**Android HAL Interface Definition Language (HIDL)**

**I. Introduction**

A new element of Android O is Project Treble. This is a major architectural change in the Android operating system framework designed to make it easier and faster for manufacturers to update devices to the new Android system at a lower cost.

Under the new architecture, the framework and hal run in different processes, and all HALs are implemented using new HIDL technology. As part of this change, devices running Android 8.0 must support a bound or straight-through HAL:

* Bind HAL. HAL in HAL Interface Definition Language (HIDL). These HALs replace the traditional HAL and legacy HAL used in earlier versions of Android. In a bound HAL, the Android framework communicates with the HAL through Binder Interprocess Communication (IPC) calls. All devices that ship with Android 8.0 or later at the time of launch must support only the bound HAL.
* Straight-through HAL. Traditional HAL or legacy HAL packaged in HIDL. These HALs encapsulate the existing HAL and are available in both bind mode and Same-Process mode. Devices that are upgraded to Android 8.0 can use a straight-through HAL.

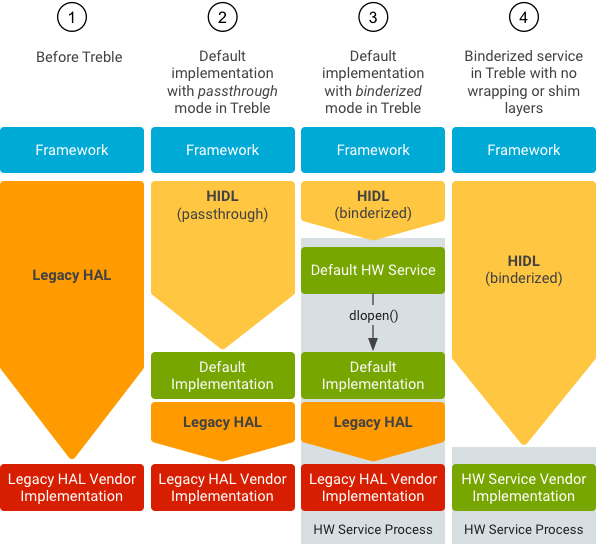


Figure 1: HIDL architecture overview

**II. Implement steps**

1. **Write driver to control Volume down button.**

Add new file: **fih\_btn\_test.c** to **android/kernel/msm**

**4.9/drivers/vendor\_name** folder

Modify Makefile: obj += **fih\_btn\_test.o**



1. **Write a .hal file**

We need to write a IVoltest.hal defines the interface that is called from Framework to HAL

First, create the ‘voltest’ directory in ‘android/vendor/vendor\_name/interfaces’ directory, then create a new folder named 1.0 in the voltest, and add the hal file in the 1.0 folder.

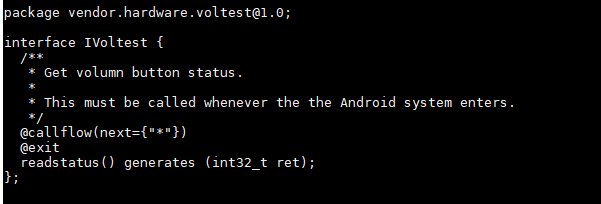


Figure 2: IVoltest.hal content.

1. **Write the main logic**

Create new default folder in 1.0 folder.

Write **Voltest.h** file and **Voltest.cpp** file, we will implement the method we just defined in the hal file in Voltest.cpp. code show as below:

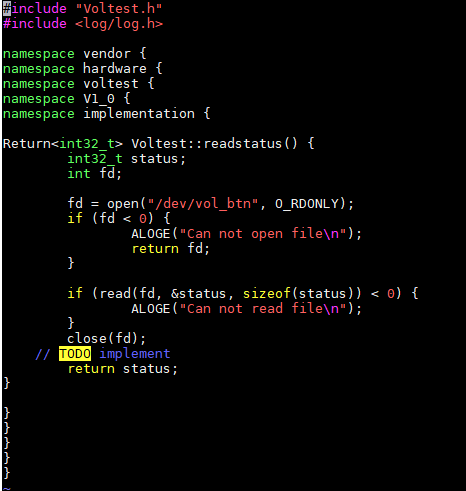


Figure 3: Voltest.cpp content

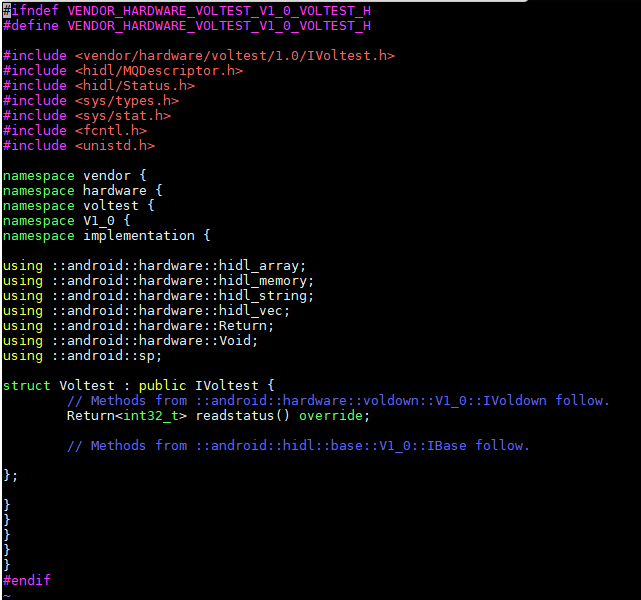


Figure 4: Voltest.h content

Create new Android.bp file to build above two files

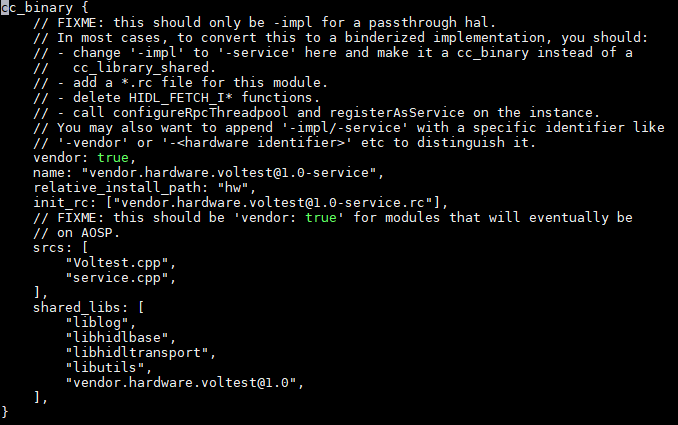


Figure 5: Android.bp content

Then there is the [vendor.hardware.voltest@1.0-service.rc](mailto:vendor.hardware.voltest@1.0-service.rc) startup script, the service.cpp entry.

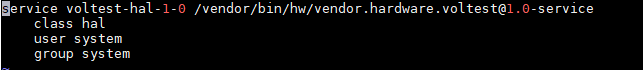


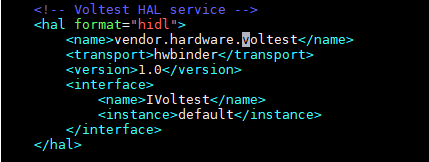
Figure 6: [vendor.hardware.voltest@1.0-service.rc](mailto:vendor.hardware.voltest@1.0-service.rc) content



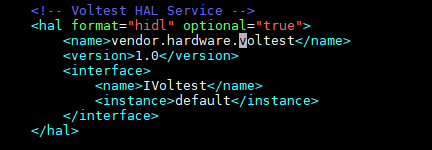
Figure 7: service.cpp content

The main code is finished. After that, we need to modify the configuration file so that we can write the hal

Modify the manifest.xml file, add the following code:



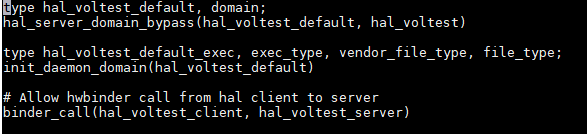
Modify the vendor\_framework\_compatibility\_matrix.xml file:



Modify **device/vendor\_name/sepolicy/file\_contexts** file to **automatic** load HIDL service on booting time.



Add new file ***hal\_voltest.te*** to ***device/vendor\_name/sepolicy*** folder



Modify **device/vendor\_name/sepolicy/hwservice\_contexts**



1. **Create HIDL java file.**

To create HIDL java file, we run below command:

**hidl-gen -o ./tmp –Ljava -rvendor.hardware:vendor/vendor\_name/interfaces -randroid.hidl:system/libhidl/transport** [**vendor.hardware.voltest@1.0**](mailto:vendor.hardware.voltest@1.0)

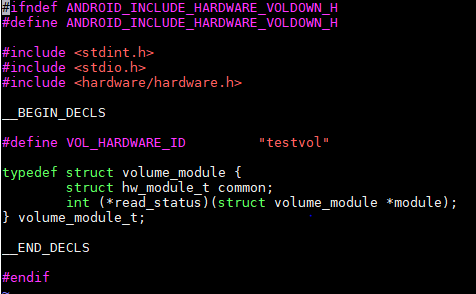
This java file will be called from application through binder.

Add following line to **device/fih\_qcom/fih\_sdm429\_64/device.mk** file to build this java



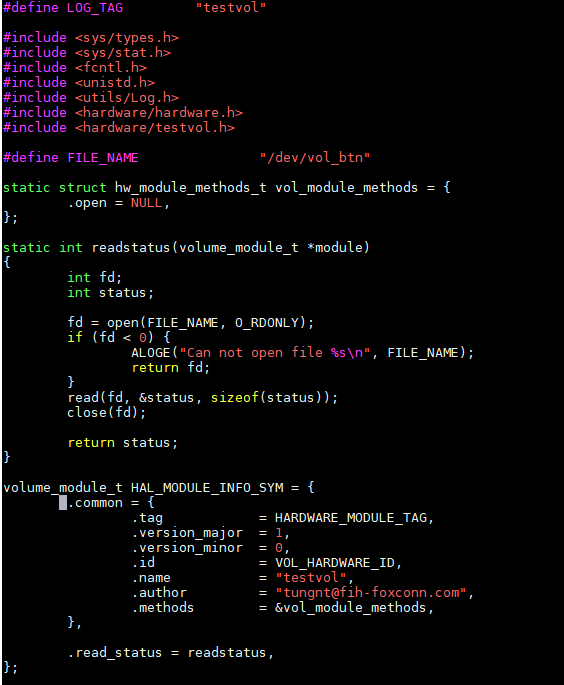
**HIDL AND HAL COMBINE (PASSTHROUGH)**

1. **HAL**
2. Header



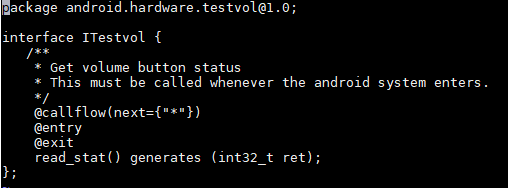
testvol.h

1. C file



testvol.c

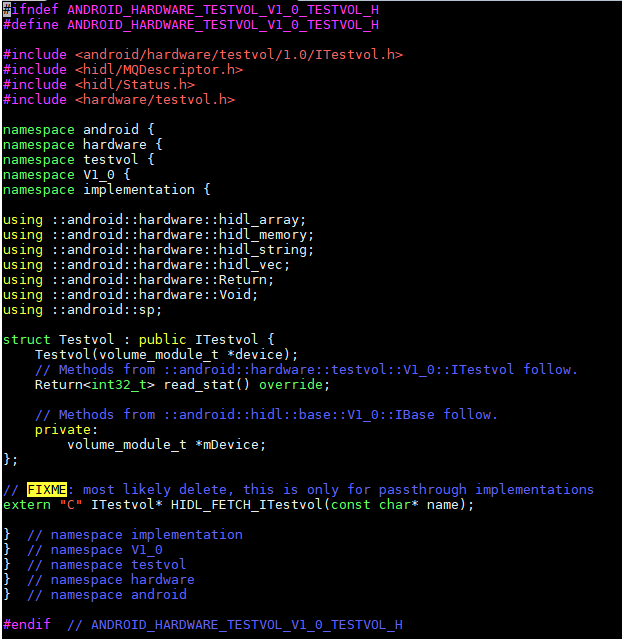
1. **HIDL**
2. Itestvol.hal



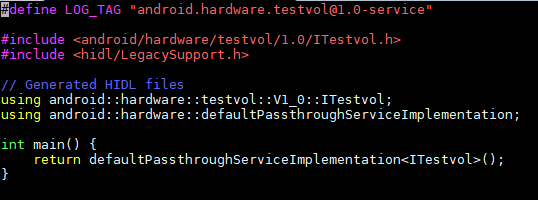
1. Testvol.cpp



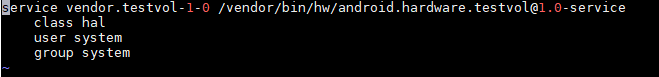
1. Testvol.h



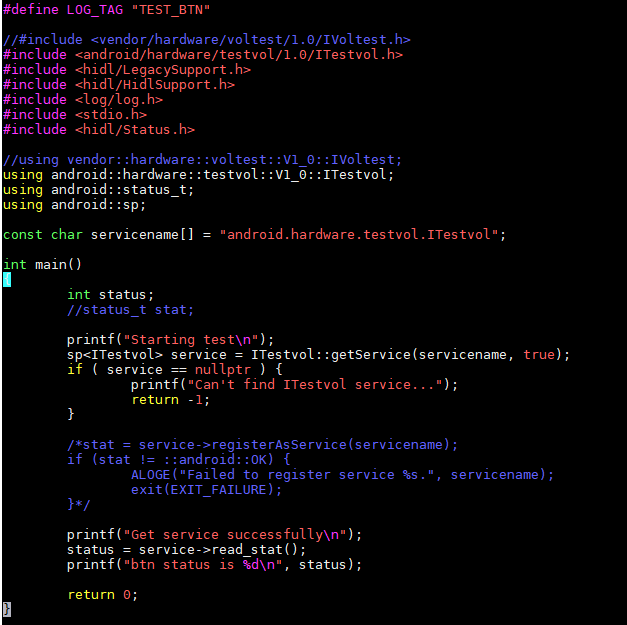
1. Service.cpp



1. android.hardware.testvol@1.0-service.rc



1. file to test hidl



Note: with passthrough

Android.bp will be

