Treble

Broadcom Proprietary and Confidential

Release Note: This document can be shared with customers.

Treble from Android

The information regarding Android treble can be found
https://source.android.com/devices/architecture/treble.

Integration on Bcm Devices

The treble integration on bcm solution has several aspects which are explained here.

Which Treble?

There are two variants of treble supported by Android:

- "**treble**" device implement some 'lite' version of treble where most HAL are run in pass through mode, selinux is not fully enforcing complete separation of vendor and core (android system), etc.
  - This version is primarily targeted for devices which have an existing Android integration pre O and which cannot be full treble compliant for various reasons (discussed later on).
- "**full treble**" device implement the full set of requirements from treble, including binder-ized HAL, proper selinux enforcement, etc.

What is needed for Full Treble Support

In order for a device to be full treble compliant, it requires to have the following in particular:

- A partition table layout which has a separate vendor and system partitions.
  - optionally an odm partition can be used, but is not necessary.
  - for a|b update compliant devices, both the vendor and system partitions (as well as the odm partition if applicable) must have a dual partitions.
- A version of the bootloader which has support for the multiple (including vendor) partitions and early-mount of the partitions (through device tree configuration).

Which Device Supports What?

Avko|Banff

- those devices are "legacy" designs and have typically shipped with a pre-O version of Android.
- in order to become a full treble device they would require a bootloader and partition table upgrade, while both of those aspects could be brought into a device through OTA (multi stage OTA), doing so on field deployed devices is a risky operation especially if redundancy for bootloader and partition table failures was not built into the device support.
- for those reasons, it was considered un-necessary to turn those devices into a full-treble mode device in the O scope; they are consequently supporting "treble" alone (not "full treble").

Cypress|Dawson|Elfin|Fundy

- those devices are supporting the "full treble" integration in Android O onward.

What are "nvi" Devices?

- each reference device has a "_nvi" variant associated with it, as example: avko and avko_nvi.
- those devices are available in the device configuration profile.
- **"nvi"** stands for **"non vendor image"** this variant is created for cases where a device cannot be modified to fit the treble requirements of providing a vendor image, as example the device cannot be field upgraded to a newer partition table from an existing partition table without vendor image.
- in this case, "full treble" cannot be achieved, but "treble" can and is setup for the target in a single system image baseline.
- those devices are meant to be used for field upgrade to O from prior deployed version of Android which cannot satisfy the bootloader and partition table changes requirements.
- Google is aware of the limitations associated with such devices due to their field deployment status and as such is willing to accept exceptions to treble support due to the device limitations; however further discussion on a case by case basis should be conducted with Google directly.

Interface Definition

The main aspect of treble is about defining and enforcing fixed interfaces. Luckily the default Broadcom integration does not change any existing HAL interface (never has even in versions prior to O), therefore treble-izing a device is relatively straightforward.

Hals Integration

Broadcom integration does not modify any of the HAL interface of the existing HALs provided by google, which means that for treble devices, All HALs implementation are derived from the Android "default" HAL implementation for a given component.

The "default" HAL implementation from Google would already provide passthrough (treble) and binderized (full treble) variants of the HALs and continue to dynamically load the corresponding hardware module (the bcm implementation of the HAL).

For customer integration to treble, changes made to bcm HALs reference implementation that do not require HAL API change can therefore continue to exists in the HAL realm. However, if a HAL API change was made and reflected into the Android realm, then a new treble HAL needs to be generated. Google documentation on HAL creation needs to be followed in such case.

"inexus" HIDL

Certain nexus functionality need to be available to core (Android) services as well as vendor (Broadcom) services; as example any service requiring graphics support (gles, vulkan) requires to authenticate to the nexus server.

In the treble integration this means that such service requires to be constructed over "hwbinder", rather than the formerly used "binder" interface; furthermore the service requires to be a HIDL defined interface, rather than the formerly used AIDL interface. for a definition of the interface mode required by treble, refer to **https://source.android.com/devices/architecture/hidl/binder-ipc**.

Broadcom provides a "nexus" HIDL implementation which satisfies the treble requirements, the interface is available in the code tree under "vendor/broadcom/bcm_platform/hals/nexus". Since it is a HIDL implementation, it is only provided in .hal format and generated in c|cpp format at build time, this ensures that it is always in sync with the Android HIDL framework (as example bug fixes and security updates in the HIDL interface would be picked up automatically).

Usage of the nexus HIDL interface is piped through the Broadcom integration. It is possible to extend the nexus HIDL HAL for custom operations, however it is strongly suggested to consult with Broadcom prior to implementing extensions to the HAL to ensure such is the right path to be taken based on the function desired.

Vendor Private Interface

For vendor privately managed services which do not need to communicate with the core (Android) services, it is possible to maintain a AIDL interface rather than changing to a HIDL interface.

The change is mostly transparent to the code path, the only requirement is to ensure that the proper "vndbinder" is used (rather than the "binder").

In the Broadcom integration, there is a service implementing such AIDL interface allowing the hardware composer (hwc) to talk to other Broadcom services (such as the omx media pipeline). This service remains unchanged for treble, aside from becoming a "vendor" service using the vendor service manager.

Code Paths

In order to integrate treble, it has been in some occasion needed to fork the code implementation, forking the code was deemed a better solution that code pollution due to #ifdef's given the lack of interface compatibility between treble and non treble devices.

At the same time treble was integrated, code cleanup was also enforced to keep the Broadcom integration clear.

When such code forking has taken place, there will be a "treble" subdirectory and a "legacy" subdirectory containing the forked code. "full treble" devices will use the "treble" code path (built time selection) while other devices (non full treble, including treble "lite") will use the "legacy" code path.

Selinux Policy

The second aspect of the treble integration which is quite distinct between "full treble" devices and not is the selinux policy enforcement.

"full treble" devices may require code path changes as well to satisfy the treble requirements due to the limitations imposed by the selinux policy.

Similarly to the code path forking mentioned above, the selinux device policy is forked for "treble" (full treble) device and for "legacy" (any other) devices.

vndk

The last and not least aspect of the integration is to do with the vndk integration. The android integration follows the vndk rules strictly and exactly.

Prior to android P, the vndk is created by the vendor integration based on existing template; in P integration and later, the vndk is directly generated and validated by the android build process.