**Android Camera Framework Architecture**

* The goal of android project is to create succesfully real-world product that improves the mobile experience for end users.
* **Camera use cases**

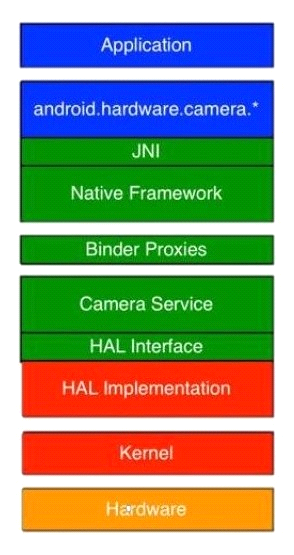
* Live Preview of camera stream
* capture a frame
* video recording of a camera stream
* **Android Camera**

The Android framework includes support for various cameras and camera features available on devices, allowing you to capture pictures and videos in your applications. This document discusses a quick, simple approach to image and video capture for users.

* The Android framework supports capturing images and video through the **android.hardware.camera** API .
* To take pictures with this class, use the following steps:

1. Obtain an instance of Camera from [open(int)](https://developer.android.com/reference/android/hardware/Camera#open(int)).
2. Get existing (default) settings with [getParameters()](https://developer.android.com/reference/android/hardware/Camera#getParameters()).
3. If necessary, modify the returned [Camera.Parameters](https://developer.android.com/reference/android/hardware/Camera.Parameters) object and call [setParameters(android.hardware.Camera.Parameters)](https://developer.android.com/reference/android/hardware/Camera#setParameters(android.hardware.Camera.Parameters)).
4. Call [setDisplayOrientation(int)](https://developer.android.com/reference/android/hardware/Camera#setDisplayOrientation(int)) to ensure correct orientation of preview.
5. Call [startPreview()](https://developer.android.com/reference/android/hardware/Camera#startPreview()) to start updating the preview surface. Preview must be started before you can take a picture.
6. Call [stopPreview()](https://developer.android.com/reference/android/hardware/Camera#stopPreview()) to stop updating the preview surface.
7. Call [release()](https://developer.android.com/reference/android/hardware/Camera#release()) to release the camera for use by other applications.

**Camera Architecture :**



**framework/base/core/java/android/harware/camera.java**

**API**

**frameworks/base/core/jni/android\_hardware\_Camera.cpp.**

**frameworks/av/camera/Camera.cpp**

**frameworks/av/camera**

**Icamera**

**Icameraservice**

**Icameraclient**

**frameworks/av/services/camera/libcameraservice/CameraService.cpp**

**hardware/libhardware/include/hardware/camera.h**

**hardware/<vendor>/camera/Qcamera2/HAL/Qcamera2HWI.cpp**

**kernel/drivers/media/v4l2**

**Application Framework :**

* At the application framework level is the app's code, which utilizes the [**android.hardware.Camera**](http://developer.android.com/reference/android/hardware/Camera.html) API to interact with the camera hardware. Internally, this code calls a corresponding JNI glue class to access the native code that interacts with the camera.
* Java classes Exposed to the application for interfacing with the hardware.

**Framework/base/core/java/android/harware/camera.java**

**EX :**

public Parameters getParameters() {

Parameters p = new Parameters();

String s = native\_getParameters();

p.unflatten(s);

return p;

}

**JNI :**

* It is the glue code between java classes and native classes
* The JNI code associated with [**android.hardware.Camera**](http://developer.android.com/reference/android/hardware/Camera.html) is located in

**frameworks/base/core/jni/android\_hardware\_Camera.cpp**.

**EX :**

static jstring android\_hardware\_Camera\_getParameters(JNIEnv \*env, jobject thiz)

{

ALOGV("getParameters");

sp<Camera> camera = get\_native\_camera(env, thiz, NULL);

if (camera == 0) return 0;

**Native framework :**

* The native framework defined in **frameworks/av/camera/Camera.cpp** . This class calls the IPC binder proxies to obtain access to the camera service.

**EX** :

String8 Camera::getParameters() const

{

ALOGV("getParameters");

String8 params;

sp <::android::hardware::ICamera> c = mCamera;

if (c != 0) params = mCamera->getParameters();

return params;

}

**Binder IPC proxies :**

* The IPC binder proxies facilitate communication over process boundaries. There are three camera binder classes that are located in **frameworks/av/camera** directory that calls into camera service.
* ICameraService is the interface to the camera service,
* ICamera is the interface to a specific opened camera device.
* and ICameraClient is the device's interface back to the application framework.

**EX :**

String8 getParameters() const

{

ALOGV("getParameters");

Parcel data, reply;

data.writeInterfaceToken(ICamera::getInterfaceDescriptor());

remote()->transact(GET\_PARAMETERS, data, &reply);

return reply.readString8();

}

**Camera service :**

* The camera service, located in **frameworks/av/services/camera/libcameraservice/CameraService.cpp**, is the actual code that interacts with the HAL.
* It manages permissions and lifecycle of the camera devices.

**HAL interface :**

* hardware intercace loactaed in **hardware/libhardware/include/hardware see** camera.h and camera\_common.h files
* The Camera Hardware Abstraction Layer (HAL) is a library that is specific to the camera hardware platform
* Written by hardware vendors (like Qualcomm, TI)

**HAL implementation :**

* hardware specific implementations depends on host processor ,isp and sensor located in **hardware/<vendor>/camera/Qcamera2/HAL/Qcamera2HWI.cpp**

**Kernel drivers :**

* In general, the low-level implementation of the Camera APIs is V4L2,the camrea driver itself in **kernel/drivers/media/v4l2**
* **Implementing the HAL**

The HAL sits between the camera driver and the higher-level Android framework. The HAL interface is defined in the hardware/libhardware/include/hardware/camera.h and hardware/libhardware/include/hardware/camera\_common.h header files.

camera\_common.h defines camera\_module, a standard structure to obtain general information about the camera, such as the camera ID and properties common to all cameras (that is, whether it is a front- or back-facing camera).

### Configuring the shared library

1. Create a device/<company\_name>/<device\_name>/camera directory to contain your library's source files.
2. Create an Android.mk file to build the shared library. Ensure that the makefile contains the following lines:

LOCAL\_MODULE := camera.<device\_name>

LOCAL\_MODULE\_RELATIVE\_PATH := hw

**3 .** Specify your device has camera features by copying the necessary feature XML files in the frameworks/native/data/etc directory with your device's makefile. For example, to specify your device has a camera flash and can autofocus, add the following lines in your device's <device>/<company\_name>/<device\_name>/device.mk makefile:

For an example of a device makefile, see device/samsung/tuna/device.mk.

**4 .** Declare your camera’s media codec, format, and resolution capabilities in device/<company\_name>/<device\_name>/media\_profiles.xml and device/<company\_name>/<device\_name>/media\_codecs.xml XML files.

**5.** Add the following lines in your device's device/<company\_name>/<device\_name>/device.mk makefile to copy the media\_profiles.xml and media\_codecs.xml files to the appropriate location:

# media config xml file

PRODUCT\_COPY\_FILES += \

<device>/<company>/<device>/media\_profiles.xml:system/etc/media\_profiles.xml

**6 .** To include the Camera app in your device's system image, specify it in the PRODUCT\_PACKAGES variable in your device's device/<company>/<device>/device.mk makefile:

PRODUCT\_PACKAGES := \

Gallery2 \

...