Android OEM VHAL

Hands On Workshop

Jan Kubovy, BMW

Agenda

- Introduction & overview
- Source code & AOSP build
- Workspace setup

How to get Car Data to Apps: Classification

Capability coverage:
How much of the vehicle's capabilities are covered?

Extension speed:
How fast can new datapoints be added? If there is a new function or data point in the vehicle, how fast can you extend your API or data model?

Maintenance:
How much effort is needed for maintenance for the company which employes it?

Standard:
Is it a standard API and data model or a proprietary one?

Backward compatibility:
Is there a mechanism in place to ensure backward compatibility?

Data preparation:
Is the data already prepared for the application use or do applications need to deal with raw vehicle data?

Usage requirements:
How much effort is needed to be able to use the API? This includes availability of documentation, strong community or company behind it, acceptance in the industry, etc.

3rd Parties:
Is it usable by 3rd Parties? Is there a documentation and technical enablement, motivation to 3rd parties to use it?

How to get Car Data to Apps: Options

	In house solution		Standard API	AOSP VHAL		
	Custom Solution	Vendor API		Vendor Properties	System Properties	OEM Properties
Capability Coverage	High	High	High	High	Low	**High**
Extension speed	Fast	Fast*1	Slower	Fast	Slow	**Slower**
Maintenance	High	High	Low	Middle	Low	**Low**
Standard	Proprietary	Proprietary	Standard	Proprietary	Standard	**Standard**
Backward compatibility	No	Possible	Possible	Partial	Yes	**Yes**
Data preparation	Application	Possible	Yes	Yes	Yes	**Yes**
Usage requirements	High	High	Middle*2	Low	Low	**Low**
3rd Parties	No	No	Possible*2	No	Yes	**Yes**

1) Related to Maintenance, Backward compatibility and Usage Requirements.
2) Depends on acceptance, establishment of the standard API.

Android VHAL properties

- is identified by a unique **ID** (**`0xGATTDDDD`**)
  - **G**roup: 1 nibble
  - **A**rea: 1 nibble
  - **T**ype 1 byte
  - i**D**entifier: 2 bytes - as sequence (max 65535!)
    - Google starts at 0x0100 (VIN) in Group 0x1
    - **OEM properties use Group 0x4**
- has associated metadata
- access type (read, write)
- …

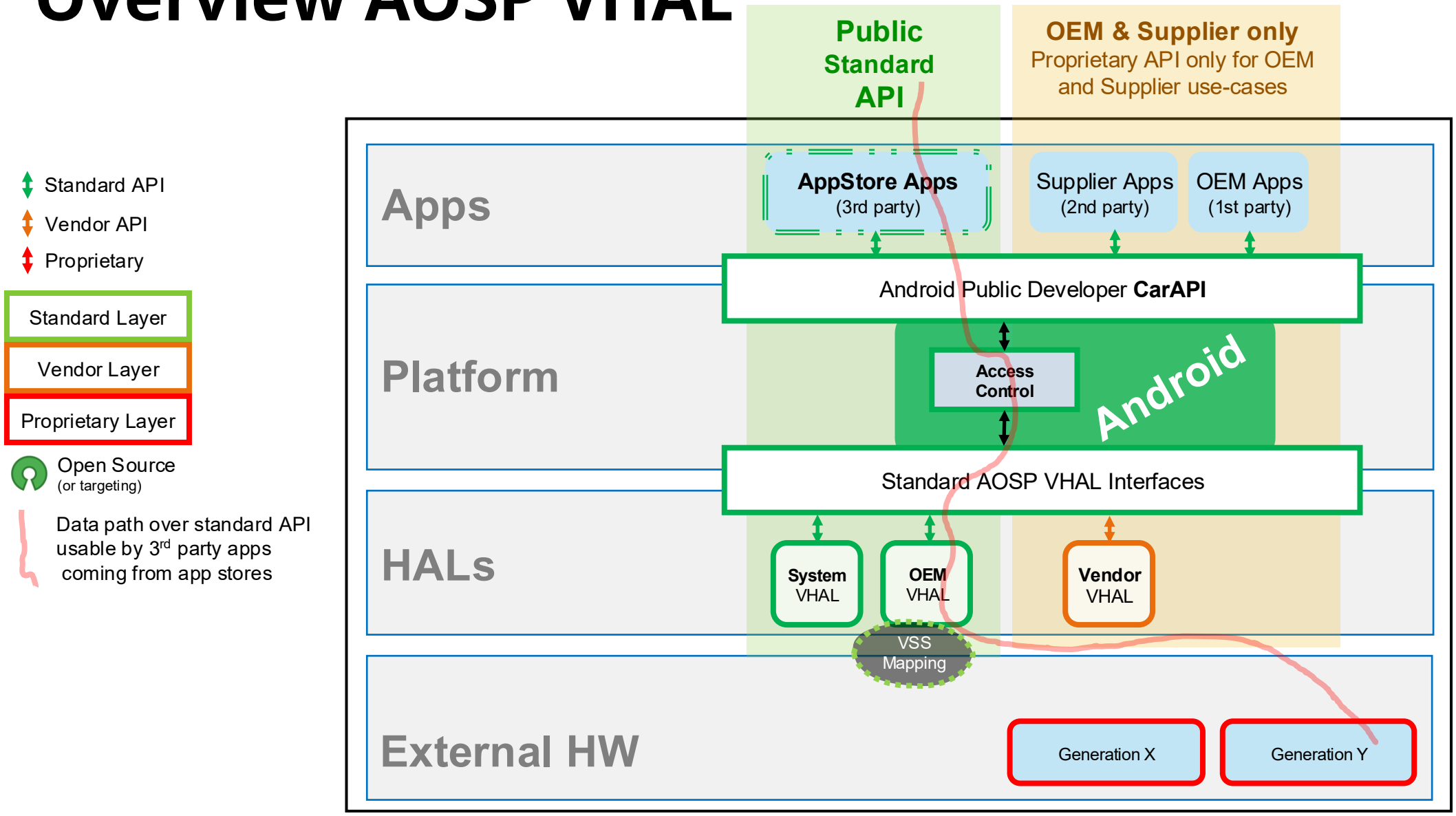
```
enum VehiclePropertyGroup {
    SYSTEM      = 0x10000000,
    VENDOR      = 0x20000000,
    BACKPORTED  = 0x30000000,
+   OEM         = 0x40000000, // Added
    MASK        = 0xf0000000,
}
```

```
enum VehiclePropertyType {
    STRING     = 0x00100000,
    BOOLEAN    = 0x00200000,
    INT32      = 0x00400000,
    INT32_VEC  = 0x00410000,
    INT64      = 0x00500000,
    INT64_VEC  = 0x00510000,
    FLOAT      = 0x00600000,
    FLOAT_VEC  = 0x00610000,
    BYTES      = 0x00700000,
    MIXED      = 0x00e00000,
    MASK       = 0x00ff0000,
}
```

```
/**
 * Declares all vehicle properties. VehicleProperty has a bitwise structure.
 * Each property must have:
 *   - a unique id from range 0x0100 - 0xffff
 *   - associated data type using VehiclePropertyType
 *   - property group (VehiclePropertyGroup)
 *   - vehicle area (VehicleArea)
 *
 * Vendors are allowed to extend this enum with their own properties. In this
 * case they must use VehiclePropertyGroup:VENDOR flag when the property is
 * declared.
 *
 * When a property's status field is not set to AVAILABLE:
 *   - IVehicle#set may return StatusCode::NOT_AVAILABLE.
 *   - IVehicle#get is not guaranteed to work.
 *
 * Properties set to values out of range must be ignored and no action taken
 * in response to such ill formed requests.
 */
```

Source:
https://cs.android.com/android/platform/superproject/main/+/main:hardware/interfaces/automotive/vehicle/aidl_property/android/hardware/automotive/vehicle/VehicleProperty.aidl

Overview AOSP VHAL



Source Code & AOSP Build

How to build: Repos

Manifest repo: https://github.com/COVESA/aosp_platform-manifest/tree/android-15 (as of April 2026)

AOSP extension

- Socket vehicle hardware implementation: https://github.com/COVESA/aosp_device_generic_car
- OEM extension: https://github.com/COVESA/aosp_platform_packages_services_Car
- OEM extension & generated code: https://github.com/COVESA/aosp_platform_hardware_interfaces
- Config files for emulator: https://github.com/COVESA/aosp_device_google_cuttlefish
- OEM property feature: https://github.com/COVESA/aosp_build_release

COVESA AOSP extensions

- Permission proxy service: https://github.com/COVESA/aosp_packages_services_oem
- Generated OEM properties and permissions: https://github.com/COVESA/aosp_vendor_car

Mock VHAL server: https://github.com/COVESA/aosp_vhal_server_mock

Test App: https://github.com/COVESA/aosp_vhal_oem_test_app

How to build: Checkout, Build & Run

- Follow: https://github.com/COVESA/aosp_platform-manifest/tree/android-15

- Before build see: https://github.com/COVESA/aosp_device_generic_car/tree/oem-vhal/emulator/vhal_sockets_vehicle_hardware/external

- Before first run:

$ adb shell dpm set-profile-owner --user 10 global.covesa.vhal.mapper.permission.proxy/.CarDeviceAdminReceiver

Workspace Setup

Emulator

- Download Emulator https://emulator.covesa.global/sdk/emulators/covesa-sys-img2-1.xml

- Unzip into `${ANDROID_SDK_ROOT}/system-images-android-35/android-automotive/`:

  - Windows: `C:\Users\<YourUser>\AppData\Local\Android\Sdk`
  - Mac: `~/Library/Android/sdk`
  - Linux: `~/Android/Sdk`

- Create `package.xml`

- Restart Android Studio

Mock Server

- Checkout: https://github.com/COVESA/aosp_vhal_server_mock
- Build:

  $ docker build -t node-server-docker .

- Run:

  $ docker run --name ws-server --rm --publish 8126:8126 node-server-docker

Run Test App

- Checkout: https://github.com/COVESA/aosp_vhal_oem_test_app
- Open it in Android Studio
- Gradle sync + build
- Run in Emulator

Grant and revoke permissions

*$ adb shell pm **grant/revoke** –user **{user_id} {package} {permission}***

- Grant permissions:

$ adb shell pm grant --user 10 global.covesa.aosp.vhal.test.app android.car.permission.oem.CABIN_SUNROOF_SHADE_IS_OPEN_READ

$ adb shell pm grant --user 10 global.covesa.aosp.vhal.test.app android.car.permission.oem.CABIN_SUNROOF_SHADE_IS_OPEN_WRITE

- Revoke permissions

$ adb shell pm revoke --user 10 global.covesa.aosp.vhal.test.app android.car.permission.oem.CABIN_SUNROOF_SHADE_IS_OPEN_READ

$ adb shell pm revoke --user 10 global.covesa.aosp.vhal.test.app android.car.permission.oem.CABIN_SUNROOF_SHADE_IS_OPEN_WRITE

Thank You!

Visit: www.covesa.global

Contact Us: help@covesa.global