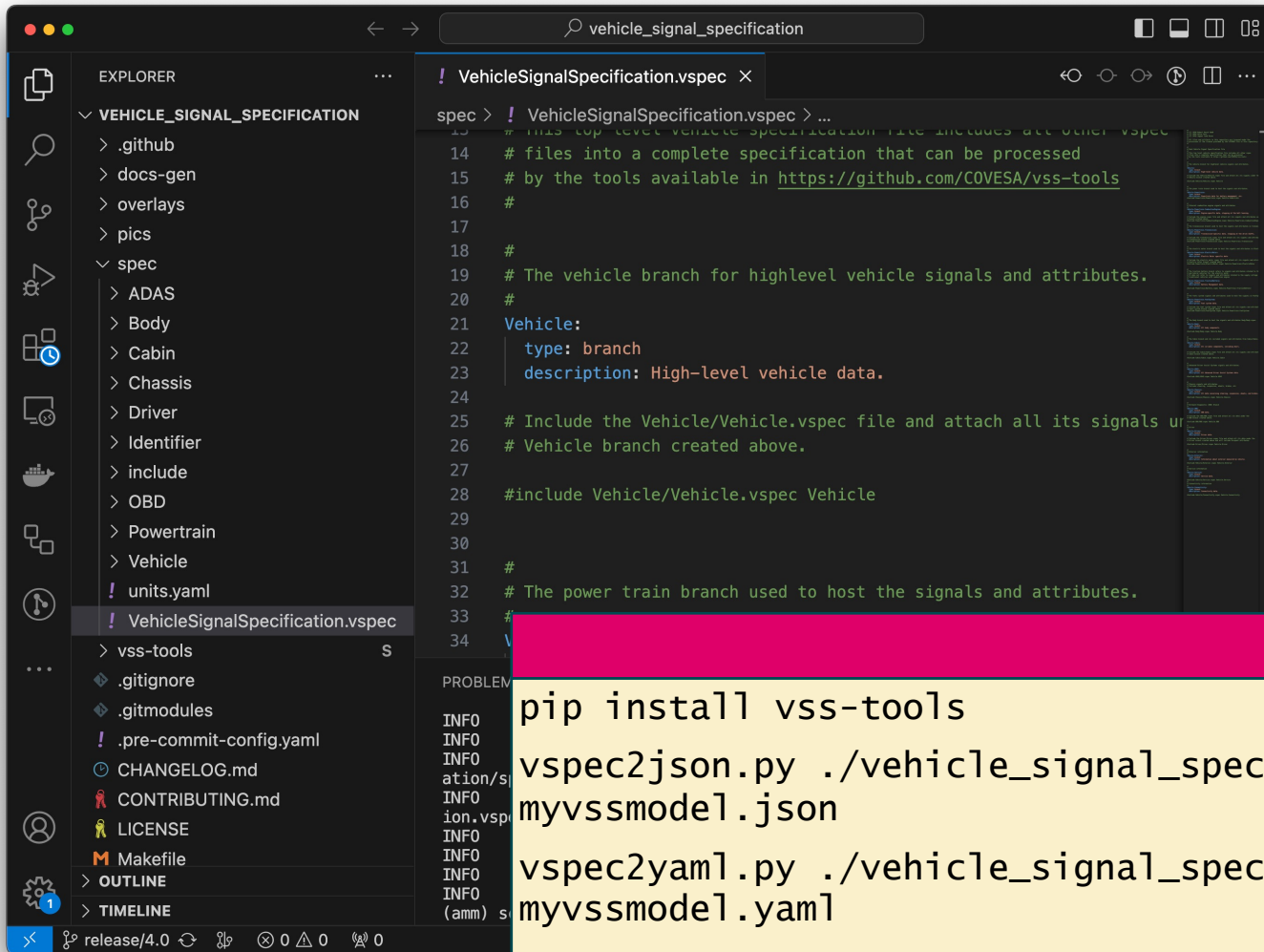
Vehicle.Speed

Vehicle.Chassis.Axle.Rowx.Wheel.L/R.Brake. ??? No temperatures

https://github.com/COVESA/vehicle_signal_specification/releases

<https://digitalauto.netlify.app>

Convert VSS model



Good to know

- COVESA vss-tools convert .vspec files to a variety of other formats (json, cs, ddsidl, etc.) for use in different techs tack
- You can easily write your own converters

Helpful commands

```
pip install vss-tools
```

```
vspec2json.py ./vehicle_signal_specification/spec/vehicleSignalSpecification.vspec  
myvssmodel.json
```

```
vspec2yaml.py ./vehicle_signal_specification/spec/vehicleSignalSpecification.vspec  
myvssmodel.yaml
```

Add additional signal in overlay

```
! VehicleSignalSpecification.vspec      ! dbc_overlay_simple.vspec X
> amm > ! dbc_overlay_simple.vspec > {} Vehicle.Chassis.Axle.Row2.Wheel.Right.Brake.Te
39 # SG_BrakeTempRR3FE : 30|10@1+ (1,-40) [0|0] "C" Receiver
40 Vehicle.Chassis.Axle.Row1.Wheel.Left.Brake.Temperature:
41   datatype: float
42   type: sensor
43   description: Brake Temperature FL
44
45 Vehicle.Chassis.Axle.Row1.Wheel.Right.Brake.Temperature:
46   datatype: float
47   type: sensor
48   description: Brake Temperature FR
49
50 Vehicle.Chassis.Axle.Row2.Wheel.Left.Brake.Temperature:
51   datatype: float
52   type: sensor
53   description: Brake T
54
55 Vehicle.Chassis.Axle.R
56   datatype: float
57   type: sensor
58   ✨description: Brake T
59
```

Good to know

- You can *add* signals to any VSS model using an overlay
- An overlay resides in separate files, so this a plus managing (composable) VSS models in your development workflow

Helpful commands

```
vspec2json.py -o overlay_simple.vspec ./vss_rel_4.1.yaml myextvss.json
```


Use VSS in-vehicle with KUKSA

```
amm — sebastian@docker: ~ — pipenv shell • docker — 104x32
  kuksa-client CLI
  0.4.2

Default tokens directory: /kuksa-client/_internal/kuksa_client/kuksa_server_certificates/jwt

Connecting to VSS server at 127.0.0.1 port 55555 using KUKSA GRPC protocol.
TLS will not be used.
INFO 2024-04-05 09:33:53,241 kuksa_client.grpc No Root CA present, it will not be possible to use a secure connection!
INFO 2024-04-05 09:33:53,241 kuksa_client.grpc.aio Establishing insecure channel
gRPC channel connected.
[Test Client> setValue Vehicle.Speed 42
OK

[Test Client> getValue Vehicle.Speed
{
  "path": "Vehicle.Speed",
  "value": {
    "value": 42.0,
    "timestamp": "2024-04-05T09:33:53.241Z"
  }
}

Test Client>
```

Good to know

- KUKSA is an in-vehicle server for VSS signals
- It is an Eclipse SDV Open Source project under Apache-2.0 license (commercial support available)

Helpful commands

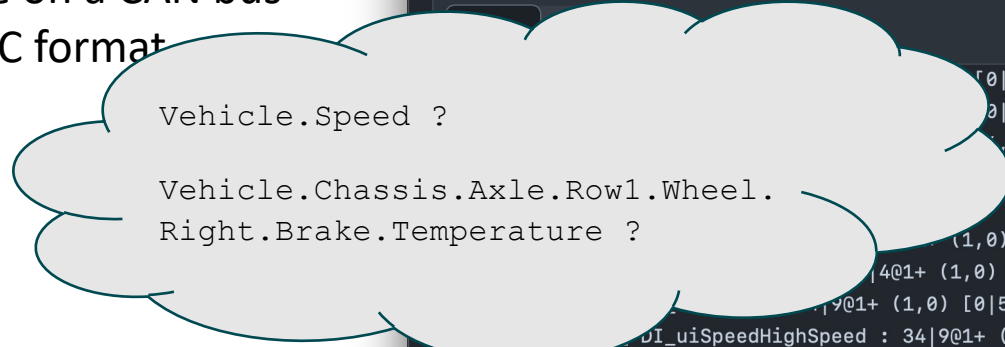
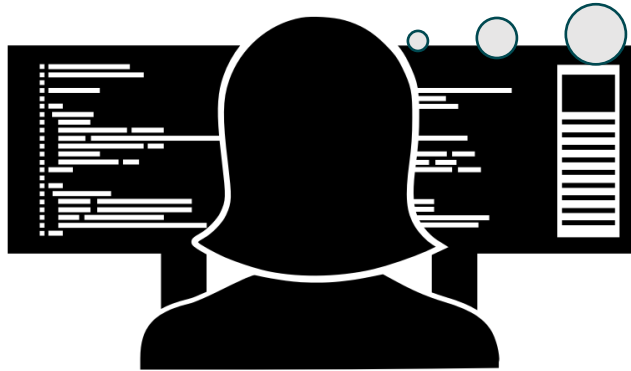
```
docker run -it --rm --net=host ghcr.io/eclipse-kuksa/kuksa-databroker:latest
docker run -it --rm --net=host ghcr.io/eclipse-kuksa/kuksa-python-sdk/kuksa-client:latest
docker run -it --rm --net=host -v $(pwd):/data ghcr.io/eclipse-kuksa/kuksa-databroker:latest --vss /data/myvssmodel.json
```

CAN – Where is my data in a vehicle?

The raw signals are not available in VSS

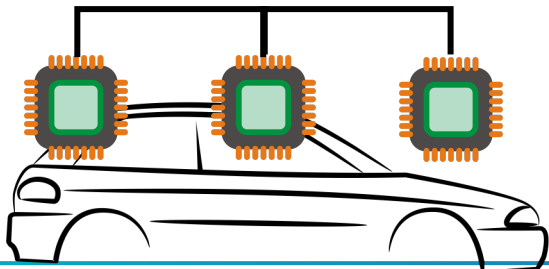
Let's assume sensor data is available on a CAN bus

CAN signals may be described in DBC format



```
github.com/eclipse-kuksa/kuksa-can-prov
Files main kuksa-can-provider / Model3CAN.dbc Edit this file
Raw [Icons]
[0|102.3] "%" Receiver
[0|102.3] "%" Receiver
[0|100] "%" Receiver
(1,0) [0|255] "" Receiver
[401+ (1,0) [0|15] "" Receiver
[901+ (1,0) [0|510] "" Receiver
DI_uiSpeedHighSpeed : 34|901+ (1,0) [0|510] "" Receiver
2660 SG_DI_uiSpeedUnits : 33|101+ (1,0) [0|1] "" Receiver
2661 SG_DI_vehicleSpeed : 12|1201+ (0.08,-40) [-40|285] "kph" Receiver
2662
2663 BO_ 680 ID2A8CMPD_state: 8 VehicleBus
2664 SG_CMPD_inputHVCCurrent : 32|901+ (0.1,0) [0|50] "A" Receiver
2665 SG_CMPD_inputHVPower : 21|1101+ (10,0) [0|20000] "W" Receiver
2666 SG_CMPD_inputHVVoltage : 41|1101+ (0.5,0) [0|1000] "V" Receiver
2667 SG_CMPD_powerLimitActive : 55|101+ (1,0) [0|1] "" Receiver
2668 SG_CMPD_powerLimitTooLowToStart : 62|101+ (1,0) [0|1] "" Receiver
2669 SG_CMPD_ready : 63|101+ (1,0) [0|1] "" Receiver
2670 SG_CMPD_speedDuty : 11|1001+ (0.1,0) [0|100] "%" Receiver
2671 SG_CMPD_speedRPM : 0|1101+ (10,0) [0|20000] "RPM" Receiver
2672 SG_CMPD_state : 56|401+ (1,0) [0|15] "" Receiver
2673 SG_CMPD_wasteHeatState : 60|201+ (1,0) [0|3] "" Receiver
2674
2675 BO_ 1029 ID405VIN: 8 VehicleBus
2676 SG_VINB405 m17 : 8|5601+ (1,0) [0|7.20576E+016] "" Receiver
2677 SG_VINC405 m18 : 8|5601+ (1,0) [0|7.20576E+016] "" Receiver
2678 SG_VINA405 m16 : 8|5601+ (1,0) [0|7.20576E+016] "" Receiver
```

CAN bus

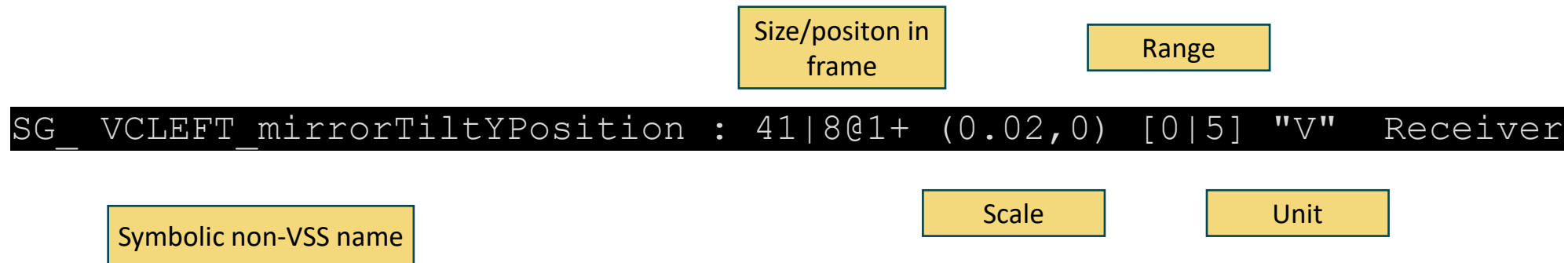


Deeply Embedded

CAN DBC

Many automotive busses and communication protocols use description of the data

- DBC for CAN, ARXML for various AUTOSAR communication needs, DDS-IDL for DDS systems etc.
- We focus on DBC for CAN



Can we describe the relation to VSS signals?

DBC overlay: Describe VSS-DBC relation in VSS

```
Vehicle.Body.Mirrors.DriverSide.Tilt:  
  datatype: int8  
  type: actuator  
  dbc2vss:  
    signal: VCLEFT_mirrorTiltYPosition  
    transform:  
      math: "floor((x*40)-100)"
```

Good to know

- VSS allows you to define custom metadata that will be processed by VSS-tools

Standard VSS definition

Reference DBC signal name

Apply transform
(example converts 0..5V into -100/+100 range demanded by VSS)

Helpful commands

```
vspec2json.py -o overlay_simple.vspec -o ./dbc_overlay.vspec -e dbc2vss --json-  
./vss_rel_4.1.yaml myextvsswithdbc.json
```

Interlude: What if Tilt not supported?

```
Vehicle.Body.Mirrors.DriverSide.Tilt:  
  datatype: int8  
  type: actuator  
  dbc2vss:  
    signal: VCLEFT_mirrorTiltYPosition  
    transform:  
      math: "floor((x*40)-100)"
```

Good to know

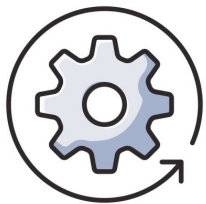
- VSS allows you to define custom metadata that will be processed by VSS-tools

Standard VSS definition

Reference DBC signal name

Apply transform
(example converts 0..5V into -100/+100 range demanded by VSS)

“tilt sensor A”



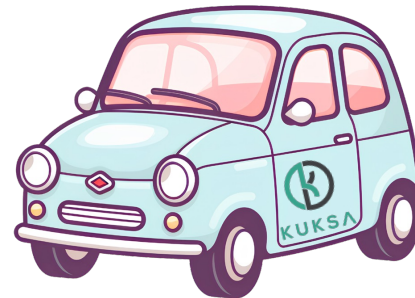
“tilt sensor B”



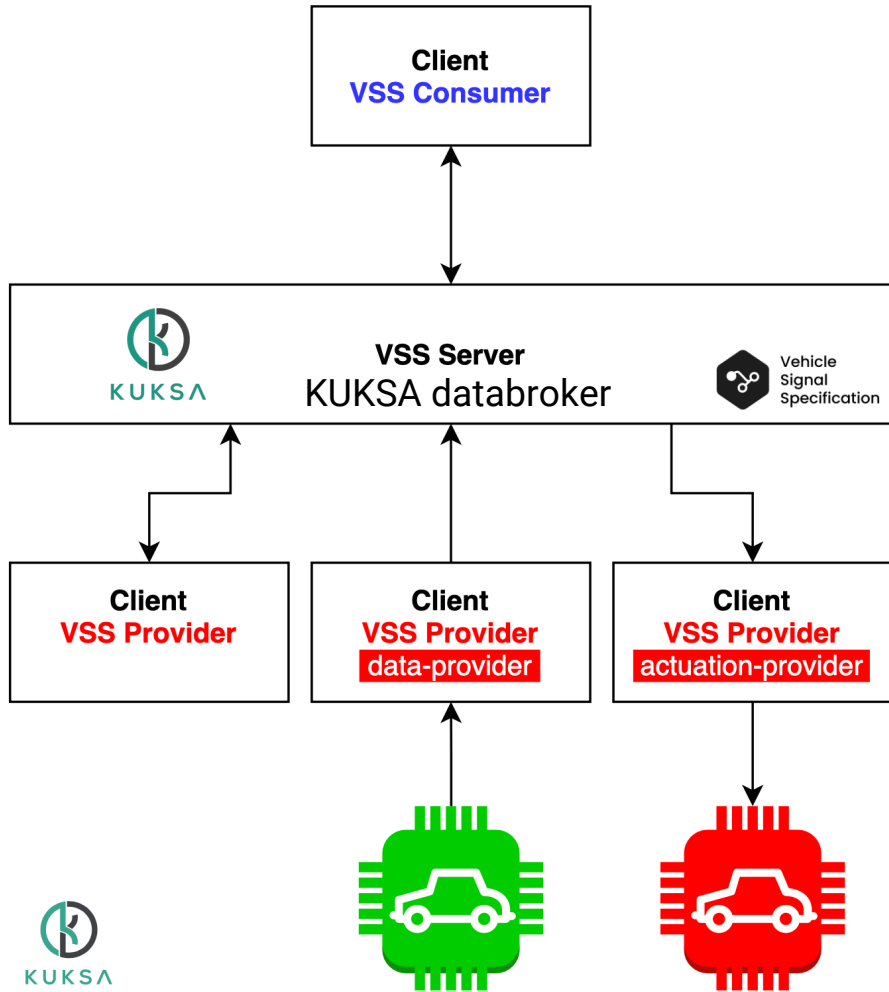
“without tilt”



VEHICLE SPECIFIC MAPPINGS

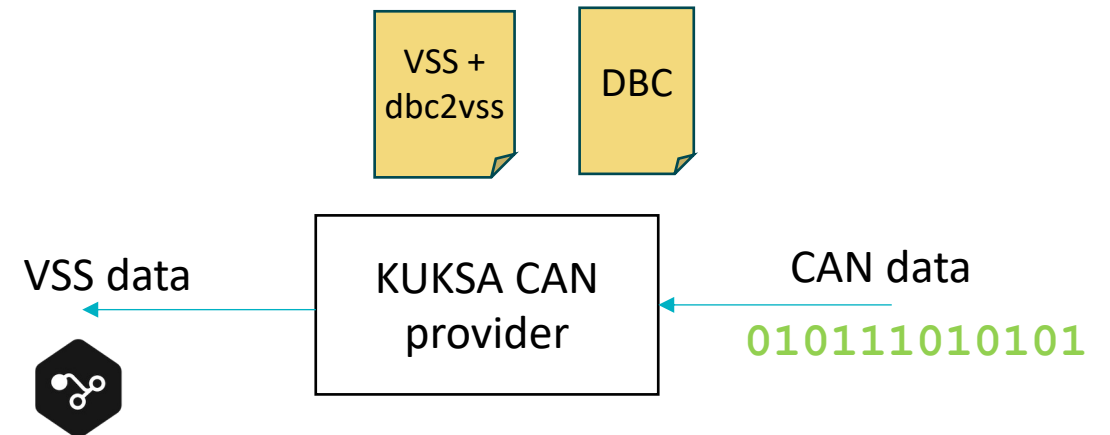


KUKSA CAN provider



VSS provider syncs of the vehicle with VSS model of the server

- **data-provider** makes sure that the actual state of a vehicle is represented in VSS (historically known as “feeder”)
- **actuation-provider** ensure that the target value of a VSS actuator is reflected by the actual state of a vehicle



Running CAN provider

```
testdemo — docker run -it --rm --net=host -v ~/Documents/Dev/testdemo:/data ghcr.io/eclipse-kuksa/kuksa-can...
2024-04-15 14:50:43,975 INFO dbcfeeder: Reading configuration from file: config/dbc_feeder.ini
2024-04-15 14:50:43,975 INFO dbcfeeder: DBC2VAL mode is: True
2024-04-15 14:50:43,975 INFO dbcfeeder: VAL2DBC mode is: False
2024-04-15 14:50:43,975 INFO dbcfeeder: Path to token information not given
2024-04-15 14:50:43,976 INFO dbcfeeder: Starting CAN feeder
2024-04-15 14:50:43,976 INFO dbcfeederlib.dbcparser: Reading definitions from DBC file Model3CAN.dbc
2024-04-15 14:50:44,166 INFO dbcfeederlib.dbc2vssmapper: Reading CAN->VSS mapping definitions from file /data/myextvsswithdbc.json
2024-04-15 14:50:44,166 INFO dbcfeederlib.dbc2vssmapper: Reading default CAN signal values from file dbc_default_values.json
2024-04-15 14:50:44,167 INFO dbcfeederlib.databrokerclientwrapper: Connecting to Data Broker using 127.0.0.1:55555
2024-04-15 14:50:44,167 INFO dbcfeederlib.databrokerclientwrapper: No token path specified. KUKSA.val Databroker must run without authentication!
2024-04-15 14:50:44,167 INFO kuksa_client.grpc: No Root CA present, it will not be possible to use a secure connection!
2024-04-15 14:50:44,167 INFO kuksa_client.grpc: Establishing insecure channel
2024-04-15 14:50:44,172 INFO dbcfeeder: Setting up reception of CAN signals
2024-04-15 14:50:44,172 INFO dbcfeeder: Using DBC reader
2024-04-15 14:50:44,172 INFO dbcfeederlib.canreader: Using CAN frame ID whitelist=[{'can_id': 1022, 'can_mask': 2047}, {'can_id': 599, 'can_mask': 2047}]
2024-04-15 14:50:44,172 INFO dbcfeederlib.canplayer: Starting repeated replay of CAN messages from log file /data/candemo.log
2024-04-15 14:50:44,173 INFO dbcfeederlib.databrokerclientwrapper: Connectivity to data broker changed to: ChannelConnectivity.READ...
2024-04-15 14:50:44,173 INFO c
2024-04-15 14:50:44,174 INFO c
2024-04-15 14:50:44,174 INFO c
2024-04-15 14:50:44,174 INFO c
2024-04-15 14:50:44,175 INFO c
2024-04-15 14:50:44,176 INFO c
2024-04-15 14:50:44,176 INFO c
2024-04-15 14:50:44,176 INFO c
2024-04-15 14:50:44,177 INFO c
```

```
scs2rng — docker run -it --rm --net=host ghcr.io/eclipse-kuksa/kuksa-python-sdk/kuksa-client:latest — 86...
"timestamp": "2024-04-15T14:51:46.573352+00:00"
},
"metadata": {
  "data_type": "UNSPECIFIED",
  "entry_type": "UNSPECIFIED"
}
},
"fields": [
  "VALUE"
]
}
]
[
{
  "entry": {
    "path": "Vehicle.Chassis.Axle.Row2.Wheel.Left.Brake.Temperature",
    "value": {
      "value": 29.0,
      "timestamp": "2024-04-15T14:51:46.574605+00:00"
    },
    "metadata": {
      "data_type": "UNSPECIFIED",
      "entry_type": "UNSPECIFIED"
    }
  },
  "fields": [
    "VALUE"
  ]
}
]
[
{
```

Helpful commands

```
docker run -it --rm --net=host -v $(pwd):/data ghcr.io/eclipse-kuksa/kuksa-can-provider/can-provider:0.4 --mapping /data/myextvssmodelwithdbc.json --dumpfile /data/candemo.log
```

Making an App

```
async def present():
    while True:
        print("⊛ Processed: " + str(nbr_vss_signals) + " VSS signals!")
        print("🔧 FL brake: " + str(vss_values['Vehicle.Chassis.Axle.Row1.Wheel.Left.Brake.Temp
        print("🔧 FR brake: " + str(vss_values['Vehicle.Chassis.Axle.Row1.Wheel.Right.Brake.Temp
        print("🔧 RL brake: " + str(vss_values['Vehicle.Chassis.Axle.Row2.Wheel.Left.Brake.Temp
        print("🔧 RR brake: " + str(vss_values['Vehicle.Chassis.Axle.Row2.Wheel.Right.Brake.Temp
        print("🚀 Max speed since last time: " + str(vss_values['Vehicle.Speed']))
        reset_vss_values()
        await asyncio.sleep(5)
```

Good to know

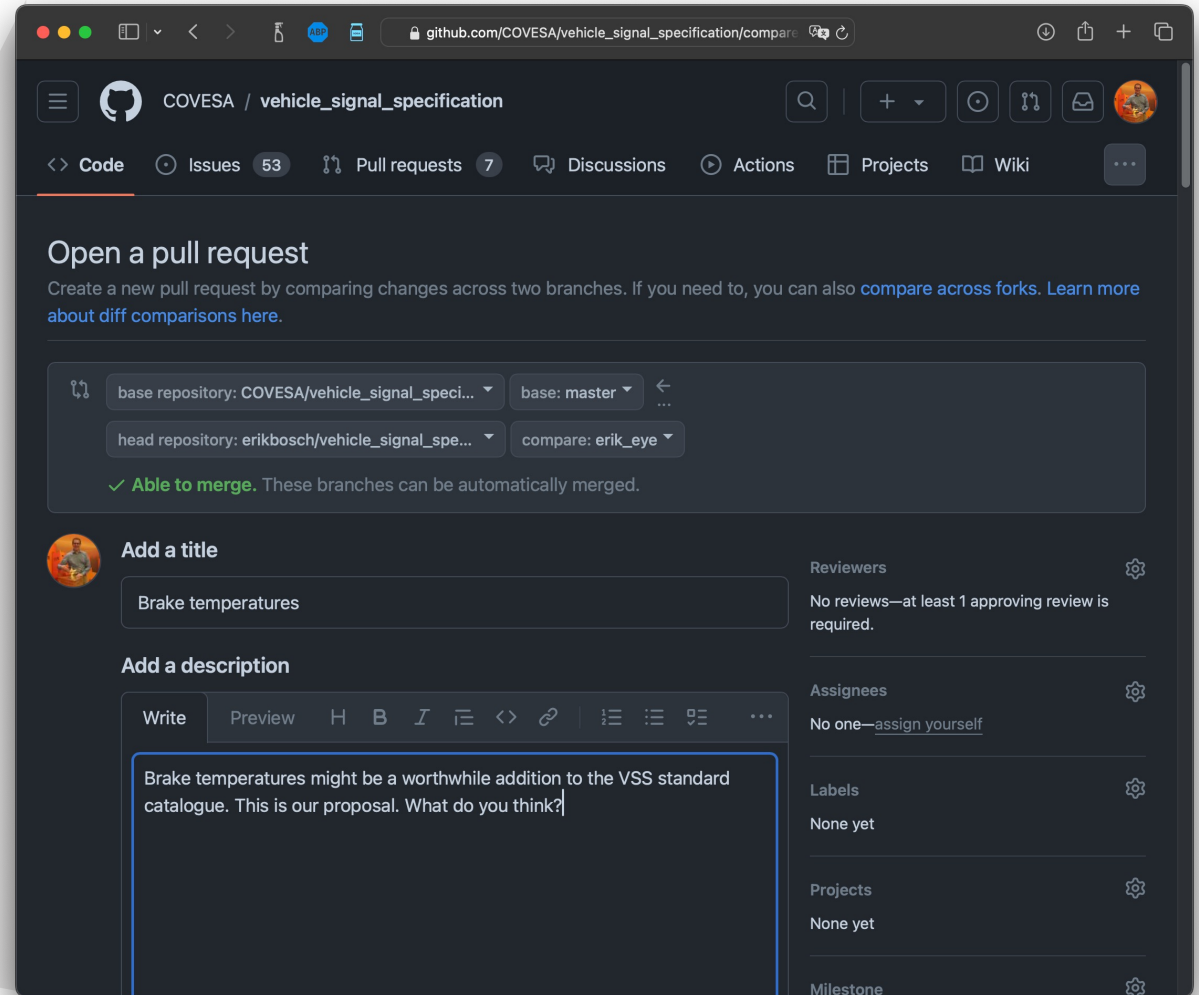
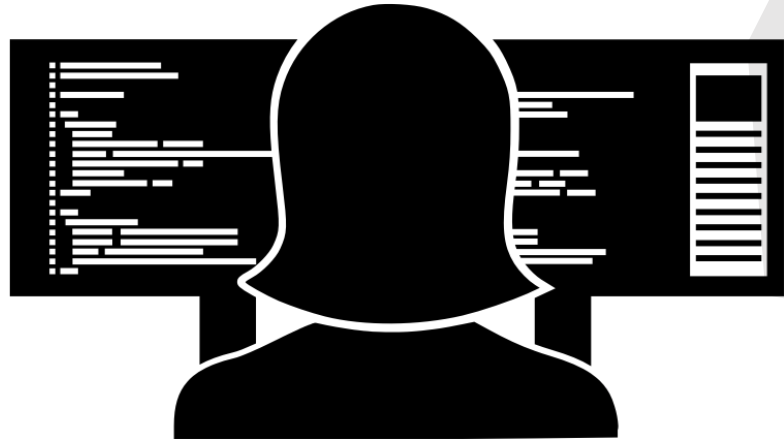
- KUKSA Python SDK is an easy way to use the KUKSA GRPC API
- You can use GRPC directly in any supported language
- You can use the KUKSA Android SDK for Smartphones, Tablets and Android Automotive

Helpful commands

```
pip install kuksa-client
python app.py
```


Happy End

Would have been even easier if the signal had already been there

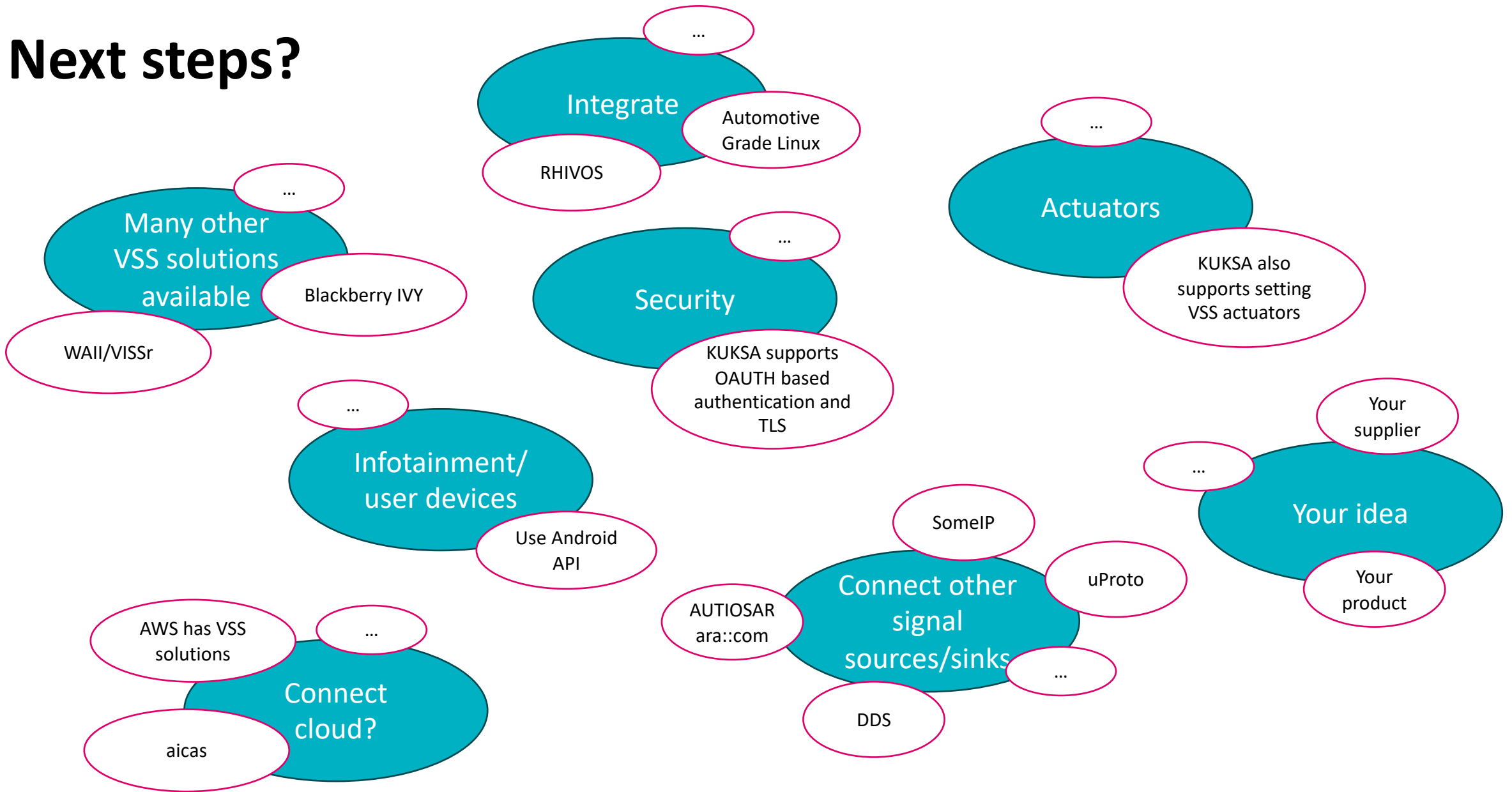


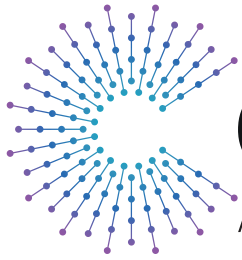
Summary

We have seen

- How to use the vss-tools to convert VSS into other useful formats
- How to add custom VSS signals and converting them using the tools
- How to model and add custom VSS metadata VSS signals and converting them using the tools
- How to run the Eclipse KUKSA software to work with VSS signals in-vehicle
- How to use a CAN provider automatically converting raw CAN signals into valid VSS signals using a configuration
- How to write an application using VSS signals

Next steps?





COVESA

Accelerating the future of connected vehicles

COVESA VSS



Vehicle
Signal
Specification

https://covesa.github.io/vehicle_signal_specification/

/me



<http://sdv.expert>

KUKSA



KUKSA

<https://eclipse.github.io/kuksa.website/>

Examples



<https://wiki.covesa.global/>

ETAS OSS



<https://www.etas.com/en/open-source-software.php>