

The Automotive R&D Survival Guide: Decoupling Hardware Code in AAOS



Esmaeel Moradi
@esmaeel-moradi700



The Automotive R&D Survival Guide:

Decoupling Hardware Code in AAOS

**Why HAL, Kernel, and the Vendor Partition are Critical for
Future OTA Updates. (A Guide for AOSP Experts)**



Esmameil Moradi
@esmaeeil-moradi700



The Challenge: **Death to Tightly Coupled Code!**

If a company forks AOSP Google releases a major new version, our proprietary code must not break. This is the mission of Project Treble and VINTF.

Isolation:

Hardware-specific code (Vendor) must be 100% separate from the System Framework.

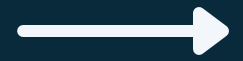
The only communication channel must be through Stable Interfaces (Stable AIDL).



Esmaeeil Moradi
@esmaeeil-moradi700

HAL:

The C/C++ Gateway to Vehicle Hardware



To port a proprietary sensor (e.g., a high-speed TPMS sensor) and ensure longevity:

1. Kernel/DT: Define the sensor's physical wiring, addresses, and interrupts in the Device Tree (DT). Write the Linux Kernel Driver.

2. Stable AIDL: Define the stable interface (e.g., `ti.tpms.ITirePressureSensor`) in hardware/interfaces.

3. Vendor Service: Implement the C++ HAL service and ensure it installs exclusively to the Vendor partition.

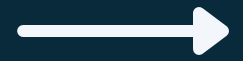
Result:

The Android Framework calls the standard AIDL interface, ensuring your hardware works, regardless of the System OS flavor.



Esmaeel Moradi
@esmaeel-moradi700

The Build System: Deploying and Authorizing Your HAL



Building the HAL isn't enough; we need to guarantee it launches on boot and has the correct permissions:

Build Config:

1. Use `PRODUCT_PACKAGES` in the device's `.mk` files to ensure the HAL binary installs to the Vendor partition.
2. **Init Service:** Create a specific `init.<name>.rc` file to automatically launch the HAL service (class `hal`) during boot.
3. **SELinux:** Define strict SELinux policies (`.te` files) to grant Binder permissions and explicit read/write access to the underlying kernel device node (e.g., `/dev/tpms_sensor`).



Esmaeel Moradi
@esmaeeil-moradi700



When C++ Crashes: Debugging in the Embedded Environment

Your Native HAL service crashes (SIGSEGV). We have no GUI and no standard IDE. Our tools are:

- 1. Logcat & debuggerd:** Monitor logcat for the initial crash message and raw stack trace generated by debuggerd.
- 2. Tombstone Analysis:** Pull the full crash file from `/data/tombstones/`.
- 3. Symbolization:** Use AOSP build tools (like symbolize) to map raw memory addresses (e.g., pc 0x...) back to the exact C++ function name and line number in your source code.



Esmaeel Moradi
@esmaeeil-moradi700

✦✦ Is your team prepared for
the next generation of AAOs?

Share your experience with
Vendor HAL challenges in the
comments! 👇



Esmaeeil Moradi
@esmaeeil-moradi700