# Unleashing the Software Defined Vehicle with VSS and Virtualization

Dr. Bernd Hardung

26.09.2024

© Elektrobit 2023

# The Levels of Software Defined Vehicles

Elektrobit

| | SDV LEVEL 0<br>Software-enabled | SDV LEVEL 1<br>Connected | SDV LEVEL 2<br>Updateable | SDV LEVEL 3<br>Upgradeable | SDV LEVEL 4<br>Software Platform | SDV LEVEL 5<br>Innovation Platform |
|---|---|---|---|---|---|---|
| **Customer experience** | **Mechanical functions made possible via software** | **Information and control via connectivity** | **Vehicle maintenance via OTA updates** | **New functions via software upgrades** | **Car always feels new via cross-generation software upgrades** | **Full customization via ecosystem of 3rd party** |
| **Scope of adaptability and dynamics** | • No Updates<br>• No connectivity | • Static software<br>• Dynamic environment | • Static functionality<br>• Dynamic software | • Dynamic functionality<br>• Static target hardware | • Dynamic target hardware<br>• Singular software provider | • Diverse software providers |
| **Examples** | • Parking assitance<br>• Adaptive cruise control | • Live traffic information<br>• Mobile phone companion | • Navigation update<br>• Security patches | • New function in a single car line | • New function delivered to all vehicle generations | • New 3rd party function deployed to a multi-brand fleet |

**Optional Domain Restriction** — **Example: SDV Level 4 for cockpit domain**

# The Levels of Software Defined Vehicles

Elektrobit

| | SDV LEVEL 0<br>Software-enabled | SDV LEVEL 1<br>Connected | SDV LEVEL 2<br>Updateable | SDV LEVEL 3<br>Upgradeable | SDV LEVEL 4<br>Software Platform | SDV LEVEL 5<br>Innovation Platform |
|---|---|---|---|---|---|---|
| **Customer experience** | **Mechanical functions made possible via software** | **Information and control via connectivity** | **Vehicle maintenance via OTA updates** | **New functions via software upgrades** | **Car always feels new via cross-generation software upgrades** | **Full customization via ecosystem of 3rd party** |
| **Scope of adaptatibility and dynamics** | • No Updates<br>• No connectivity | • Static software<br>• Dynamic environment | • Static functionality<br>• Dynamic software | • Dynamic functionality<br>• Static target hardware | • Dynamic target hardware<br>• Singular software provider | • Diverse software providers |
| **Examples** | • Parking assitance<br>• Adaptive cruise control | • Live traffic information<br>• Mobile phone companion | • Navigation update<br>• Security patches | • New function in a single car line | • New function delivered to all vehicle generations | • New 3rd party function deployed to a multi-brand fleet |

**Optional Domain Restriction** — **Example: SDV Level 4 for cockpit domain**

What is the fast-path?

## Typical Enablers

| | SDV LEVEL 0 | SDV LEVEL 1 | SDV LEVEL 2 | SDV LEVEL 3 | SDV LEVEL 4 | SDV LEVEL 5 |
|---|---|---|---|---|---|---|
| **EE architecture and hardware enablers** (cumulative across levels) | • Microcontroller-based ECUs<br>• In-vehicle networks | • Central connectivity unit<br>• Mobile connection<br>• Microprocessor-based ECUs<br>• Infotainment system with display and touch control | • A/B memory on ECUs<br>• Limited over-allocation of hardware resources | • Zonal architectures with reduced number of ECUs | • Reduction of hardware variants across vehicles and vehicle generations | |
| **Software architecture enablers** (cumulative across levels) | | • High-level operating systems<br>• Internet communication<br>• Layered security architecture | • In-vehicle update verification<br>• Dynamic operating systems<br>• Partially updateable ECU | • Service-oriented architectures<br>• Virtualization | • SDK with version and variant handling<br>• Hardware-independent applications | • Strong isolation technology within ECUs<br>• In-ECU intrusion detection<br>• Data-access/Device-access |
| **Development practices enablers** (cumulative across levels) | | • Convergence of internet technology and in-vehicle software | | • Automated CI/CT/CD environments<br>• Virtual testing | • Controlled deprecation of software and API versions | • Automated security scanning and vetting process of functions |
| **Business enablers** (cumulative across levels) | | • Subscription for long-term use of services | • Maintenance contracts with ECU suppliers | • Aftermarket revenue generation for new functions<br>• OEMs directly sourcing software or own development | • Decoupling of vehicle and software platform business cases | • Monetization possibilities for 3rd party function providers |

Author: Dr. Moritz Neukirchner

# How to fast forward to Level 5

- Restrict use-cases to limit complexity, e.g. exclude real-time and safety
- Standardize deployment format so that applications are hardware-agnostic and OEM-agnostic
- Standardize data semantics so that applications are OEM-agnostic
- Standardize API to access vehicle data
- Establish scalable development environment that does not rely upon hardware

© Elektrobit 2024 | Confidential information

# How to fast forward to Level 5

- Restrict use-cases to limit complexity, e.g. exclude real-time and safety
- Standardize deploym nostic and OEM-agnostic
- Standardize data sem
- Standardize API to ac
- Establish scalable development environment that does not rely upon hardware

Approach for all:

Build on solutions that are already usable in productive environments

Don‘t duplicate what‘s already there

# Getting concrete

**Restrict** to applications that can reasonably be executed in a **cockpit HPCs**

Use **Android apk** – well-established portable deployment format

Use **VSS and Android VHAL** for data semantics – commonly used and standardized by a community

Use Android **VHAL API** for access – no new API implementation required

Enable **AAOS development**, debug, and test in a cloud setup

© Elektrobit 2024 | Confidential information

# Vision: Virtual IVI Development for Android™

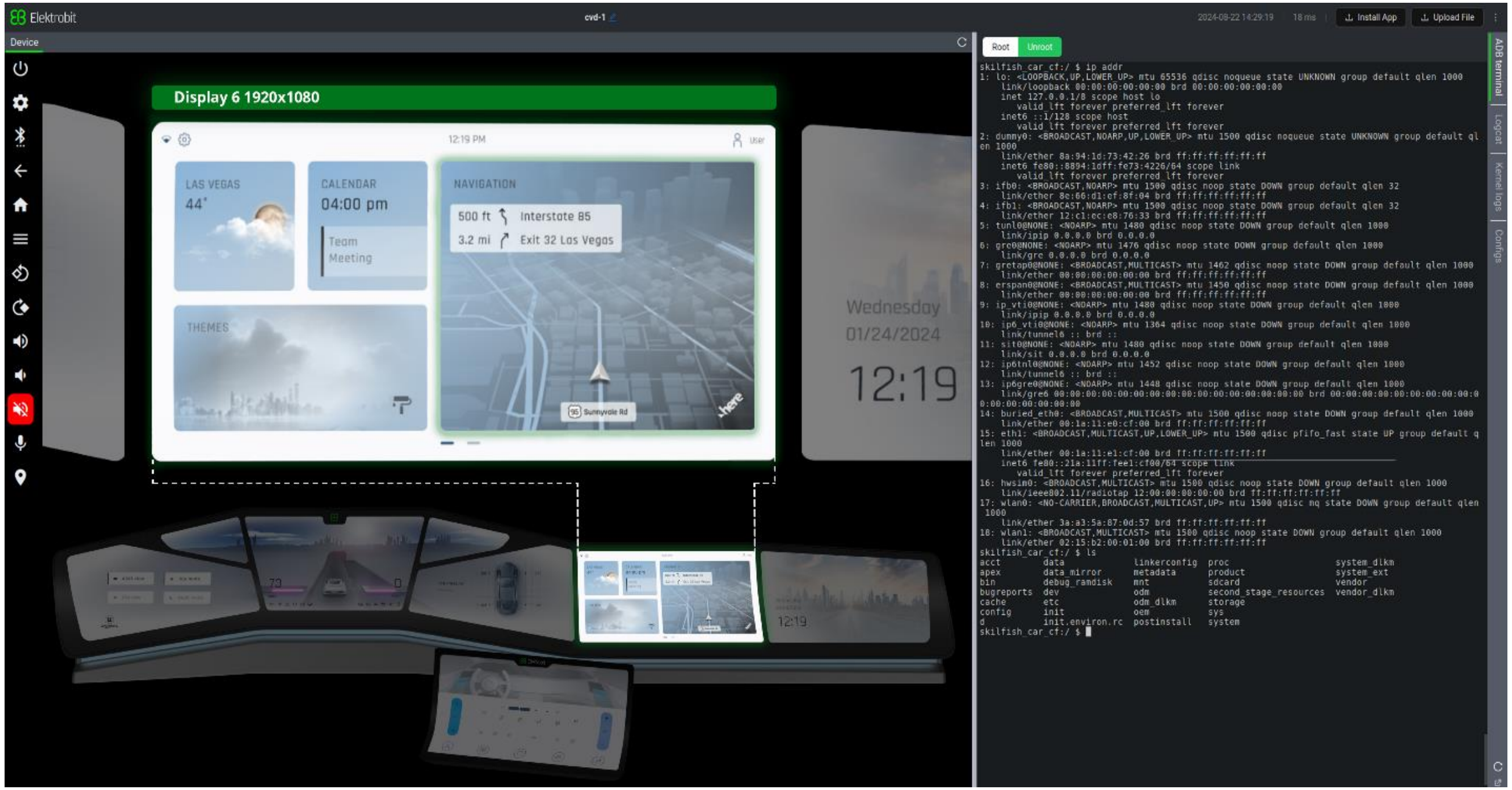
„Software Defined Vehicle“ - Define your vehicle value with your Software!

**Replicate** your screen topology in the virtual cockpit

**Abstract** vehicle data for application developers to create true platform portability

**Run** the same Android apps in the virtual and physical cockpit via standard apk deployment

**Available** for ‘On-Premise‘ or cloud infrastructure

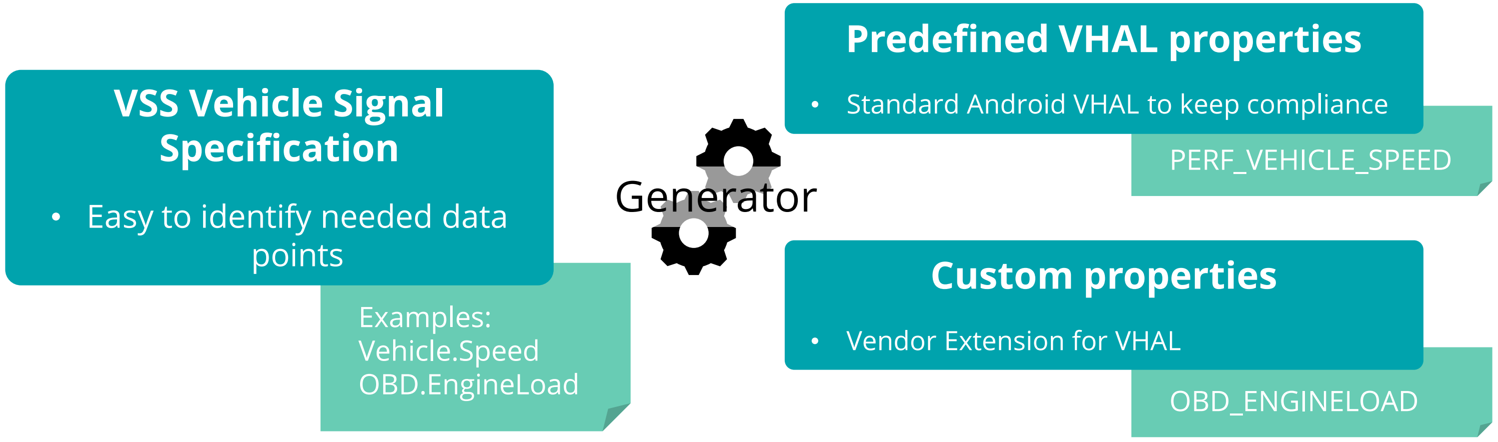


```
skilfish_car_cf:/ $ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: dummy0: <BROADCAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN group default ql
en 1000
    link/ether 8a:94:1d:73:42:26 brd ff:ff:ff:ff:ff:ff
    inet6 fe80::8894:1dff:fe73:4226/64 scope link
       valid_lft forever preferred_lft forever
3: ifb0: <BROADCAST,NOARP> mtu 1500 qdisc noop state DOWN group default qlen 32
    link/ether 8e:66:d1:cf:8f:04 brd ff:ff:ff:ff:ff:ff
4: ifb1: <BROADCAST,NOARP> mtu 1500 qdisc noop state DOWN group default qlen 32
    link/ether 12:c1:ec:e8:76:33 brd ff:ff:ff:ff:ff:ff
5: tunl0@NONE: <NOARP> mtu 1480 qdisc noop state DOWN group default qlen 1000
    link/ipip 0.0.0.0 brd 0.0.0.0
6: gre0@NONE: <NOARP> mtu 1476 qdisc noop state DOWN group default qlen 1000
    link/gre 0.0.0.0 brd 0.0.0.0
7: gretap0@NONE: <BROADCAST,MULTICAST> mtu 1462 qdisc noop state DOWN group default qlen 1000
    link/ether 00:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff
8: erspan0@NONE: <BROADCAST,MULTICAST> mtu 1450 qdisc noop state DOWN group default qlen 1000
    link/ether 00:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff
9: ip_vti0@NONE: <NOARP> mtu 1480 qdisc noop state DOWN group default qlen 1000
    link/ipip 0.0.0.0 brd 0.0.0.0
10: ip6_vti0@NONE: <NOARP> mtu 1364 qdisc noop state DOWN group default qlen 1000
    link/tunnel6 :: brd ::
11: sit0@NONE: <NOARP> mtu 1480 qdisc noop state DOWN group default qlen 1000
    link/sit 0.0.0.0 brd 0.0.0.0
12: ip6tnl0@NONE: <NOARP> mtu 1452 qdisc noop state DOWN group default qlen 1000
    link/tunnel6 :: brd ::
13: ip6gre0@NONE: <NOARP> mtu 1448 qdisc noop state DOWN group default qlen 1000
    link/gre6 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00 brd 00:00:00:00:00:00:00:00:00:0
0:00:00:00:00:00:00
14: buried_eth0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether 00:1a:11:e0:cf:00 brd ff:ff:ff:ff:ff:ff
15: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default q
len 1000
    link/ether 00:1a:11:e1:cf:00 brd ff:ff:ff:ff:ff:ff
    inet6 fe80::21a:11ff:fee1:cf00/64 scope link
       valid_lft forever preferred_lft forever
16: hwsim0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ieee802.11/radiotap 12:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff
17: wlan0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq state DOWN group default qlen
 1000
    link/ether 3a:a3:5a:87:0d:57 brd ff:ff:ff:ff:ff:ff
18: wlan1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether 02:15:b2:00:01:00 brd ff:ff:ff:ff:ff:ff
skilfish_car_cf:/ $ ls
acct        data             linkerconfig  proc                     system_dlkm
apex        data_mirror      metadata      product                  system_ext
bin         debug_ramdisk    mnt           sdcard                   vendor
bugreports  dev              odm           second_stage_resources   vendor_dlkm
cache       etc              odm_dlkm      storage
config      init             oem           sys
d           init.environ.rc  postinstall   system
skilfish_car_cf:/ $ █
```

# Exposing VSS via Android VHAL

Step 1
Select required properties

Step 2
Generate mapping to
Android VHAL or Vendor extension



**VSS Vehicle Signal Specification**

- Easy to identify needed data points

Examples:
Vehicle.Speed
OBD.EngineLoad

**Predefined VHAL properties**

- Standard Android VHAL to keep compliance

PERF_VEHICLE_SPEED

**Custom properties**

- Vendor Extension for VHAL

OBD_ENGINELOAD

## What the app developer sees

Step 3
Use out-of-the-box APIs and SDK
- Standard apk
- Standard APIs and SDK
- Standard permission scheme

**AAOS with enhanced VHAL**

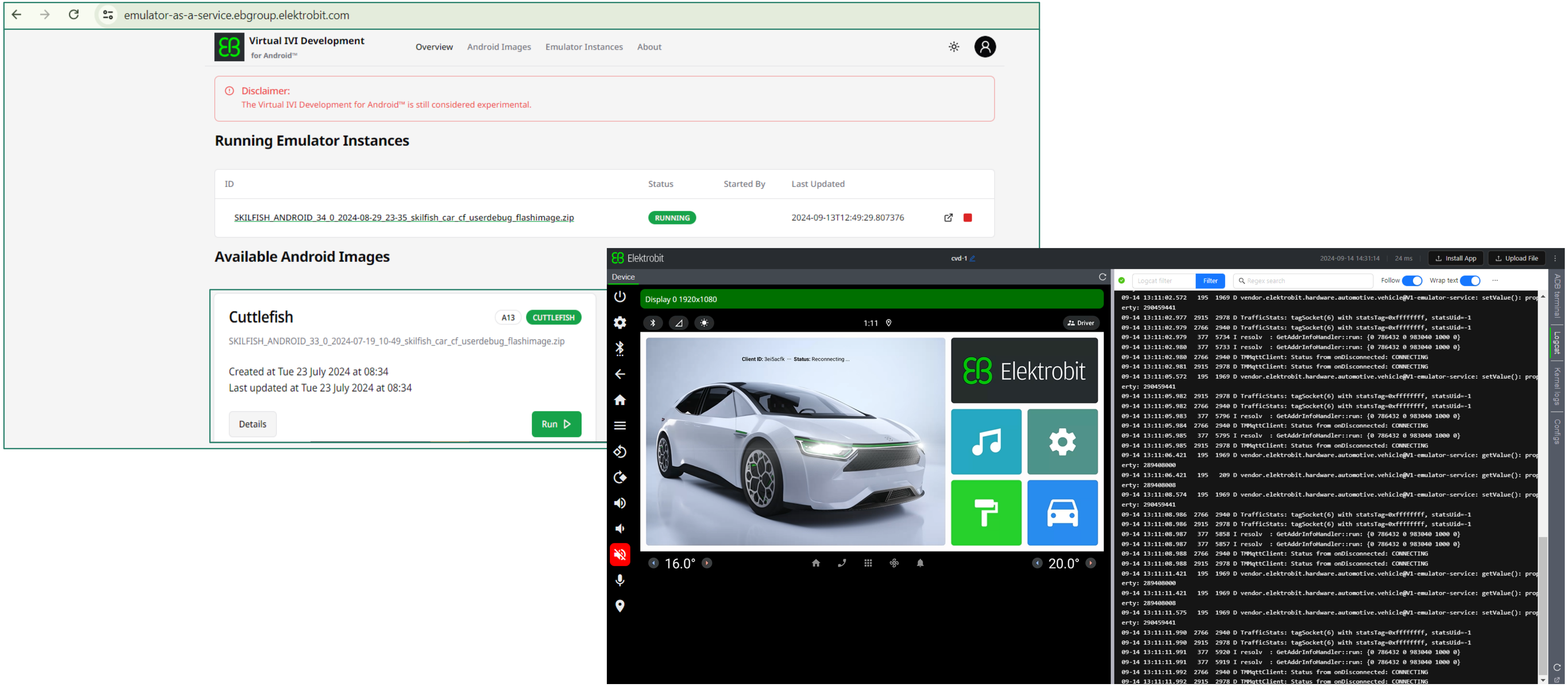
- Used in virtual environment
- Used on physical target hardware

Step 4
Use identical test tooling for VHAL stimulation on physical and virtual target

**ADB + vehicle simulation**

- Simulates VHAL properties
- Enables realistic virtual testing
- Use same tests on virtual and physical environment

# Virtual IVI Development for Android™



Disclaimer:
The Virtual IVI Development for Android™ is still considered experimental.

## Running Emulator Instances

| ID | Status | Started By | Last Updated |
|---|---|---|---|
| SKILFISH_ANDROID_34_0_2024-08-29_23-35_skilfish_car_cf_userdebug_flashimage.zip | RUNNING | | 2024-09-13T12:49:29.807376 |

## Available Android Images

**Cuttlefish** A13 CUTTLEFISH

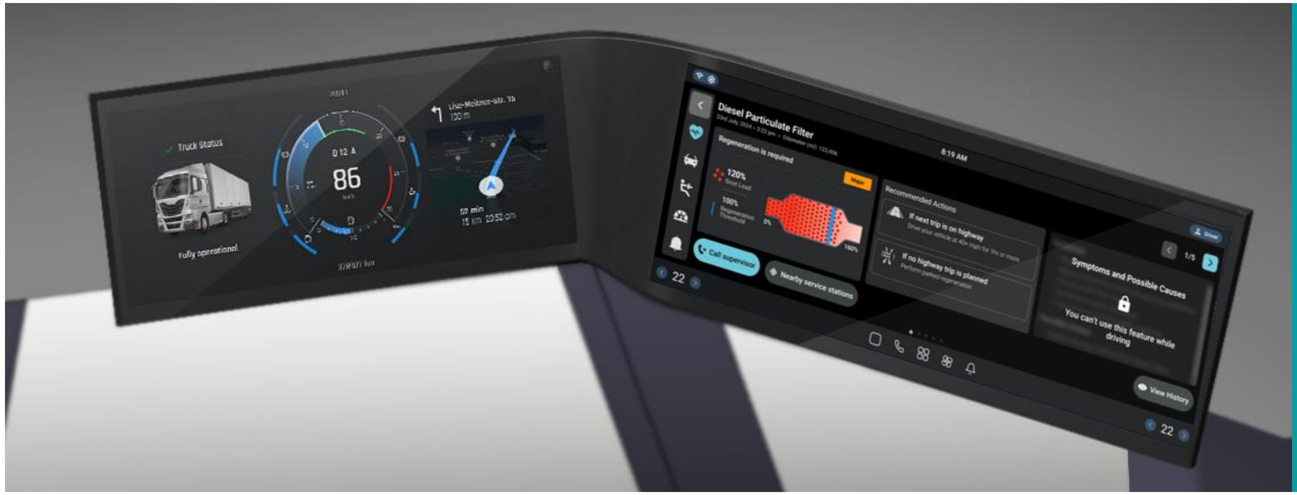
SKILFISH_ANDROID_33_0_2024-07-19_10-49_skilfish_car_cf_userdebug_flashimage.zip

Created at Tue 23 July 2024 at 08:34
Last updated at Tue 23 July 2024 at 08:34

Details Run

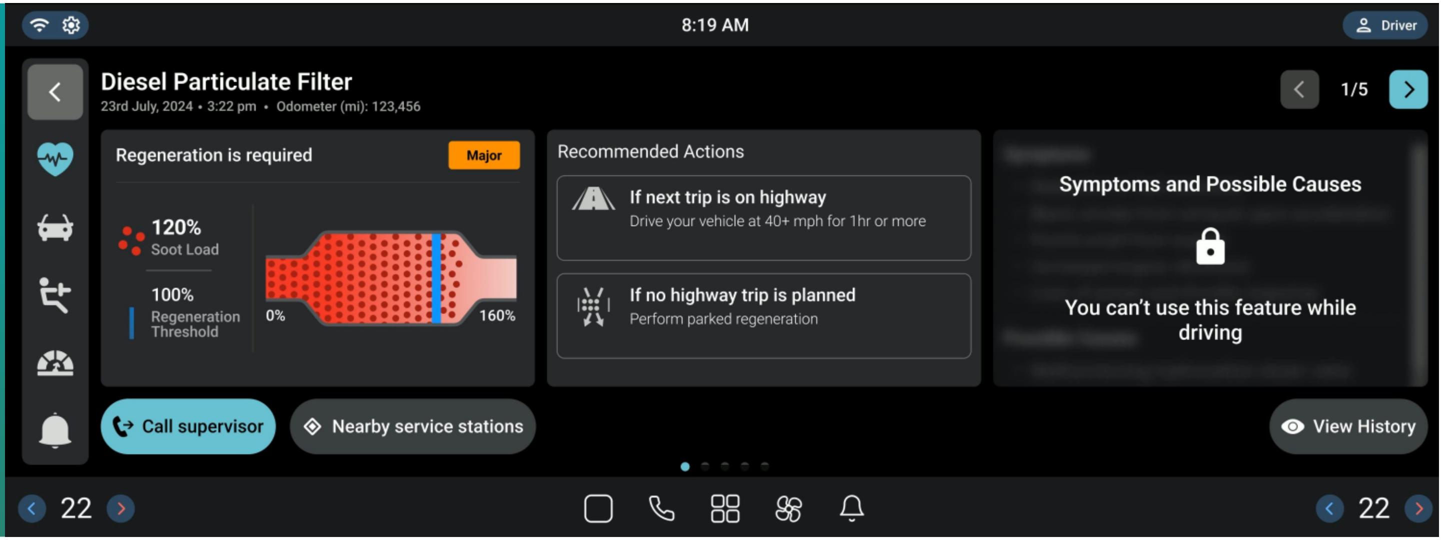
```
09-14 13:11:02.572   195  1969 D vendor.elektrobit.hardware.automotive.vehicle@V1-emulator-service: setValue(): prop
erty: 290459441
09-14 13:11:02.977  2915  2978 D TrafficStats: tagSocket(6) with statsTag=0xffffffff, statsUid=-1
09-14 13:11:02.979  2766  2940 D TrafficStats: tagSocket(6) with statsTag=0xffffffff, statsUid=-1
09-14 13:11:02.979   377  5734 I resolv  : GetAddrInfoHandler::run: {0 786432 0 983040 1000 0}
09-14 13:11:02.980   377  5733 I resolv  : GetAddrInfoHandler::run: {0 786432 0 983040 1000 0}
09-14 13:11:02.980  2766  2940 D TMMqttClient: Status from onDisconnected: CONNECTING
09-14 13:11:02.981  2915  2978 D TMMqttClient: Status from onDisconnected: CONNECTING
09-14 13:11:05.572   195  1969 D vendor.elektrobit.hardware.automotive.vehicle@V1-emulator-service: setValue(): prop
erty: 290459441
09-14 13:11:05.982  2915  2978 D TrafficStats: tagSocket(6) with statsTag=0xffffffff, statsUid=-1
09-14 13:11:05.982  2766  2940 D TrafficStats: tagSocket(6) with statsTag=0xffffffff, statsUid=-1
09-14 13:11:05.983   377  5796 I resolv  : GetAddrInfoHandler::run: {0 786432 0 983040 1000 0}
09-14 13:11:05.984  2766  2940 D TMMqttClient: Status from onDisconnected: CONNECTING
09-14 13:11:05.985   377  5795 I resolv  : GetAddrInfoHandler::run: {0 786432 0 983040 1000 0}
09-14 13:11:05.985  2915  2978 D TMMqttClient: Status from onDisconnected: CONNECTING
09-14 13:11:06.421   195  1969 D vendor.elektrobit.hardware.automotive.vehicle@V1-emulator-service: getValue(): prop
erty: 289408000
09-14 13:11:06.421   195   209 D vendor.elektrobit.hardware.automotive.vehicle@V1-emulator-service: getValue(): prop
erty: 289408008
09-14 13:11:08.574   195  1969 D vendor.elektrobit.hardware.automotive.vehicle@V1-emulator-service: setValue(): prop
erty: 290459441
09-14 13:11:08.986  2766  2940 D TrafficStats: tagSocket(6) with statsTag=0xffffffff, statsUid=-1
09-14 13:11:08.986  2915  2978 D TrafficStats: tagSocket(6) with statsTag=0xffffffff, statsUid=-1
09-14 13:11:08.987   377  5858 I resolv  : GetAddrInfoHandler::run: {0 786432 0 983040 1000 0}
09-14 13:11:08.987   377  5857 I resolv  : GetAddrInfoHandler::run: {0 786432 0 983040 1000 0}
09-14 13:11:08.988  2766  2940 D TMMqttClient: Status from onDisconnected: CONNECTING
09-14 13:11:08.988  2915  2978 D TMMqttClient: Status from onDisconnected: CONNECTING
09-14 13:11:11.421   195  1969 D vendor.elektrobit.hardware.automotive.vehicle@V1-emulator-service: getValue(): prop
erty: 289408000
09-14 13:11:11.421   195  1969 D vendor.elektrobit.hardware.automotive.vehicle@V1-emulator-service: getValue(): prop
erty: 289408008
09-14 13:11:11.575   195  1969 D vendor.elektrobit.hardware.automotive.vehicle@V1-emulator-service: setValue(): prop
erty: 290459441
09-14 13:11:11.990  2766  2940 D TrafficStats: tagSocket(6) with statsTag=0xffffffff, statsUid=-1
09-14 13:11:11.990  2915  2978 D TrafficStats: tagSocket(6) with statsTag=0xffffffff, statsUid=-1
09-14 13:11:11.991   377  5920 I resolv  : GetAddrInfoHandler::run: {0 786432 0 983040 1000 0}
09-14 13:11:11.991   377  5919 I resolv  : GetAddrInfoHandler::run: {0 786432 0 983040 1000 0}
09-14 13:11:11.992  2766  2940 D TMMqttClient: Status from onDisconnected: CONNECTING
09-14 13:11:11.992  2915  2978 D TMMqttClient: Status from onDisconnected: CONNECTING
```

# Example: Intangles predictive maintenance application



Physical

Virtual



8:19 AM

Driver

## Diesel Particulate Filter

23rd July, 2024 • 3:22 pm • Odometer (mi): 123,456

1/5

Regeneration is required — Major

120% Soot Load

100% Regeneration Threshold

0% 160%

Recommended Actions

If next trip is on highway
Drive your vehicle at 40+ mph for 1hr or more

If no highway trip is planned
Perform parked regeneration

Symptoms and Possible Causes

You can't use this feature while driving

Call supervisor

Nearby service stations

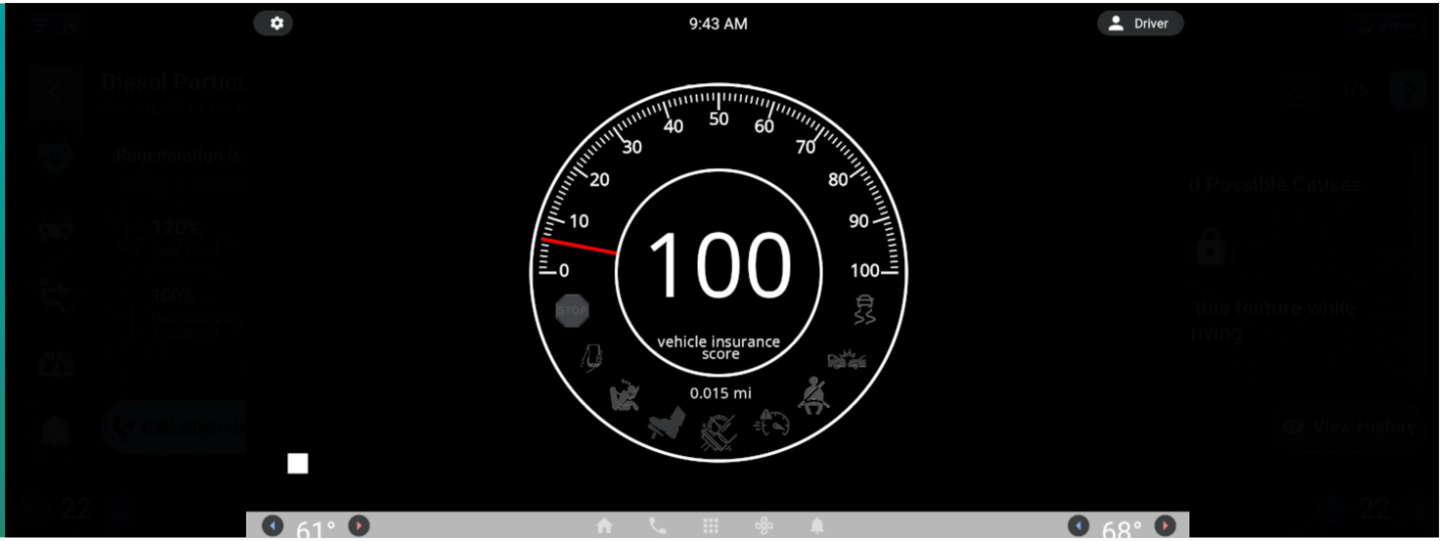
View History

22 22

# Example: MOTER technologies vehicle insurance score

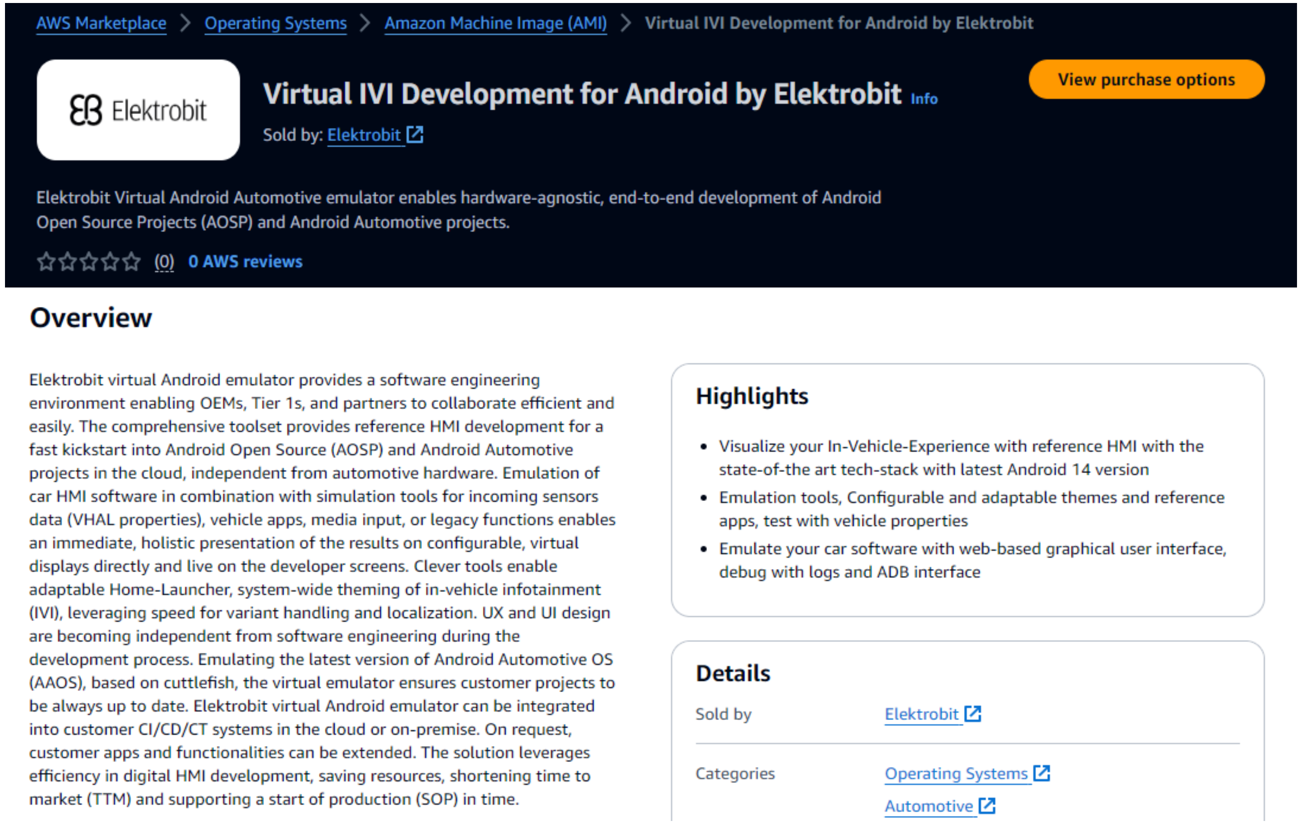
Physical

Virtual



# Virtual IVI Development on AWS Marketplace

Private-Beta is now available now on AWS Marketplace – you can register today



AWS Marketplace > Operating Systems > Amazon Machine Image (AMI) > Virtual IVI Development for Android by Elektrobit
Elektrobit
View purchase options

## Virtual IVI Development for Android by Elektrobit Info

Sold by: Elektrobit

Elektrobit Virtual Android Automotive emulator enables hardware-agnostic, end-to-end development of Android Open Source Projects (AOSP) and Android Automotive projects.

☆☆☆☆☆ (0) 0 AWS reviews

### Overview

Elektrobit virtual Android emulator provides a software engineering environment enabling OEMs, Tier 1s, and partners to collaborate efficient and easily. The comprehensive toolset provides reference HMI development for a fast kickstart into Android Open Source (AOSP) and Android Automotive projects in the cloud, independent from automotive hardware. Emulation of car HMI software in combination with simulation tools for incoming sensors data (VHAL properties), vehicle apps, media input, or legacy functions enables an immediate, holistic presentation of the results on configurable, virtual displays directly and live on the developer screens. Clever tools enable adaptable Home-Launcher, system-wide theming of in-vehicle infotainment (IVI), leveraging speed for variant handling and localization. UX and UI design are becoming independent from software engineering during the development process. Emulating the latest version of Android Automotive OS (AAOS), based on cuttlefish, the virtual emulator ensures customer projects to be always up to date. Elektrobit virtual Android emulator can be integrated into customer CI/CD/CT systems in the cloud or on-premise. On request, customer apps and functionalities can be extended. The solution leverages efficiency in digital HMI development, saving resources, shortening time to market (TTM) and supporting a start of production (SOP) in time.

### Highlights

- Visualize your In-Vehicle-Experience with reference HMI with the state-of-the art tech-stack with latest Android 14 version
- Emulation tools, Configurable and adaptable themes and reference apps, test with vehicle properties
- Emulate your car software with web-based graphical user interface, debug with logs and ADB interface

### Details

| | |
|---|---|
| Sold by | Elektrobit |
| Categories | Operating Systems<br>Automotive |

Register for the private beta

# Contact us

## Dr. Bernd Hardung

Vice President
Head of Business Solution Architecture

+49 172 7210590
bernd.hardung@elektrobit.com
elektrobit.com

© Elektrobit 2023