Application Report
SPRACO0–August 2019

Enabling Android Automotive on Your TI Development Board

Joshua Allen Shafran and Praneeth Bajjuri

ABSTRACT

This application report walks you through the steps necessary to enable Android™ Automotive on your TI development board.

The Android Automotive OS and Human-Machine Interface (HMI) provide users with a simplified interface for accessing applications while on the road. The OS is an Android-based infotainment system that allows for a vehicle-optimized, stand-alone Android device to be placed inside the head unit. In place of the phone apps, users install driver-optimized versions of the app directly onto the console. This streamlined experience enables the driver to perform efficient tasks, such as climate control, maps, phone, audio, and so forth, in the convenience of their vehicle.

Contents

List of Figures

Trademarks

1 Introduction

This document shows the key differences between the standard (tablet) configuration and the Android Automotive interface. The look and feel of the applications will differ in the sense that they will be specially tuned towards an automotive experience. It also shows how to switch between the different target types (tablet and automotive) at build time. The AM57xx BeagleBoard-X15 Sitara platform was used since this is the only currently supported TI platform for development in the Android Open Source Project. The same work accomplished on the AM57xx BeagleBoard-X15 can be extended to a new platform; the AM65x EVM module is used as an example. Finally, the future work involved when enabling these features on a more automotive-tuned platform once it's available is discussed.

- By default, the TI development board will launch with a tablet user interface (UI).

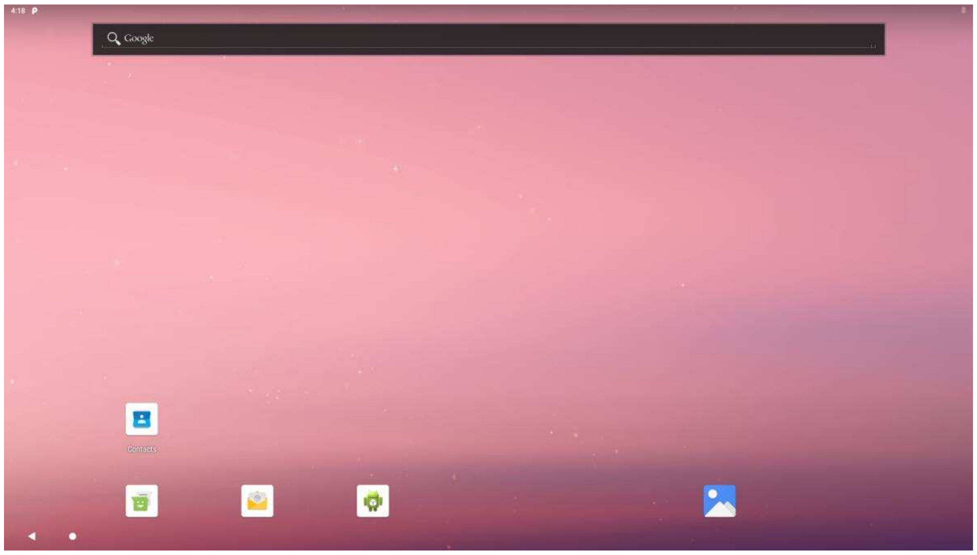

Figure 1. Home Screen for Tablet Configuration

- After creation of the automotive target on the TI development board, the following user interface will be available.

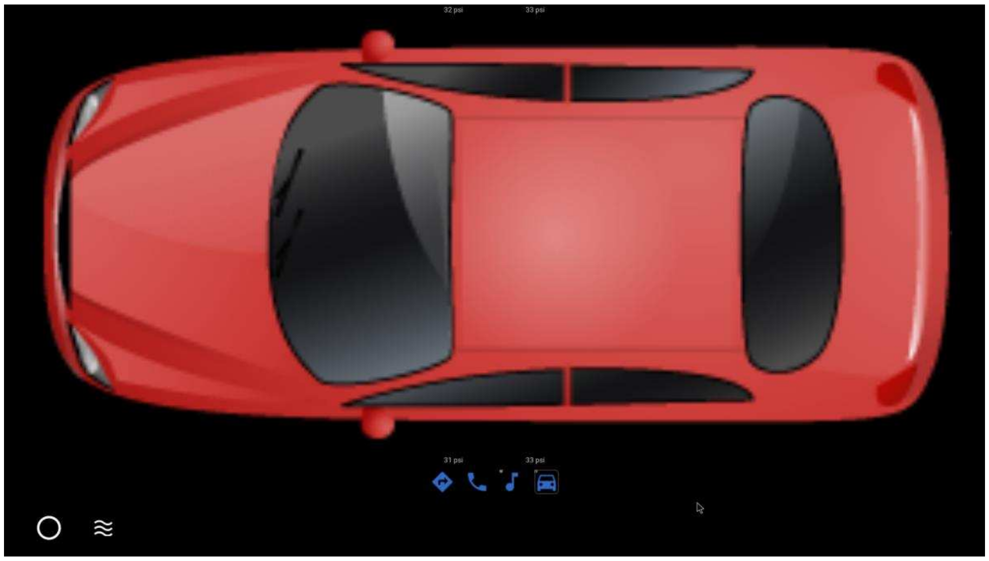

Figure 2. Start Screen for Automotive Configuration, Depicting a Car With PSI Information Near Tires

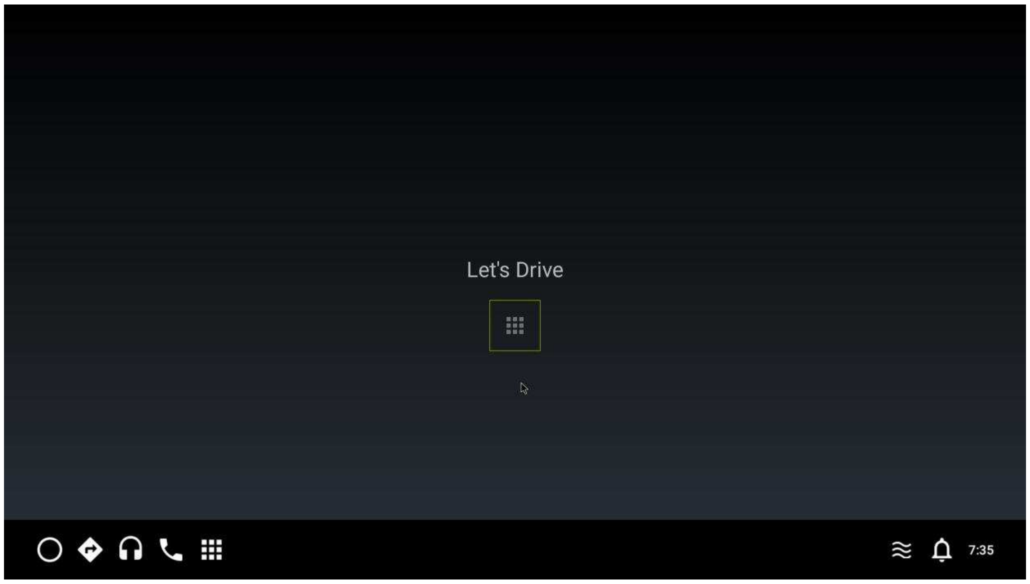

Figure 3. "Let's Drive" Interface for Automotive Configuration (allowing you to interact with applications by selecting the icon in the middle)

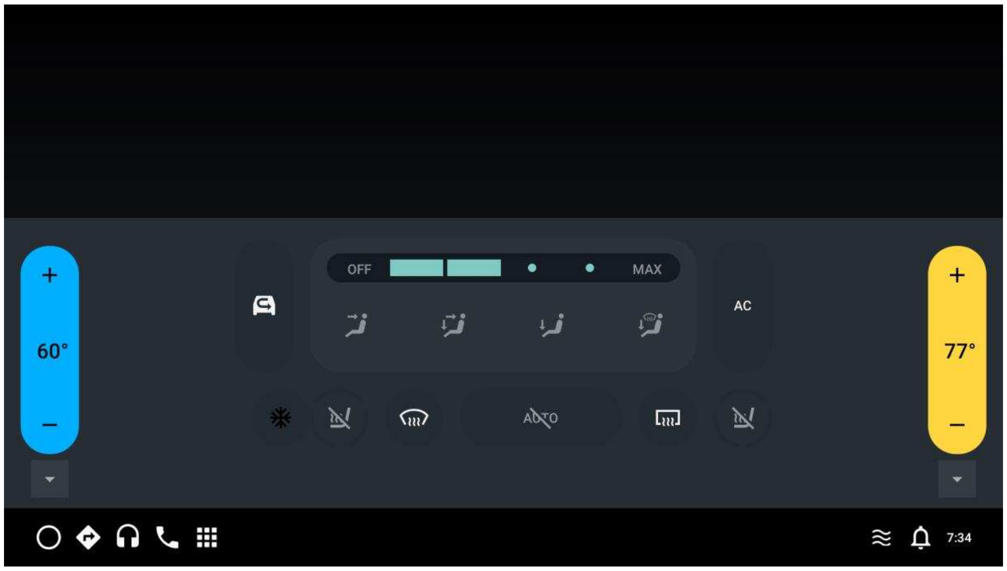

Figure 4. Climate Control Interface for Automotive Configuration (allowing you to adjust temperature, fan speed and direction, seat warmers, and so forth)

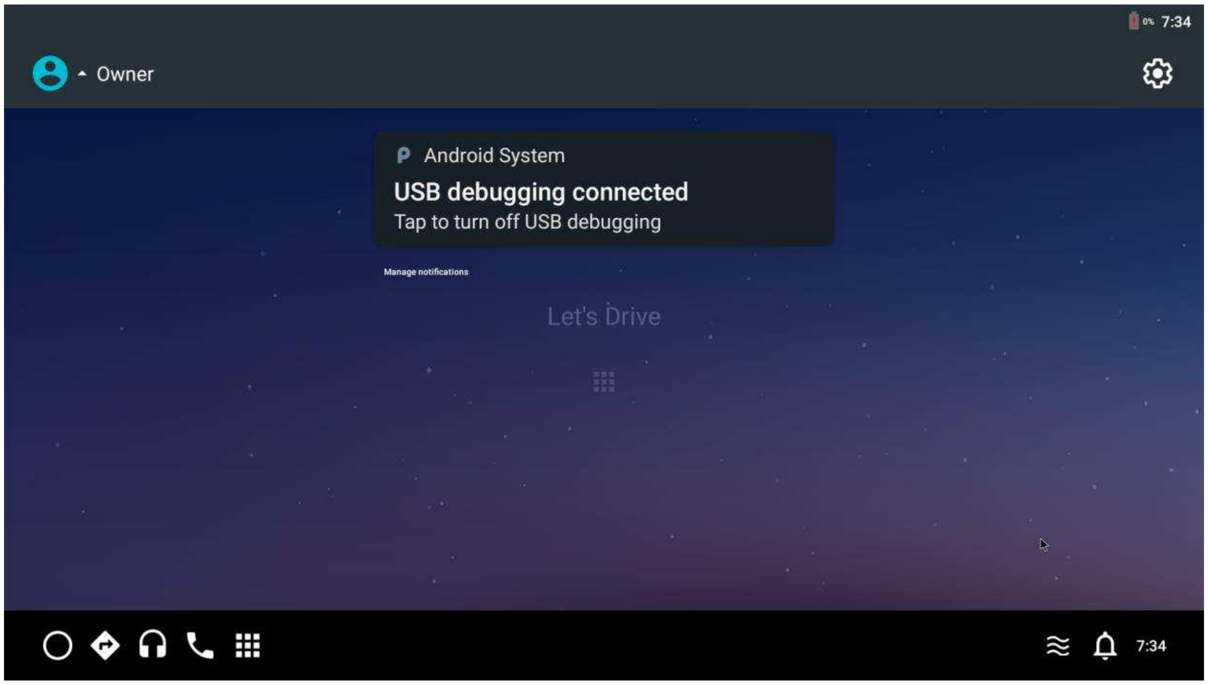

Figure 5. Notification Pane for Automotive Configuration (presenting you with up-to-date notifications regarding your vehicle)

2 Implementation

2.1 Prerequisites

1. Check to make sure your setup meets the hardware and software requirements as given by Android.
2. Environment Setup:
   a. **Android Pastry Version**: Pie (based on http://git.ti.com/cgit/cgit.cgi/android/manifest.git/log/?h=pie-core-release)
   b. **Linux Kernel**: ti-android-linux-4.19.y
   c. **Toolchain**: gcc-arm-8.3-2019.03-x86_64-arm-linux-gnueabihf
3. Follow the steps provided in the Processor SDK Android 1.1 Getting Started Guide to install the necessary components and 1.3 Building the SDK to prepare for development on the TI microprocessor.
4. Establish your Build Environment for Android.
5. Ensure that your device meets the Automotive Requirements defined by the Android Compatibility Definition Document.

2.2 Software Changes

NOTE: Please change directory to your Android File System before continuing.

1. Changes inside *device/ti/beagle_x15/AndroidProducts.mk*:
   a. These changes add a new lunch combo called “beagle_x15_auto-userdebug” and it’s path to the relative top-level makefile.

```
PRODUCT_MAKEFILES += \
-     $(LOCAL_DIR)/beagle_x15.mk \
+     beagle_x15:$(LOCAL_DIR)/beagle_x15.mk \
+     beagle_x15_auto:$(LOCAL_DIR)/auto/beagle_x15.mk \
COMMON_LUNCH_CHOICES := \
      beagle_x15-userdebug \
+     beagle_x15_auto-userdebug \
```

2. Changes inside *device/ti/beagle_x15/BoardConfig.mk*:
   a. Conditional is needed to differentiate the content that is included depending on the TARGET_PRODUCT (macro set by lunch combo).
   b. DEVICE_MANIFEST_FILE adds the auto-specific manifest file to the build.

```
BOARD_SEPOLICY_DIRS += \
    device/ti/beagle_x15/sepolicy

+ifeq ($(TARGET_PRODUCT), beagle_x15_auto)
+BOARD_SEPOLICY_DIRS += \
+   packages/services/Car/car_product/sepolicy
+
+DEVICE_MANIFEST_FILE += device/ti/beagle_x15/auto/manifest.xml
+endif
```

3. Create an *auto* directory inside of *device/ti/beagle_x15*.

NOTE: The following steps will all be completed inside the *device/ti/beagle_x15/auto* directory.

4. Create a top-level makefile for the "beagle_x15_auto-userdebug" lunch combo.
   a. The top-level makefiles are inherited to include all necessary components (PRODUCT_PACKAGES, PRODUCT_COPY_FILES, PRODUCT_PROPERTY_OVERRIDES, and so forth) to enable desired features.
      i. *auto/device.mk* and *car.mk* are crucial to enabling all automotive-specific features.
      ii. The content inside the *car.mk*, the common makefile for all car builds, can be found here.
      iii. *car.mk* then inherits *car_base.mk*, the base platform for car builds, whose content can be found here.
      iv. Inside *car_base.mk*, the auto-specific PRODUCT_PROPERTY_OVERLAYS are added to customize the product at build time.
   b. Insert the following content:

```
$(call inherit-product, device/ti/beagle_x15/device.mk)
$(call inherit-product, device/ti/beagle_x15/auto/device.mk)
$(call inherit-product, $(SRC_TARGET_DIR)/product/full_base.mk)
$(call inherit-product, packages/services/Car/car_product/build/car.mk)

PRODUCT_NAME := beagle_x15_auto
PRODUCT_DEVICE := beagle_x15
PRODUCT_BRAND := Android
PRODUCT_MODEL := AOSP Auto on BeagleBoard X15
PRODUCT_MANUFACTURER := Texas Instruments Inc
```

5. Create the *device.mk* to declare the auto-specific files and modules needed for the device.
   a. The *android.hardware.automotive.vehicle@2.0-service* PRODUCT_PACKAGE is the Android Automotive hardware abstraction layer (HAL), which is the interface definition between the car and the vehicle network service.
   b. The *frameworks/../../../android.hardware.type.automotive.xml* PRODUCT_COPY_FILE determines that the Android device implementation is classified as “Automotive”.
   c. Insert the following content:

```
PRODUCT_PACKAGES += \
        android.hardware.automotive.vehicle@2.0-service \

PRODUCT_COPY_FILES += \

frameworks/native/data/etc/android.hardware.type.automotive.xml:$(TARGET_COPY_OUT_VENDOR)/etc/p
ermissions/android.hardware.type.automotive.xml \

frameworks/native/data/etc/android.hardware.screen.landscape.xml:$(TARGET_COPY_OUT_VENDOR)/etc/
permissions/android.hardware.screen.landscape.xml \

PRODUCT_PROPERTY_OVERRIDES += \
        android.car.drawer.unlimited=true \
        android.car.hvac.demo=true \
        com.android.car.radio.demo=true \
        com.android.car.radio.demo.dual=true \
```

6. Create the *manifest.xml* to provide the HAL(s) specific to automotive.
   a. Beagle_x15 requires the *android.hardware.automotive.vehicle* inclusion to define the product as automotive.
   b. Insert the following content:

```
<manifest version="1.0" type="device">
    <hal format="hidl">
        <name>android.hardware.automotive.vehicle</name>
        <transport>hwbinder</transport>
        <version>2.0</version>
        <interface>
            <name>IVehicle</name>
            <instance>default</instance>
        </interface>
    </hal>
</manifest>
```

3 Deploy Instructions

3.1 Build Instructions

1. Change directory to your Android File System.
2. To setup your build environment, run the following script:

```
$ . ./build/envsetup.sh
```

3. To view the “lunch” menu and select a combo (TARGET_PRODUCT):

```
$ lunch
```

4. Select the desired TARGET_PRODUCT (<board>_auto-userdebug).
5. Export the path to your Kernel directory:

```
$ export KERNELDIR=<path_to_kernel_directory>
```

6. Build:

```
$ make -j<number_of_cores>
```

3.2 Boot Instructions

1. Once build is complete, prepare your eMMC binaries and images:
   a. Make a copy of the prebuilt-images directory to be used as a working directory for flashing.

NOTE: Review the *<Android_FS>/device/ti/<board>/fastboot.sh* for exact images to copy into flashing directory.

AM5

```
cd ~/ti-processor-sdk-android-beagle_x15-<version>/board-support/
cp -rv prebuilt-images emmc_files
cp -v ./aosp-
<version>/out/target/product/beagle_x15/{boot.img,boot_fit.img,recovery.img,system.img,userdata
.img,vendor.img} emmc_files
cp -v ./aosp-<version>/device/ti/beagle_x15/fastboot.sh emmc_files
cp -v ./aosp-<version>/out/host/linux-x86/bin/{simg2img,mkbootimg,fastboot,adb} emmc_files
cp -v ./linux-<version>/arch/arm/boot/zImage emmc_files/zImage-beagle_x15.bin
cp -v ./linux-<version>/arch/arm/boot/dts/am57*.dtb emmc_files
cp -v ./u-boot-<version>/MLO emmc_files
cp -v ./u-boot-<version>/u-boot.img emmc_files
```

AM6

```
cd ~/ti-processor-sdk-android-am65xx-evm-<version>/board-support/
cp -rv prebuilt-images emmc_files
cp -v ./aosp-
<version>/out/target/product/am65xevm/{boot_fit.img,system.img,userdata.img,vendor.img}
emmc_files
cp -v ./aosp-<version>/device/ti/am65xevm/fastboot.sh emmc_files
cp -v ./aosp-<version>/out/host/linux-x86/bin/{simg2img,mkbootimg,fastboot,adb} emmc_files
cp -v ./aosp-<version>/out/host/linux-
x86/lib64/{libc++.so,libcutils.so,liblog.so,libselinux.so,libpcre2.so} emmc_files
cp -v ./u-boot_build/a53/u-boot.img emmc_files
cp -v ./u-boot_build/a53/tispl.bin emmc_files
cp -v ./u-boot_build/r5/tiboot3.bin emmc_files
```

2. Ensure the device is plugged in with USB Host and Debug UART/Serial Debug.
3. Open a terminal debugger to view console output from the device (**dmesg**: kernel ring buffer, **logcat**: log of system messages):

```
$ sudo picocom -b 115200 /dev/ttyUSB0
```

4. Enable fastboot on the device through the terminal debugger:

```
=> fastboot 0
```

5. In a different terminal, change directory to *emmc_files* and run the *fastboot.sh* script:

```
$ cd ~/ti-processor-sdk-android-<board>-<version>/board-support/emmc_files
$ sudo ./fastboot.sh; sudo ./fastboot reboot
```

4 Testing

4.1 Prerequisites/Setup for Automotive Testing

4.1.1 Compatibility Test Suite (CTS)

- Runs on your desktop and executes a series of test cases on the connected device(s).
- The intent is to reveal incompatibilities or malfunctions, ensure API compatibility and Android Compatibility Definition Document (CDD) compliance while developing the automotive configuration and additional related software components (Overview of CTS).

1. Desktop machine setup:
   a. Ensure the latest versions of both Android Debug Bridge (adb) and Android Asset Packaging Tool (AAPT) are installed and those tools' location(s) are added to the system path of your machine:

   ```
   $ sudo apt-get install adb aapt
   ```

   NOTE: The SDK should already come with adb and aapt so the above command will either update/upgrade them or return the message:
   - "aapt is already the newest version <version_number>"
   - "adb is already the newest version <version_number>"

   b. Download the CTS packages matching your devices' Android version and the Application Binary Interfaces (ABIs) your devices support.

   c. Unzip the CTS packages inside of *~/ti-processor-sdk-android-<board>-<version>/board-support/emmc_files*:

   ```
   $ cd ~/ti-processor-sdk-android-<board>-<version>/board-support/emmc_files
   $ unzip ~/Downloads/android-cts-<version>.zip .
   ```

   d. You may need to override the ro.product.first_api_level property in order to ensure upgrade requirements are applied.
      i. Find the valid override values on Codenames, Tags, and Build Numbers.
      ii. The following would be done inside the relative *device.mk* file:

   ```
   PRODUCT_PROPERTY_OVERRIDES += ro.product.first_api_level=<valid_override_value>
   ```

4.1.2 Vendor Test Suite (VTS)

- Used for automated HAL and OS Kernel testing.
  – Complete the steps given at Systems Testing with VTS.

4.2 Automotive Testing

4.2.1 Compatibility Test Suite (CTS)

1. Running CTS Tests:
   a. While inside the *~/ti-processor-sdk-android-<board>-<version>/board-support/emmc_files* directory, run the following commands:
   ```
   $ cd android-cts/tools
   $ sudo ./cts-tradefed
   ```
   b. Ensure that the Device Under Test (DUT) is found when launching cts-tradefed.
      i. The following output should be seen:
      ```
      Android Compatibility Test Suite [Version] (5550264)
      Use "help" or "help all" to get more information on running commands.
      [Date] [Time] I/DeviceManager: Detected new device #
      cts-tf >
      ```
      NOTE: The following commands will all be completed inside the *~/ti-processor-sdk-android-<board>-<version>/board-support/emmc_files* directory.

      ii. If the above message is not seen, run the following command:
      ```
      $ sudo ./adb devices
      ```
      iii. If your device does not show up here, run the following commands:
      ```
      $ sudo ./adb kill-server
      $ sudo ./adb start-server
      ```
      iv. Run the following command:
      ```
      $ sudo ./adb devices
      ```
      v. If your device still does not show up, reboot the device and retry the above steps.
   c. Once the DUT is found, continue.
      i. *CtsCarTestCases* will run car-specific test cases:
      ```
      cts-tf > run cts --module CtsCarTestCases
      ```
   d. "cts-tf > list modules" will list all modules and "cts-tf > list plans" will list all plans.
   e. To exit CTS, "cts-tf > exit".
2. Test Results:
   a. Located in *android-cts/results/<time_stamp>*:
   ```
   ~/ti-processor-sdk-android-<board>-<version>/board-support/emmc_files$ cd android-
   cts/results/<time_stamp>.
   ```

4.2.2 Vendor Test Suite (VTS)

1. Running VTS Tests:
   ```
   ~$ vts-tradefed
   ```
   a. The full list of plans can be found here.
   ```
   vts-tf > list plans
   ```
   b. We are interested in the *vts-hal-auto*, described as the "VTS automotive vehicle HAL test":
   ```
   vts-tf > run vts-hal-auto
   ```
2. Test Results:
   a. Path should be given on completion of test:
   ```
   ~$ cd <Android_FS>/out/host/linux-x86/vts/android-vts/results/<time_stamp>
   ```

5 Open Source

Automotive feature enablement on AM57xx BeagleBoard-X15 is merged to Android Open Source Project.

6 Extending to a New Platform

The work that was completed on the AM57xx BeagleBoard-X15 in the **Implementation → Software Changes** section can be extended to new platforms. The AM65x EVM is used as an example here. The differences between what was given in **Implementation → Software Changes** and what is used for AM65x EVM are as follows:

- Automotive feature enablement on AM65x EVM is merged to Texas Instruments Pie Repository

7 Known Bugs

NOTE: These are known issues and may be resolved in the future depending on their impacts.

1. Sepolicy output while the device is running:

```
~/ti-processor-sdk-android-<board>-<version>/board-
support/emmc_files$ sudo ./adb shell dmesg | grep denied | audit2allow -
p <Android_FS>/out/target/product/beagle_x15/obj/ETC/sepolicy_intermediates/sepolicy
```

   a. May see the following:

```
#============== carservice_app ==============
allow carservice_app runtime_event_log_tags_file:file map;

#============== hal_vehicle_default ==============
allow hal_vehicle_default node:tcp_socket node_bind;
allow hal_vehicle_default self:tcp_socket { accept bind create listen };
```

   b. These cannot be resolved at this time due to SEPolicy neverallow.

2. Power manager service is failing.

   a. May see the following errors:

```
E CarPowerManagerNative: Received unknown bootReason = 0
E PowerTestService: ERROR: Could not read bootReason!!
```

3. CTS *test_result_failures.html* output when tested on pie-core-release branch with an AM57xx BeagleBoard-X15 and an AM65x EVM.
   - *test_result_failures.html* returned one failed test.
     – android.car.cts.CarBluetoothTest#testRequiredBluetoothProfilesExist
   - Details:
     – java.lang.NullPointerException: Attempt to invoke virtual method ‘boolean android.bluetooth.BluetoothAdapter.disable()’ on a null object reference
   - This is expected since the AM57xx BeagleBoard-X15 and AM65x EVM do not natively support bluetooth at this time.

8 Future Work

- Enable Android Auto Human-Machine Interface on new automotive-tuned TI platforms.
- Migrate to Android Q Pastry Version:
  – Add multiple hardware display support:
    - **Interactive:** Individual input mapping, decorative elements and IME (soft keyboard)
    - **Driver:** Main IVI display with all UX restrictions applied
    - **Passenger:** Add support for more types of apps and individual launchers for each screen
  – Multiple users with “headless” system, where a user is not associated with a real user and does not have a lock screen.
  – Multi-zone Audio:
    - A zone is a collection of audio devices and each user device can only be in one zone.
    - Within each zone, audio usage determines which device the audio is routed to.

9 FAQ

Question: What is important about this topic?

Answer: This topic allows the entire Android Open Source community to benefit from an additional feature: Android Automotive on all currently supported Texas Instruments ARM32B and ARM64B platforms, including AM57xx BeagleBoard-X15 and AM65x EVM. Android Automotive will also be applicable to all new automotive-tuned TI platforms.

Question: Who is the intended end user?

Answer: The intended end user(s) include the Automotive Business Unit for Digital Cockpit implementation targeted at infotainment customers who need a graphically rich user interface with an Android extensive app store on a Texas Instruments digital application processor.

Question: When will this document be used?

Answer: This document is intended to implement Android Automotive on any application processor that is designed to showcase and use the functionality of Android.

10 References

- Jacinto 6 EVM on d-oreo-mr1-release: Changes for defining auto platform
- Google Cuttlefish Auto (cf_x86_64_auto-userdebug)
- AOSP Generic Car (aosp_car_x86_64-userdebug)