GENIVI®

Open Connectivity Foundation

2016-06-29

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

Purpose of expert group

Specify, Standardize, and Implement Core connectivity protocols and services between the IVI system and remote entities.

Use proven open source technologies to ensure that all protocols and services can be implemented securely and robustly.

Collaborate with existing organizations (W3C, OCF, IEEE, etc) to ensure broad adoption and acceptance, and to avoid duplication and competition.

RVI Main Activities

Core

Connectivity

- Utilize a wide array of data links to setup communication to and from vehicle, either P2P or via backend serve
- Provide encryption for secrecy, non repudiation, replay attack protection, etc
- Work with OMA, IEEE, and other organizations to standardize RVI and integrate existing communication standards

Authentication

- Prove the identity of communicating parties
- Use best-of-breed open source technologies to drive peer-reviewed security

Authorization

- Prove to remote parties the right to discover and invoke their services.

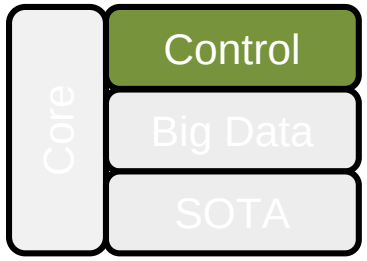
Service Discovery

- Announce services available to remote parties

Service Invocation

- Invoke services and report the result over unreliable data links that may change during execution
- Support retry and store & forward of service invocations to alleviate transient connectivity

RVI Control



Vehicle Integration

- Utilize Networking EG components to integrate with vehicle bus
- Use W3C-based signal standards to control vehicle

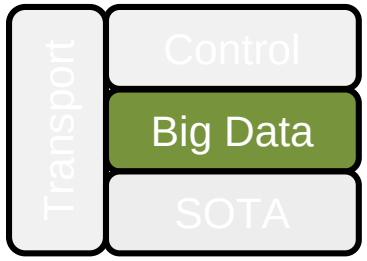
Service Protocol

- Define vehicle control protocols between vehicle and remote entities

Web Services

- Use W3C-based standards to define web services for remote vehicle control

RVI Big Data



Data collection

- Integrate with Genivi components to harvest data
- Use dynamically OTA-loadable code to securely collect and pre-process data

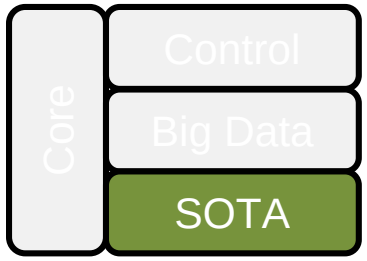
Reporting

- Specify and implement in-vehicle reporting services and their protocols

Data gathering

- Server-side data reception, storage, and web service access
- Work with big data industry actors to integrate analytics and data feeds into a larger ecosystem

RVI SOTA



Implement and standardize SOTA Client

- Integrate SOTA Client with System Infrastructure and its Software Management activities
- Implement existing transport protocol standards in addition to Transport

Implement and standardize SOTA Server

- Specify and implement reporting services and their protocols

Integrate with industry

- Work with vendors and community to drive RVI SOTA adoption in the automotive industry

From requirements to industry acceptance

Vehicle Requirement Phase

- Gather high-level purpose, objectives, and requirements from Genivi members

Discovery Phase

- Explore requirements and APIs through coding and experiments carried out in public

Specification Phase

- Use experience and consensus from discovery phase to create abstract and specific components

Acceptance Phase

- Use deliverables from specification phase to approach industry to gain wider acceptance

Technical Scope

Provide P2P based provisioning, authentication, authorization, discovery and invocation between services running inside and outside a vehicle.

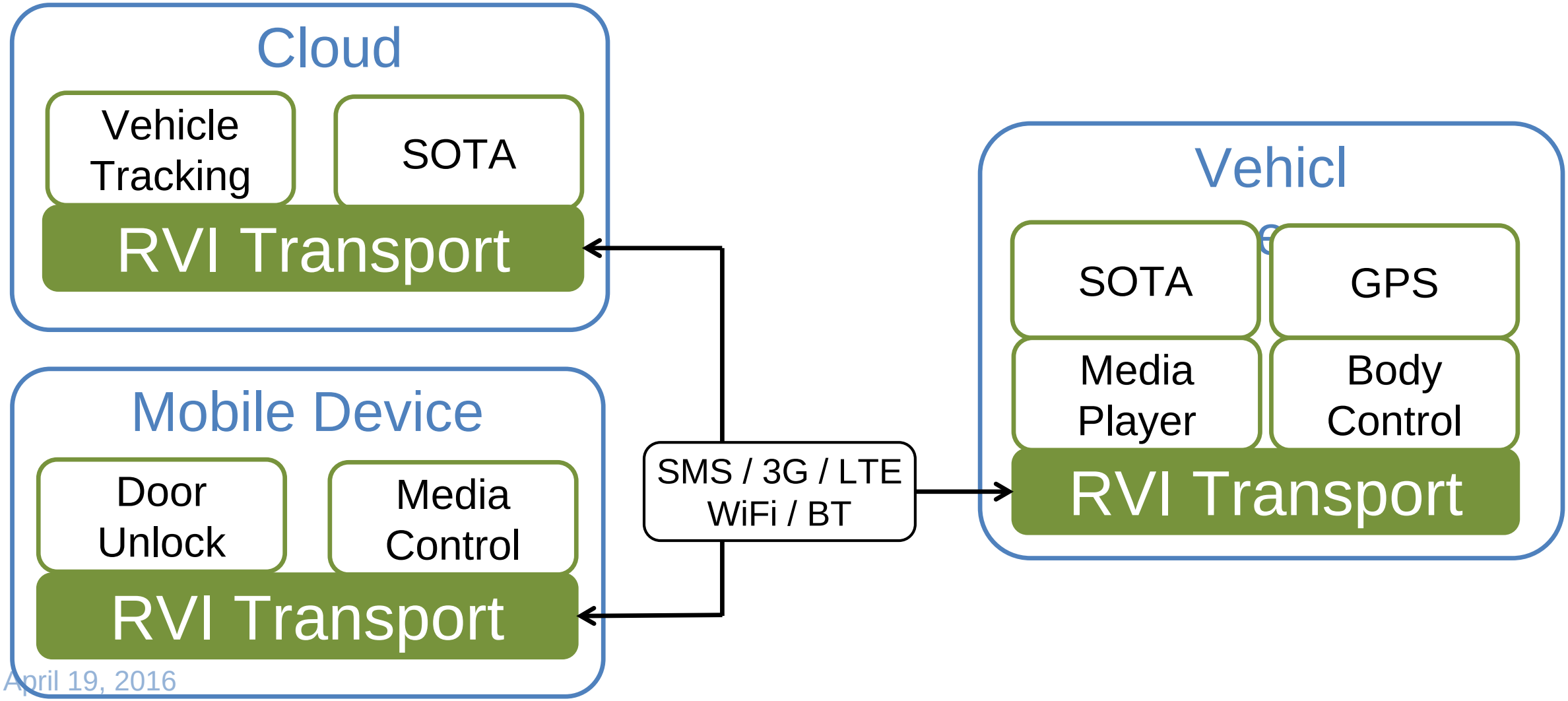
- **P2P**
  Internet connection not required for two peers to exchange services.

- **Provisioning**
  Add, delete, and modify services and network nodes.

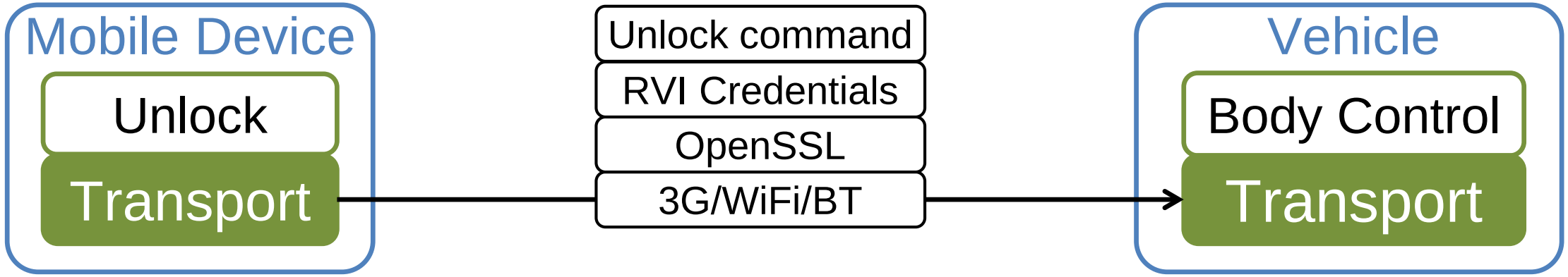
- **Authentication and Authorization**
  Proves that a service is who it claims to be, and has the right to invoke another service.

- **Discovery and Invocation**
  Allow two peers to exchange services that the other party is authorized to use, and invoke those services over any data link.

Schematics

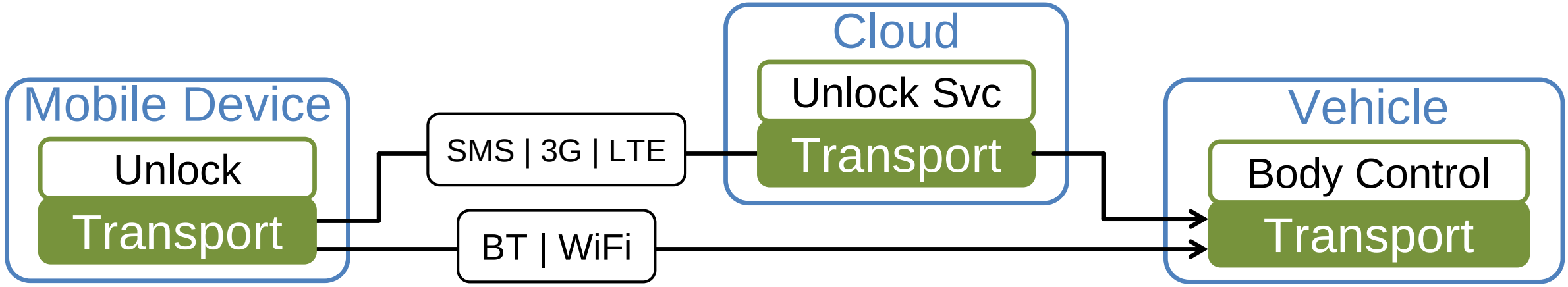

Cloud
Vehicle Tracking
SOTA
RVI Transport
Mobile Device
Door Unlock
Media Control
RVI Transport
SMS / 3G / LTE
WiFi / BT
Vehicl
SOTA
GPS
Media Player
Body Control
RVI Transport

Security



- **OpenSSL**
  TLS provides core eavesdropping and MITM attack protection
- **RVI Credentials**
  Signed by root server to prove device authenticity and its right to invoke unlock on the given vehicle
- **Unlock**
  Will only be accepted by vehicle if validated certificate specifies device's right to invoke unlock command

Robustness



- **Multi-pathed commands**
  Commands can be tried over multiple data links, both as peer-to-peer and via backend server.

- **Traffic prioritization**
  Forces high-bandwidth services to use WiFi while allowing mission-critical services to use 3G and SMS.

- **Multi-protocol**
  A single service (such as unlock) can employ different protocols and data links depending on the targeted vehicle type.

Application example – Mobile Unlock

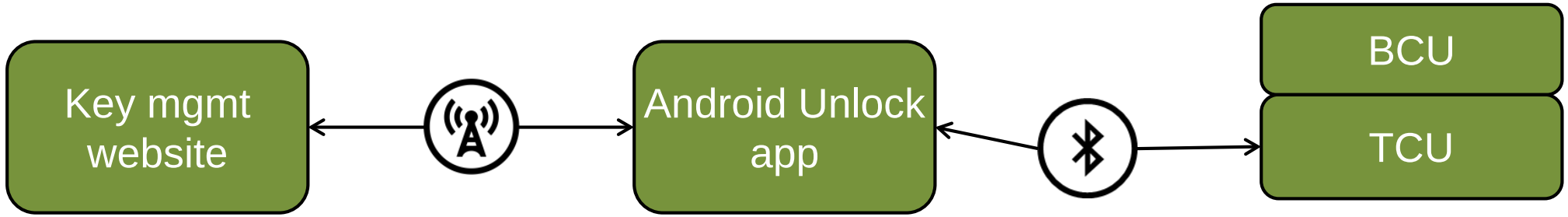
The user's story

An owner is travelling when a friend emails her to ask if he can borrow her car.

The owner brings up her guest driver app and sends a virtual car key, valid over the weekend, to her friend.

After the friend receives the key, he can walk up to the car to have his phone automatically unlock and start it.

Overview – Demo system



Key management website [Cloud]
Allow vehicle owners to provision and transmit keys to any android app

Android unlock app [Mobile Device]
Runs a background service using keys to lock/unlock select vehicles

Telematics and Body Control Unit

Receives, validates, and executes received lock/unlock command

Vehicle Signal Specification

A single standard to monitor and control all vehicles

April 19, 2016

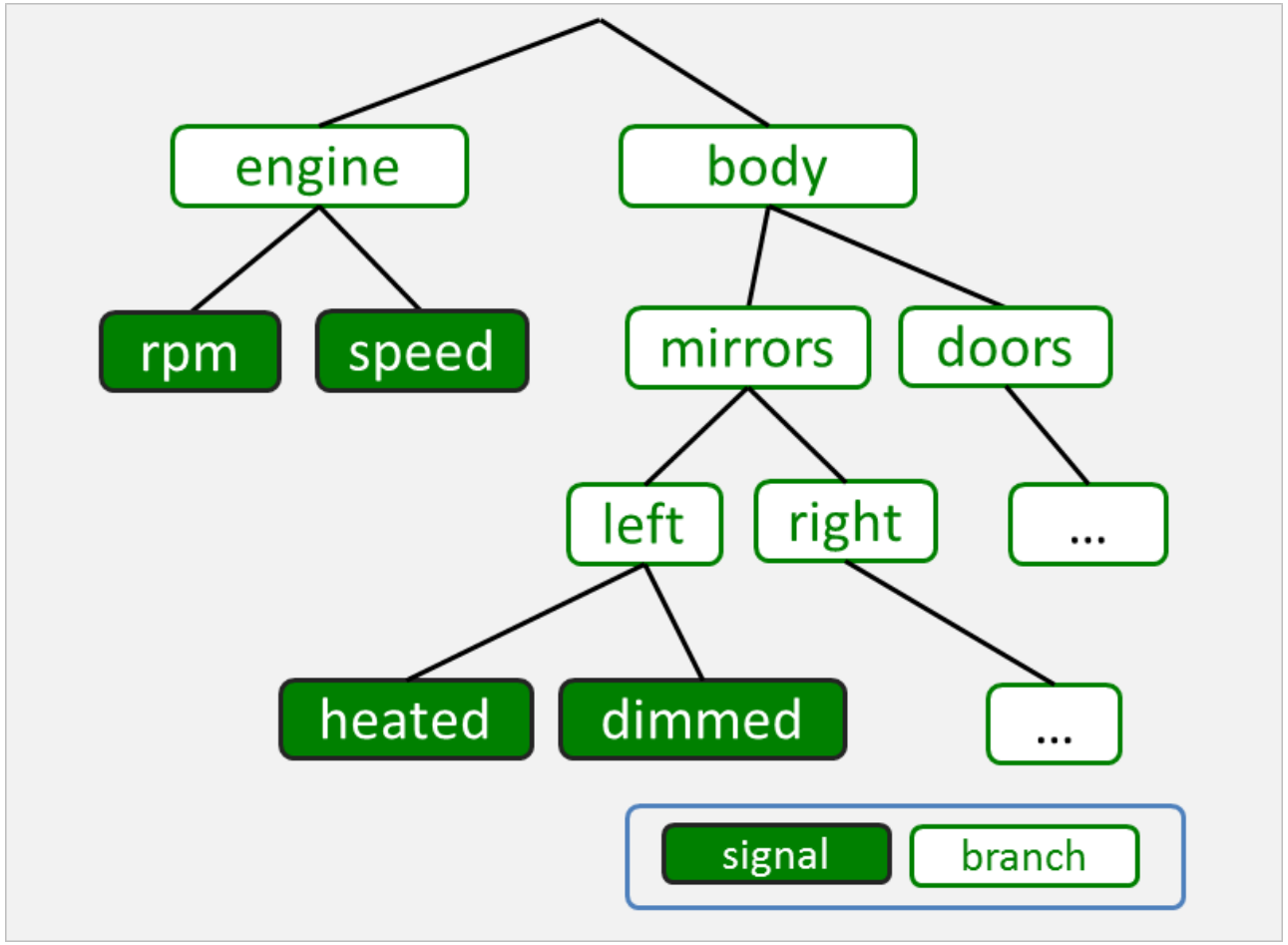
The Problem

- 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.
```

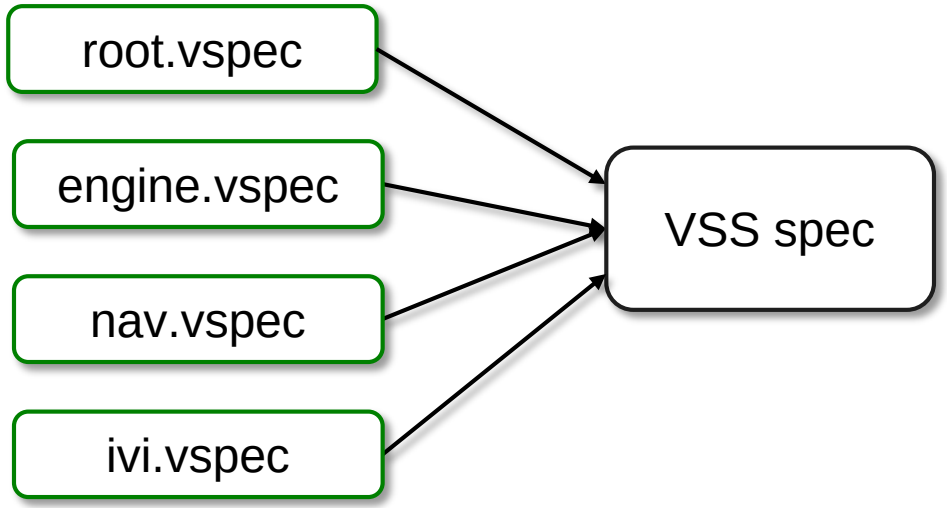
- YAML list
- Only type and description mandatory

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

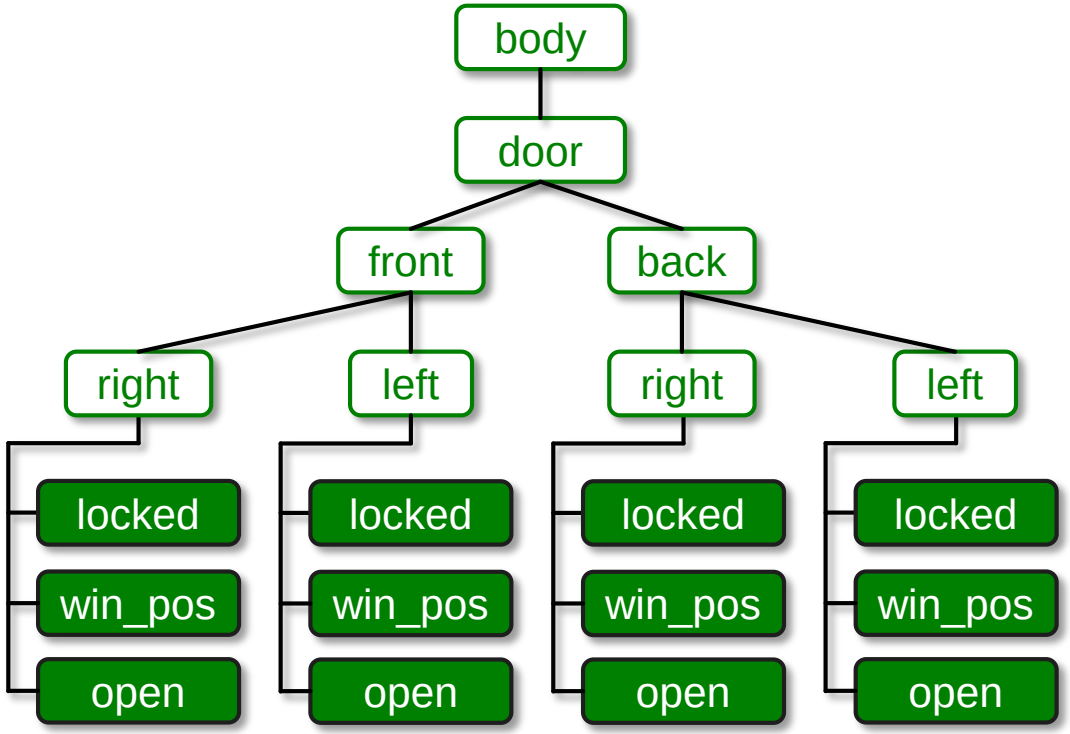


root.vspec

```
#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

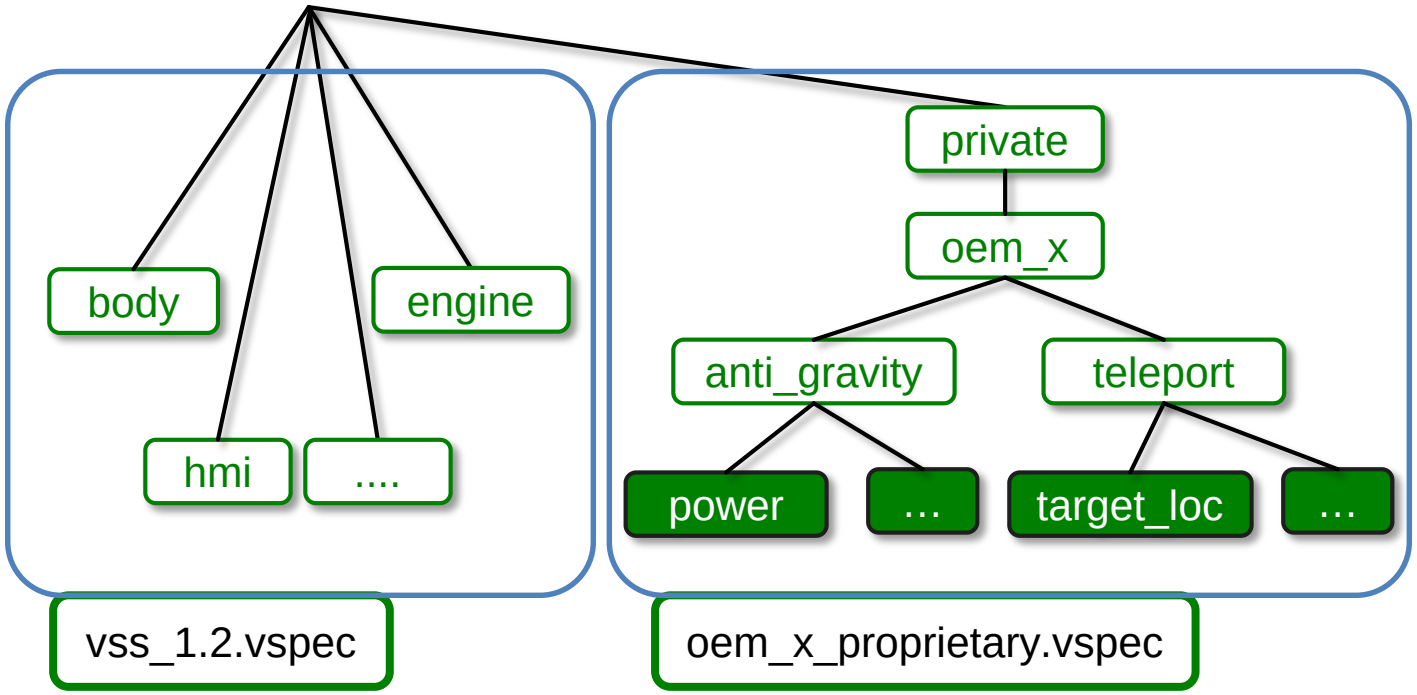
```
locked: …
win_pos: …
open: …
```

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

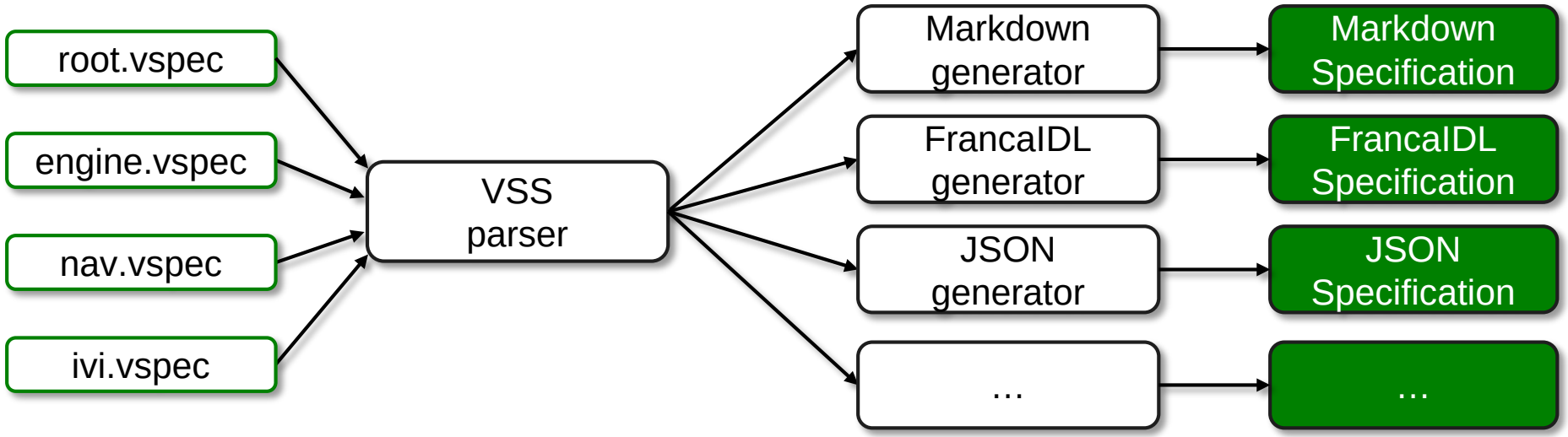



```
#include vss_1.2.vspec
private.oem_x.anti_gravity.power: …
private.oem_x.teleport.target_loc: …
…
```

oem_x_proprietary.vspec

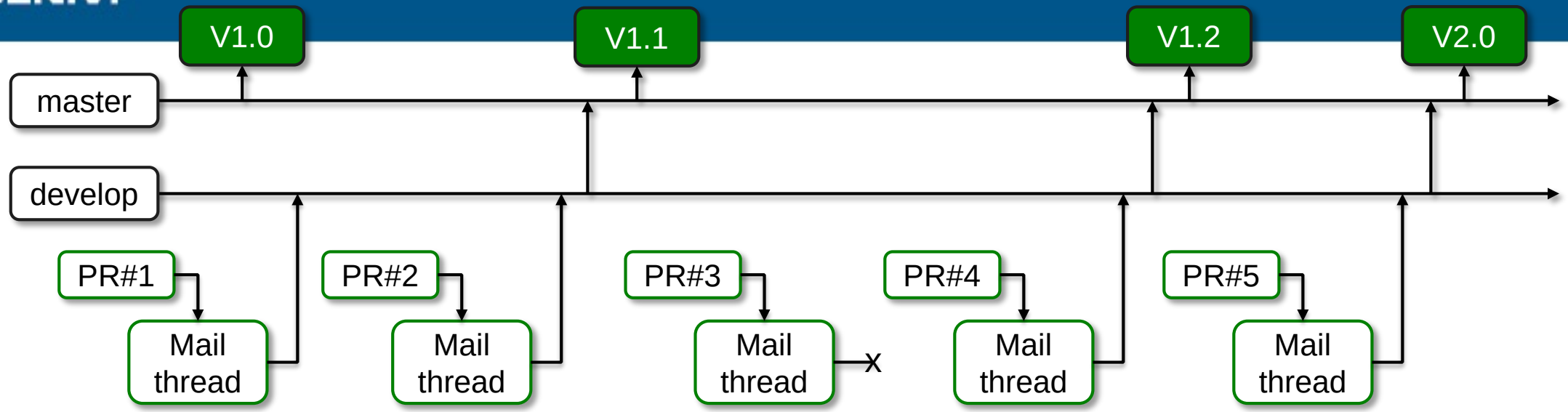
- A proprietary signal specification can use the GENIVI VSS as a starting point
- Can be used in production project to integrate with vendors
- Mature private extensions can be submitted for VSS inclusion

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