Eclipse Kuksa

Eclipse SDV Contribution Day,
Sven Erik Jeroschewski, Bosch.IO
Eclipse Kuksa
The Kuksa Ecosystem in 2018

Mission: Create a cross-vendor connected vehicle platform that relies on open standards and uses open source software to leverage the potential of a large developer community!

- use and foster Open Source solutions
  - create a harmonized composition of existing Open Source projects
  - enriched with specific Kuksa components
Software Infrastructure for Vehicles

What if ..

...we can access **standardized** car data in our cloud backend

...we can use **standard IT technologies** to run software around vehicles

...we can use these applications **to interact with the car**

...everybody has the option to use these features

...we can focus on developing customer value instead of reinventing the whole stack
Kuksa.val (Vehicle Abstraction Layer) implements an in-vehicle server for the Covesa VSS (Vehicle Signal Specification) industry standard.
Kuksa.val

VSS – Vehicle Signal Specification

A generic, extensible data model currently specified within COVESA

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

A protocol to access data based on the COVESA VSS model specified within W3C

https://www.w3.org/TR/vehicle-information-service/
System Overview
Kuksa.val
System Overview

KUKSA.val
KUKSA Hardware

CAN

Data
Kuksa.val
System Overview

KUUKSA.val

DBC Feeder
KUKSA Hardware

CAN

DBC file
VSS Mapping

Data
Kuksa.val System Overview

```
{
  "action": "subscription",
  "subscriptionId": "42",
  "path": "Vehicle.Speed",
  "value": "100"
}
```

KUUKSA.val

**DBC Feeder**

**KUUKSA Hardware**

- Query
  - REST
  - Pub/sub
    - Websocket

- DBC file
- VSS Mapping

**CAN**

Information

Data
Kuksa.val
System Overview

Data Consumer
Query
REST
Pub/sub
Websocket

KUUKSA.val
DBC Feeder
KUUKSA Hardware

Knowledge
Information
Data

DBC file
VSS Mapping

CAN
Kuksa.val

System Overview

Cloud

Application

KUKSA.val

DBC Feeder

KUKSA Hardware

DBC file

VSS Mapping

Knowledge

Information

Data

Onboard

Offboard
Kuksa.val

Features

- Kuksa.Val Server: providing access to VSS data using W3C VISS websockets or gRPC (written in C++)
- Kuksa.Val Databroker (in Rust) for accessing VSS data using gRPC
- Python & Go SDK
- Interactive test client (CLI)
- Feeders (DBC, GPS, Replay)
- Examples
  - Docker images for arm64 and x86_64
Welcome to fish, the friendly interactive shell

>
Kuksa.hardware
Open Hardware for Kuksa

- Baseboard for Raspberry Pi Compute Module 4
- Schematics are Open Source
- Access vehicle data on OBD (only CAN supported)
Updates
Upcoming topics

• Cloud:
  • Kuksa.Cloud @ Home

• Val:
  • Unified gRPC API for relevant components like Val-Server, Databroker, Python SDK, Go SDK
  • All feeders work with val-server and databroker
  • Databroker differentiates between current and target values for actuators
  • Python SDK automatically released in PyPi
Thank You!
Stay in contact:

github.com/eclipse/kuksa.val
eclipse.org/kuksa

svenerik.jeroschewski@bosch.io
Back Up
Eclipse Kuksa Cloud Backend

Target Version

Third Party Services

App Store

Core Services

Big Data Analysis

Grafana Visualization

Report Generation

Data Management

(Automatic) Deployment

KEYCLOAK

hawkBit

ditto

HONO

(Eclipse Kuksa Cloud Backend

Target Version

Third Party Services

App Store

Core Services

Big Data Analysis

Grafana Visualization

Report Generation

Data Management

(Automatic) Deployment

KEYCLOAK

hawkBit

ditto

HONO
Eclipse Kuksa Cloud Backend
Version 2021

Third Party
Services

Ambassador
API-Gateway

Grafana
Visualization

Data
Management

influxdb

Hono Influx
Connector

App Store

hawkBit

KEYCLOAK

(Automatic) Deployment

HONO
Eclipse Kuksa
Kuksa IDE

- Based on Eclipse Che
- Allows Cloud and In-Vehicle Application development
- Platform independent
- Shared workspaces
- Almost configuration free
- Docker-based: VPN planned to allow remote / network independent cross compilation
Eclipse Kuksa
Kuksa In-Vehicle Platform

**Application layer:**
- Runs 3rd party apps on the platform
- Contains a Sandbox Environment & Additional Services

**Middleware layer (Yocto layer):**
- APIs to abstract the vehicles’ E/E architecture (W3C VISS, Sensoris…)
- Communication Services to manage network access and provide data from the vehicle
- Includes communication libs, protocols, security layers,…

**OS layer:**
- Reuse of OE’s existing services, layers, HW abstractions, AGL services, etc.