Architecture

Android system architecture contains the following components:

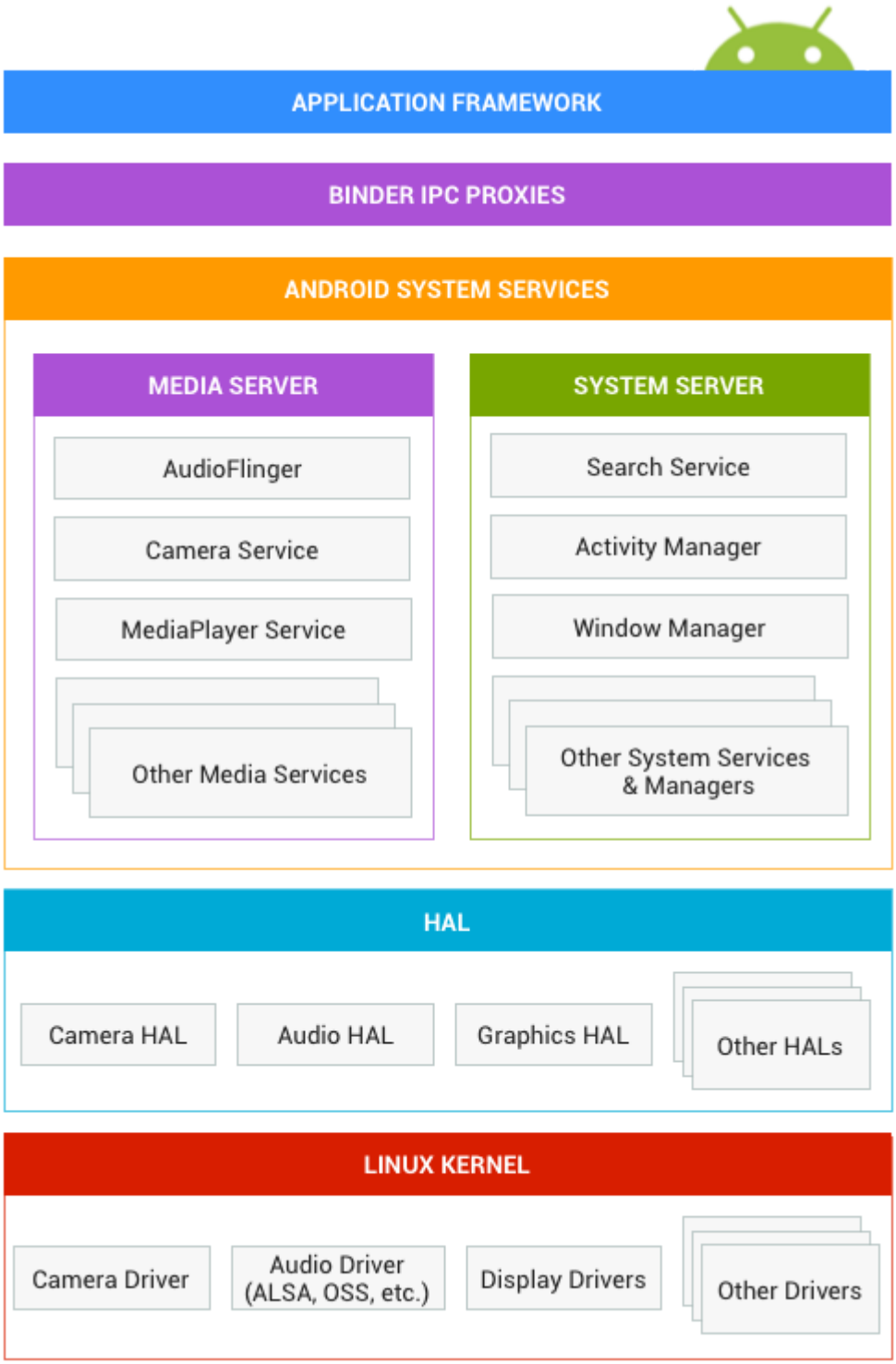

Figure 1. Android system architecture

- **Application framework**. The application framework is used most often by application developers. As a hardware developer, you should be aware of developer APIs as many

map directly to the underlying HAL interfaces and can provide helpful information about implementing drivers.

- **Binder IPC**. The Binder Inter-Process Communication (IPC) mechanism allows the application framework to cross process boundaries and call into the Android system services code. This enables high level framework APIs to interact with Android system services. At the application framework level, this communication is hidden from the developer and things appear to "just work".
- **System services**. System services are modular, focused components such as Window Manager, Search Service, or Notification Manager. Functionality exposed by application framework APIs communicates with system services to access the underlying hardware. Android includes two groups of services: *system* (such as Window Manager and Notification Manager) and *media* (services involved in playing and recording media).
- **Hardware abstraction layer (HAL)**. A HAL defines a standard interface for hardware vendors to implement, which enables Android to be agnostic about lower-level driver implementations. Using a HAL allows you to implement functionality without affecting or modifying the higher level system. HAL implementations are packaged into modules and loaded by the Android system at the appropriate time. For details, see Hardware Abstraction Layer (HAL) (https://source.android.com/devices/architecture/hal.html).
- **Linux kernel**. Developing your device drivers is similar to developing a typical Linux device driver. Android uses a version of the Linux kernel with a few special additions such as Low Memory Killer (a memory management system that is more aggressive in preserving memory), wake locks (a `PowerManager` (https://developer.android.com/reference/android/os/PowerManager.html) system service), the Binder IPC driver, and other features important for a mobile embedded platform. These additions are primarily for system functionality and do not affect driver development. You can use any version of the kernel as long as it supports the required features (such as the binder driver). However, we recommend using the latest version of the Android kernel. For details, see Building Kernels (https://source.android.com/setup/building-kernels.html).

HAL interface definition language (HIDL)

Android 8.0 re-architected the Android OS framework (in a project known as *Treble*) to make it easier, faster, and less costly for manufacturers to update devices to a new version of Android. In this new architecture, the HAL interface definition language (HIDL, pronounced

"hide-l") specifies the interface between a HAL and its users, enabling the Android framework to be replaced without rebuilding the HALs.

Note: For more details on Project Treble, refer to the developer blog posts Here comes Treble: A modular base for Android (https://android-developers.googleblog.com/2017/05/here-comes-treble-modular-base-for.html) and Faster Adoption with Project Treble (https://android-developers.googleblog.com/2018/05/faster-adoption-with-project-treble.html).

HIDL separates the vendor implementation (device-specific, lower-level software written by silicon manufacturers) from the Android OS framework via a new vendor interface. Vendors or SOC makers build HALs once and place them in a `/vendor` partition on the device; the framework, in its own partition, can then be replaced with an over-the-air (OTA) update (https://source.android.com/devices/tech/ota/) without recompiling the HALs.

The difference between the legacy Android architecture and the current, HIDL-based architecture is in the use of the vendor interface:

- In Android 7.x and earlier, no formal vendor interface exists, so device makers must update large portions of the Android code to move a device to a newer version of Android:

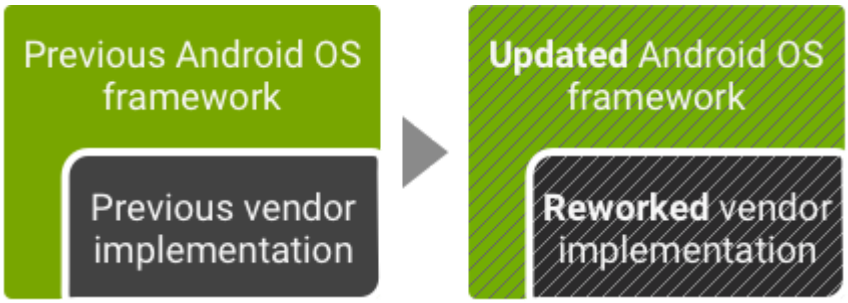

Figure 2. Legacy Android update environment

- In Android 8.0 and higher, a new stable vendor interface provides access to the hardware-specific parts of Android, so device makers can deliver new Android releases simply by updating the Android OS framework—without additional work required from the silicon manufacturers:

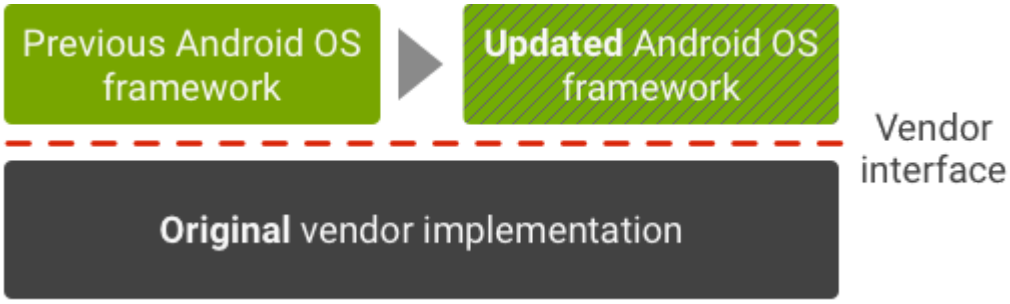

Figure 3. Current Android update environment

All new devices launching with Android 8.0 and higher can take advantage of the new architecture. To ensure forward compatibility of vendor implementations, the vendor interface is validated by the Vendor Test Suite (VTS) (https://source.android.com/devices/tech/vts/index.html), which is analogous to the Compatibility Test Suite (CTS) (https://source.android.com/compatibility/cts/). You can use VTS to automate HAL and OS kernel testing in both legacy and current Android architectures.

Architecture resources

For details on the Android architecture, see the following sections:

- HAL Types (https://source.android.com/devices/architecture/hal-types.html). Describes binderized, passthrough, Same-Process (SP), and legacy HALs.
- HIDL (General) (https://source.android.com/devices/architecture/hidl/index.html). Contains general information about the interface between a HAL and its users.
- HIDL (C++) (https://source.android.com/devices/architecture/hidl-cpp/index.html). Contains details for creating C++ implementations of HIDL interfaces.
- HIDL (Java) (https://source.android.com/devices/architecture/hidl-java/index.html). Contains details about the Java frontend for HIDL interfaces.
- ConfigStore HAL (https://source.android.com/devices/architecture/configstore/index.html). Describes APIs for accessing read-only configuration items used to configure the Android framework.
- Device Tree Overlays (https://source.android.com/devices/architecture/dto/index.html). Provides details on using device tree overlays (DTOs) in Android.
- Vendor Native Development Kit (VNDK) (https://source.android.com/devices/architecture/vndk/index.html). Describes the set of vendor-exclusive libraries for implementing vendor HALs.

- Vendor Interface Object (VINTF) (https://source.android.com/devices/architecture/vintf/index.html). Describes the objects that aggregate relevant information about a device and make that information available through a queryable API.
- SELinux for Android 8.0 (https://source.android.com/security/selinux/images/SELinux_Treble.pdf). Details SELinux changes and customizations.

Last updated June 26, 2018.