Vehicle Signal Specification

2016-04-28 10:30 – 11:30

Magnus Feuer
Head System Architect | Expert Group LEad
Jaguar Land Rover

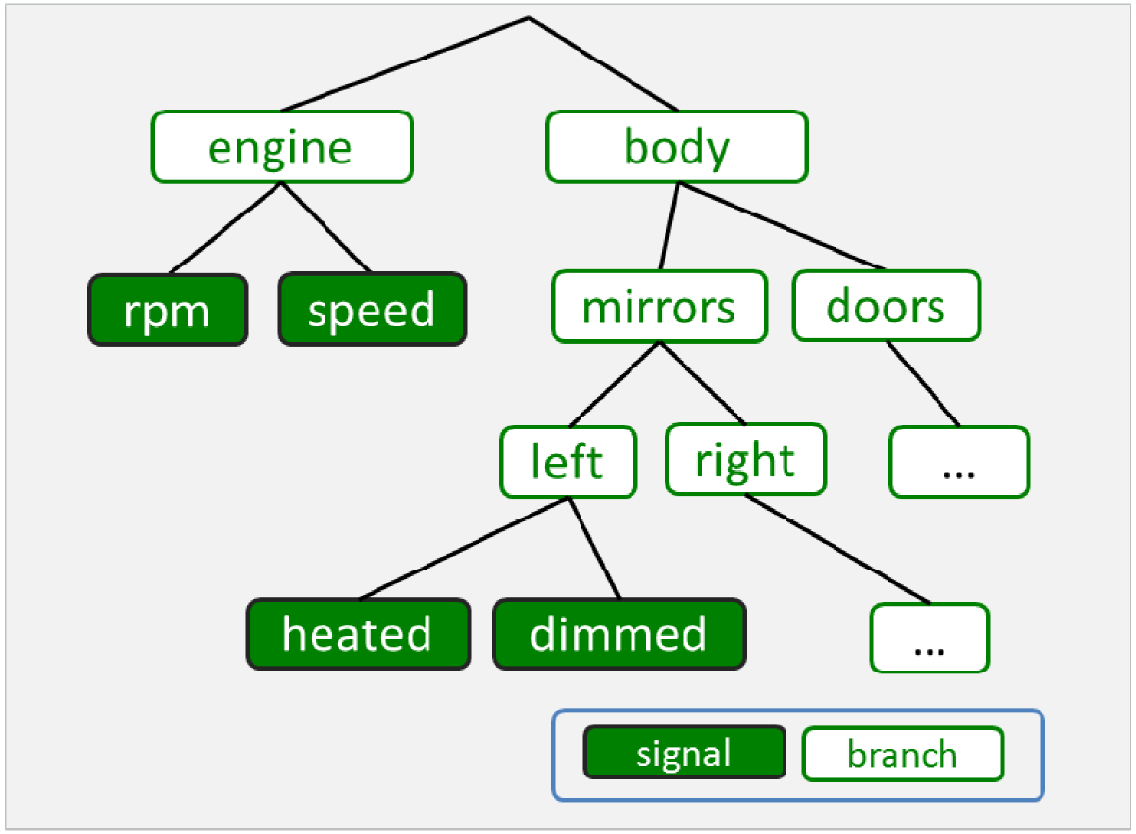
The Problem

- Vehicle state is being off boarded to Internet services
- There is no standard / process that fits the bill
- No public forum where changes can be processed in a lightweight manner
- One format does not suit all
- Decouple IVI from electric architecture

VSS - Introduction

- Standardizing signal specification
- YAML subset
- Minimum attributes
- Lightweight change process
- Single source – multiple targets
- Feed other standardization organizations (W3C, etc)
- Technically simple

VSS Signal structure



Naming Convention

```
body.mirrors.left.heated
body.mirrors.right.heated
body.door.front.left.open
body.door.back.left.open
```

- Dot notated name path
- Last component is signal

Specification source format: Branches

```
- transmission:
  type: branch
  description: Transmission-specific data, stopping at the drive shafts.
```

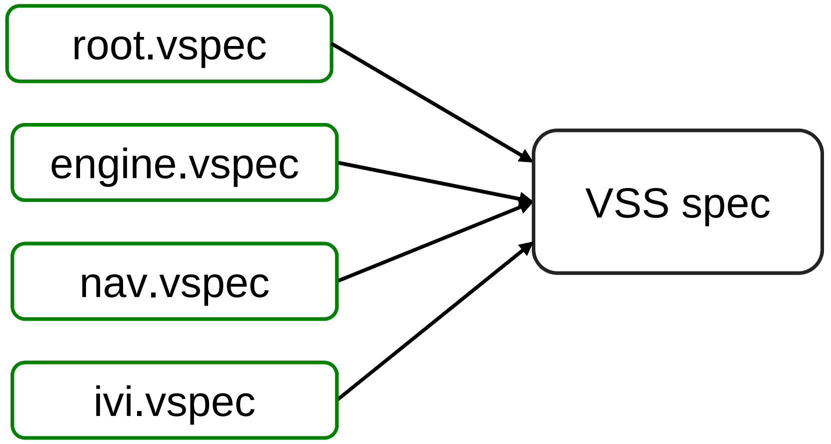
- YAML list
- Only type and description mandatory

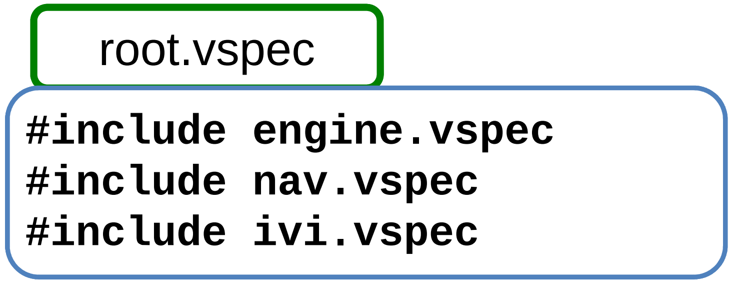
Specification source format: Signals

```
- speed:
  type: Uint16
  unit: km/h
  min: 0
  max: 350
  description: Vehicle speed
```

- Uses Franca typing
- Optional interval
- Optional SI unit type
- Can be enumerated

Signal source format

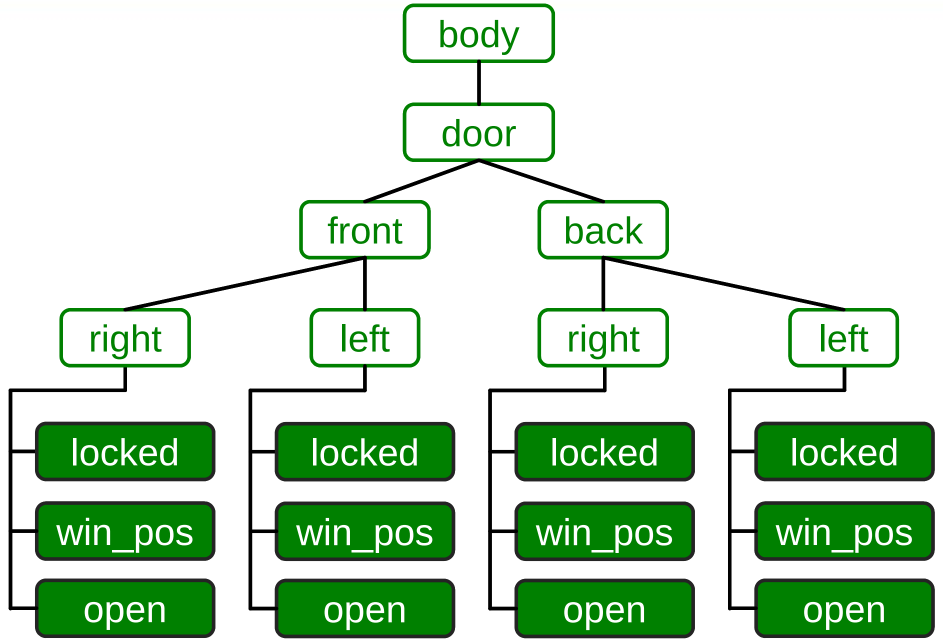



root.vspec

```
#include engine.vspec
#include nav.vspec
#include ivi.vspec
```

- Multiple files aggregated together to a uniform specification
- YAML-compliant include directives used to aggregate spec fragments
- Facilitates git(hub) working model
- Minimizes commit conflicts

Spec file re-use



door.vspec

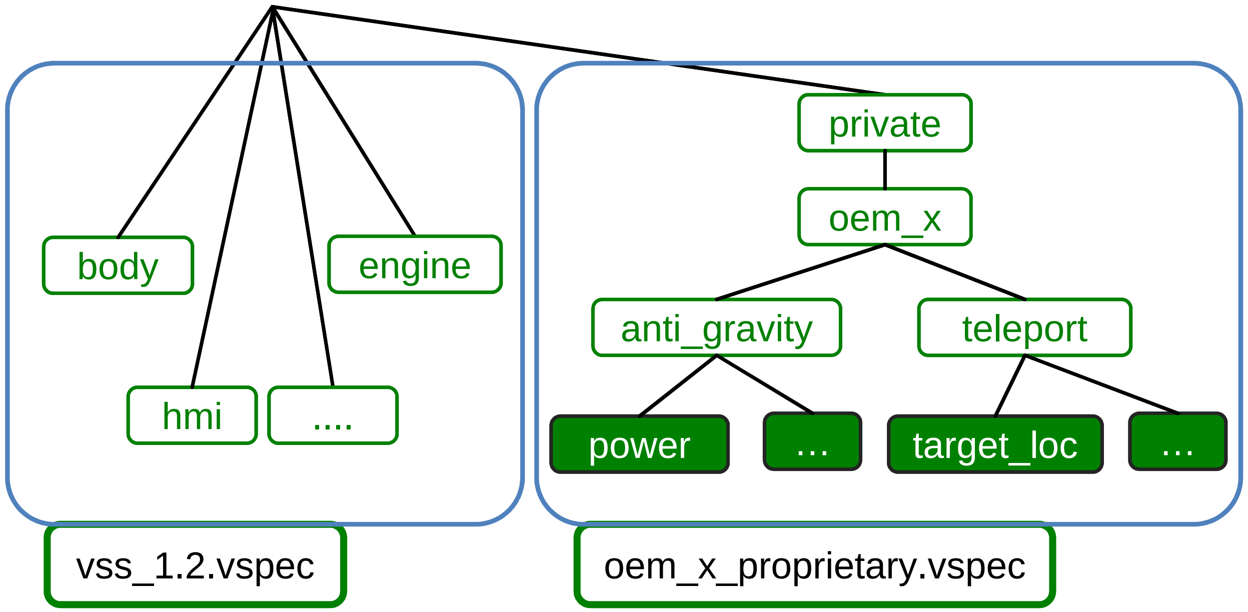
```
locked: …
win_pos: …
open: …
```

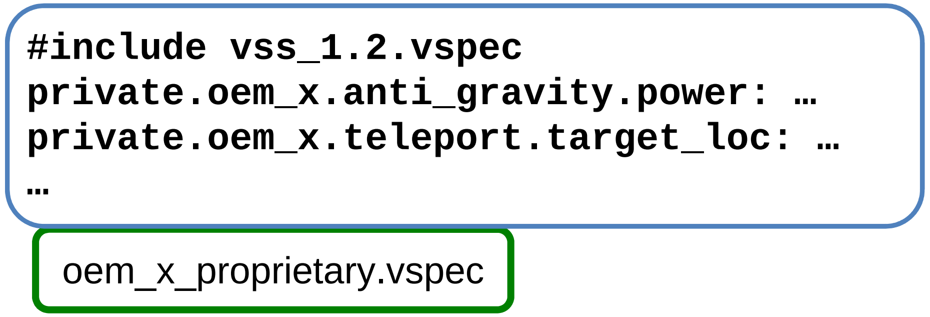
root.vspec

```
#include door.vspec body.door.front.left
#include door.vspec body.door.front.right
#include door.vspec body.door.back.left
#include door.vspec body.door.back.left
```

- YAML-compliant include directives used to aggregate specification fragments
- An update to a fragment is propagated to all locations where it is used
- Facilitates git(hub) working model

Private extensions



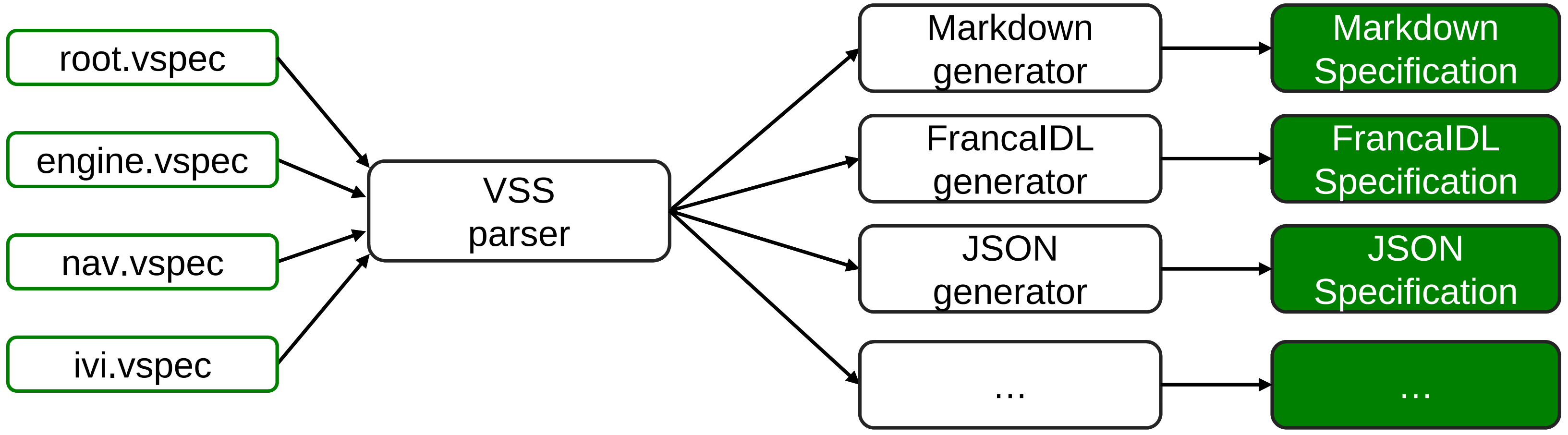


```
#include vss_1.2.vspec
private.oem_x.anti_gravity.power: …
private.oem_x.teleport.target_loc: …
…
```

oem_x_proprietary.vspec

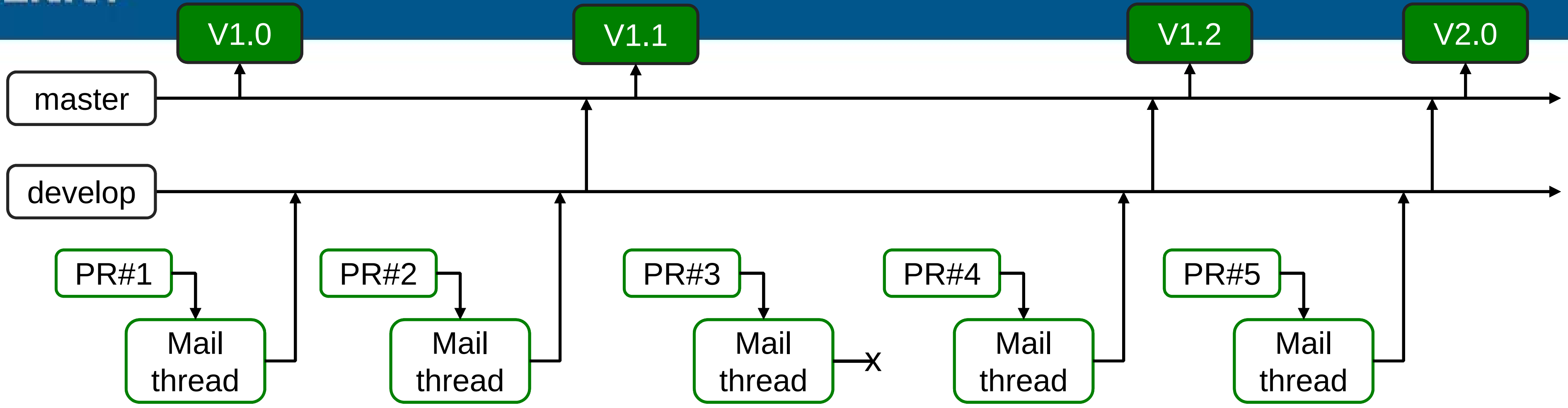
- A proprietary signal specification can use the GENIVI VSS as a starting point
- Can be used in production project to integrate with vendors
- Mature private extensions can be submitted for VSS inclusion

Generating target specifications



- Parser loads and interprets specification files
- Generators produces target documents and specifications
- Targets can be used as input to production projects and other organizations
- Additional generators can be added as needed.

Release management



- Pull requests submitted by anyone
- Mail discussion on genivi-projects list to approve request into develop branch
- Develop branch merged into master prior to tagged release
- Major number changes when existing tree structure is changed

More Info

`github.com/PDXostc/vehicle_signal_specification`

Demo time