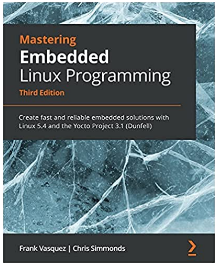
The Android Automotive Vehicle HAL

Chris Simmonds

Embedded World 2022

License

About Chris Simmonds


Mastering
Embedded
Linux Programming
Third Edition
Create fast and reliable embedded solutions with Linux 5.4 and the Yocto Project 3.1 (Dunfell)
Frank Vasquez | Chris Simmonds

- Consultant and trainer
- Author of *Mastering Embedded Linux Programming*
- Working with embedded Linux since 1999
- Android since 2009
- Speaker at many conferences and workshops

"Looking after the Inner Penguin" blog at `https://2net.co.uk/`

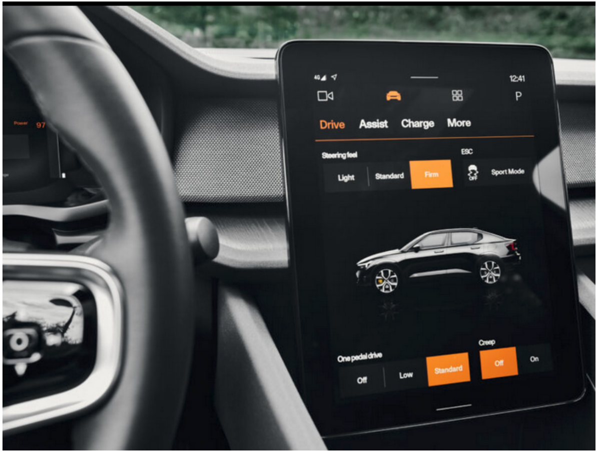
@2net_software

`https://uk.linkedin.com/in/chrisdsimmonds/`

Google and me

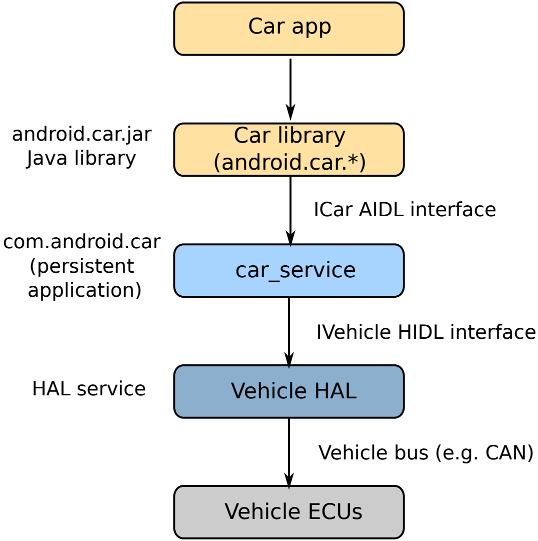
- I have no direct contact with Google
- I do not represent Google's point of view
- I have not signed any NDAs with Google

Android Automotive OS



Polestar 2

Architecture of Android Automotive



The Android Hardware Abstraction Layer

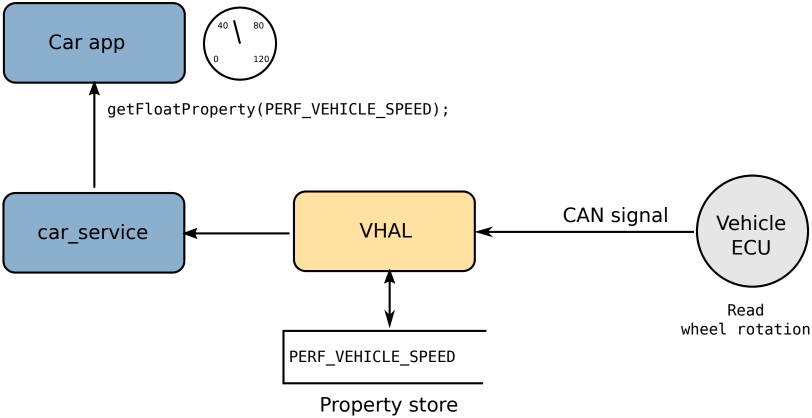
- The Hardware Abstraction Layer (HAL) sits between Android and hardware
- Divided into c. 50 interfaces
- Interfaces are written in HIDL(*) (deprecated) or Stable AIDL(**) e.g. the Vehicle HAL is IVehicle
- Most HALs implemented as a daemon (background) process

(*) HIDL = Hardware Interface Definition Language
(**) AIDL = Android Interface Definition Language

The Vehicle HAL

- The Vehicle HAL (VHAL) mediates between Android and the vehicle
- Allows apps and the Android framework to
  - Monitor variables, e.g. speed
  - Control variables, e.g. side window position
- Vehicle variables are represented as **vehicle properties**
- Properties have names such as `PERF_VEHICLE_SPEED` and `WINDOW_POS`

Monitoring, e.g. speed



Car app

```
getFloatProperty(PERF_VEHICLE_SPEED);
```

car_service

VHAL

CAN signal

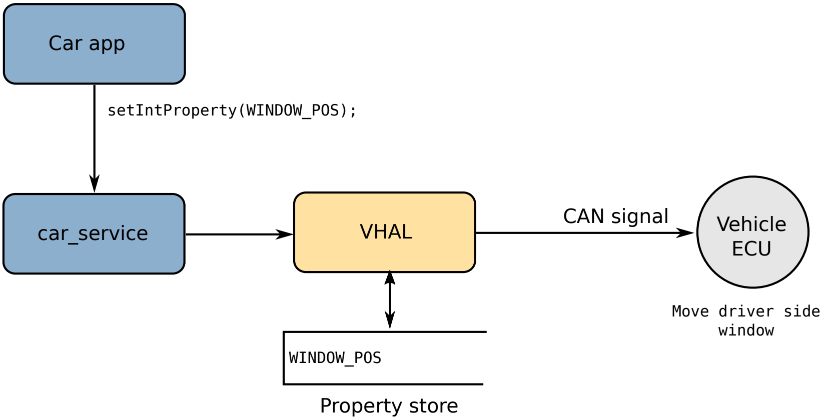
Vehicle ECU

Read wheel rotation

PERF_VEHICLE_SPEED

Property store

Controlling, e.g. window position



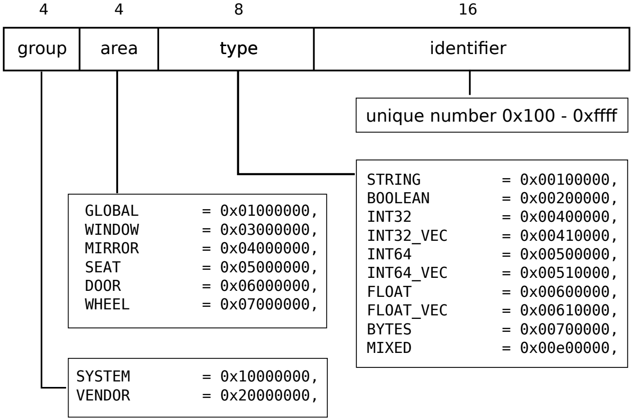
System and vendor vehicle Properties

- The VHAL defines two groups of properties:
- **SYSTEM**: c. 150 properties defined in `types.hal`
- **VENDOR**: defined by OEM, functions defined as needed

The SYSTEM properties and all the types associated with them are defined in file `hardware/interfaces/automotive/vehicle/2.0/types.hal`

Property identifier

A vehicle property is identified by a 32-bit number with this format:



4	4	8	16
group	area	type	identifier

unique number 0x100 - 0xffff

```
STRING          = 0x00100000,
BOOLEAN         = 0x00200000,
INT32           = 0x00400000,
INT32_VEC       = 0x00410000,
INT64           = 0x00500000,
INT64_VEC       = 0x00510000,
FLOAT           = 0x00600000,
FLOAT_VEC       = 0x00610000,
BYTES           = 0x00700000,
MIXED           = 0x00e00000,
```

```
GLOBAL          = 0x01000000,
WINDOW          = 0x03000000,
MIRROR          = 0x04000000,
SEAT            = 0x05000000,
DOOR            = 0x06000000,
WHEEL           = 0x07000000,
```

```
SYSTEM          = 0x10000000,
VENDOR          = 0x20000000,
```

Adding vendor properties

- The vehicle OEM should map vehicle-specific CAN signals to vehicle properties in the VENDOR group ...
- ... by creating an **extension** to `types.hal` (part of IVehicle)

```
package vendor.example.automotive.vehicle@2.0;

import android.hardware.automotive.vehicle@2.0::VehicleProperty;
import android.hardware.automotive.vehicle@2.0::VehiclePropertyGroup;
import android.hardware.automotive.vehicle@2.0::VehiclePropertyType;
import android.hardware.automotive.vehicle@2.0::VehicleArea;

enum VehicleProperty : android.hardware.automotive.vehicle@2.0::VehicleProperty {
    VENDOR_EXAMPLE = (
        0x0101
        | VehiclePropertyGroup:VENDOR
        | VehiclePropertyType:FLOAT
        | VehicleArea:GLOBAL),
};
```

The Car Service

- **Car Service** provides APIs for car applications, based on vehicle properties and other information, including:

Manager	Description
CarAudioManager	car audio, including group volumes, external sources, patches, balance and fade
CarBluetoothManager	Provides an API to interact with Car specific Bluetooth Device Management
CarDiagnosticManager	API for monitoring car diagnostic data, OBD2 diagnostic freeze and live frames
CarDrivingStateManager	Returns driving state: Parked, Idling, or Moving
CarInfoManager	static information from car (VID, model, year, fuel type, etc.)
CarPropertyManager	Wrapper for Vehicle properties
CarUxRestrictionsManager	Indicates whether there is a requirement to be Distraction Optimized? Uses information from CarDrivingStateManager

Accessing properties from apps

- Apps can use **CarPropertyManager** to access vehicle properties

Getting properties

```
boolean getBooleanProperty(int prop, int area)
float getFloatProperty(int prop, int area)
int getIntProperty(int prop, int area)
```

Setting properties

```
void setBooleanProperty(int prop, int areaId, boolean val)
void setFloatProperty(int prop, int areaId, float val)
void setIntProperty(int prop, int areaId, int val)
```

Registering a callback to be notified when a property changes:

```
boolean registerCallback(@NonNull CarPropertyEventCallback callback,
            int propertyId, @FloatRange(from = 0.0, to = 100.0) float rate)
```

OEM vehicle applications

- The OEM will implement a suite of applications that interface with the vendor vehicle properties
  - Pre-installed
  - Signed with the platform keys
  - Privileged

Android app permissions

- Android apps need to be granted **permissions** to access services
- The risk of granting a permission is set by the **protection level**

normal	grant at install-time without prompting
dangerous	prompt user before granting
signature	grant if signature of app requesting and app declaring the perm match
privileged	(also called "system"): grant only to privileged system apps

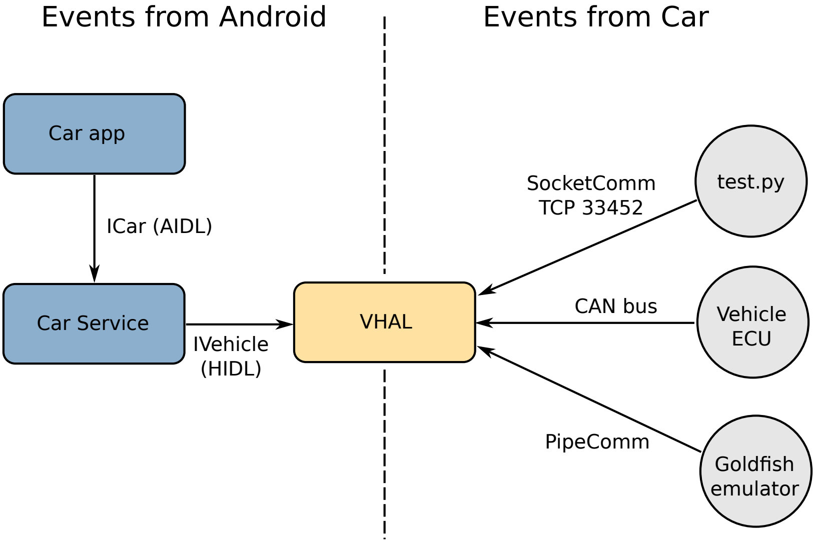
Android permissions for vehicle properties

- Car Service defines over 100 permissions for Car applications
- Only 8 can be granted to 3rd party apps (i.e. normal or dangerous)

```
CAR_INFO
READ_CAR_DISPLAY_UNITS
CONTROL_CAR_DISPLAY_UNITS
CAR_ENERGY_PORTS
CAR_EXTERIOR_ENVIRONMENT
CAR_POWERTRAIN
CAR_SPEED
CAR_ENERGY
```

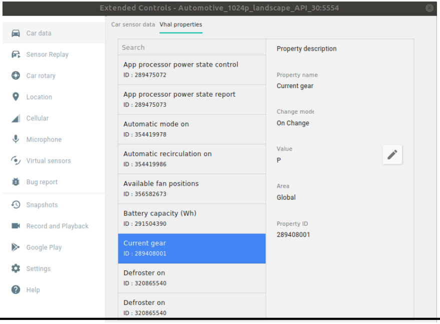
- The others are marked as **signature | privileged**
  - which are only granted to apps built by the OEM and shipped as part of the platform

Testing the VHAL



SocketComm and PipeComm

- SocketComm and PipeComm are part of the default VHAL daemon
- Both allow vehicle property reads and writes to be injected **as if they came from the car**
  - SocketComm: via TCP socket, port 33452
  - PipeComm: qemu pipe, used in Goldfish emulator:



Extended Controls - Automotive_1024p_landscape_API_30:5554

App processor power state control
ID : 289475072

App processor power state report
ID : 289475073

Automatic mode on
ID : 354419978

Automatic recirculation on
ID : 354419986

Available fan positions
ID : 356582673

Battery capacity (Wh)
ID : 291504390

Current gear
ID : 289408001

Defroster on
ID : 320865540

Defroster on
ID : 320865540

Property description

Property name
Current gear

Change mode
On Change

Value
P

Area
Global

Property ID
289408001

Example: read a property

read-prop-example.py

```
#!/usr/bin/env python

import vhal_consts_2_0 as c
from vhal_emulator import Vhal

if __name__ == '__main__':
    v = Vhal(c.vhal_types_2_0)
    v.getProperty(c.VEHICLEPROPERTY_ENV_OUTSIDE_TEMPERATURE, c.VEHICLEAREA_GLOBAL)
    reply = v.rxMsg()
    print(reply)
```

Testing

```
$ ./read-prop-example.py
Connecting local port 33005 to remote port 33452 on default device
msg_type: GET_PROPERTY_RESP
status: RESULT_OK
value {
  prop: 291505923
  value_type: 6291456
  timestamp: 18211762823719
  area_id: 0
  float_values: 25.0
  status: AVAILABLE
}
```

Conclusion

- Signals from the CAN bus are mapped to vehicle properties
- The VHAL implements IVehicle interface to Car Service; apps call Car Service
- To extend, OEM can
- add properties in the VENDOR group
- implement link from VHAL to CAN bus
- implement permissions

Slides at https://2net.co.uk/slides/aaos-vhal-csimmonds-ew-2022.pdf

Embedded Android+Automotive: a 5-day deep dive into Android Automotive
https://2net.co.uk/training/embedded-android-automotive

"Looking after the Inner Penguin" blog at http://2net.co.uk/

@2net_software

https://uk.linkedin.com/in/chrisdsimmonds/