USB/UVC Integrated Cameras

Contents

© 2018 Google LLC. All Rights Reserved. No express or implied warranties are provided for herein. All specifications are subject to change and any expected future products, features or functionality will be provided on an if and when available basis.

Overview

The Android platform supports the use of plug-and-play USB cameras (i.e. webcams) using the standard Android Camera2 API and the camera HIDL interface. Webcams generally support USB video class (UVC) drivers and on Linux the standard Video4Linux (V4L) driver is used to control UVC cameras.

With support for webcams, devices can be used in lightweight use cases such as video chatting and photo kiosks. This feature does not serve as a replacement for typical internal camera HALs on Android phones and is not designed to support performance-intensive, complex tasks involving high-resolution and high-speed streaming, AR, and manual ISP/sensor/lens control.

The new USB camera HAL process is part of the external camera provider that listens to USB device availability and enumerates external camera devices accordingly. The process has permissions and an SE policy similar to the built-in camera HAL process. Third party webcam applications that communicate directly with USB devices require the same camera permissions to access UVC devices as with any regular camera application.

Examples and sources

For more information on how to implement USB cameras, see an external camera provider reference

implementation at `ExternalCameraProvider`. The external camera device and session implementations are included in `ExternalCameraDevice` and `ExternalCameraDeviceSession`.

The Java client API includes a new `EXTERNAL` hardware level.

Implementation

The implementation must support the `android.hardware.usb.host` system feature.

Kernel support for UVC devices must also be enabled. You can enable this by adding the following to the respective kernel `deconfig` files.

```
+CONFIG_USB_VIDEO_CLASS=y
+CONFIG_MEDIA_USB_SUPPORT=y
```

Note: Make sure you also have this patch for uvcvideo.

To enable the external camera provider in the respective device build, which adds the necessary SELinux permissions, external camera configuration, and external camera provider dependency, complete the following steps:

- Add external camera config file and external camera library to `device.mk`

```
+PRODUCT_PACKAGES += android.hardware.camera.provider@2.4-impl
+PRODUCT_PACKAGES += android.hardware.camera.provider@2.4-external-service

+PRODUCT_COPY_FILES += \
+device/manufacturerX/productY/external_camera_config.xml:$(TARGET_COPY_OUT_VENDOR)/etc/
external_camera_config.xml
```

- Add external camera provider name to device Treble HAL manifest

```
<hal format="hidl">
   <name>android.hardware.camera.provider</name>
   <transport arch="32+64">passthrough</transport>
   <impl level="generic"></impl>
   <version>2.4</version>
   <interface>
       <name>ICameraProvider</name>
       <instance>legacy/0</instance>
```

```
+       <instance>external/0</instance>
    </interface>
</hal>
```

- (Optional) If the device runs in Treble passthrough mode, update `sepolicy` so `cameraserver` can access UVC camera

```
+# for external camera
+allow cameraserver device:dir r_dir_perms;
+allow cameraserver video_device:dir r_dir_perms;
+allow cameraserver video_device:chr_file rw_file_perms;
```

Here is an example of `external_camera_config.xml` (copyright lines omitted)

```
<ExternalCamera>
    <Provider>
        <ignore> <!-- Internal video devices to be ignored by external camera HAL -->
            <id>0</id> <!-- No leading/trailing spaces -->
            <id>1</id>
        </ignore>
    </Provider>
    <!-- See ExternalCameraUtils.cpp for default values of Device configurations below
-->
    <Device>
        <!-- Max JPEG buffer size in bytes-->
        <MaxJpegBufferSize bytes="3145728"/> <!-- 3MB (~= 1080p YUV420) -->
        <!-- Size of v4l2 buffer queue when streaming >= 30fps -->
        <!-- Larger value: more request can be cached pipeline (less janky)  -->
        <!-- Smaller value: use less memory -->
        <NumVideoBuffers count="4"/>
        <!-- Size of v4l2 buffer queue when streaming < 30fps -->
        <NumStillBuffers count="2"/>

        <!-- List of maximum fps for various output sizes -->
        <!-- Any image size smaller than the size listed in Limit row will report
            fps (as minimum frame duration) up to the fpsBound value. -->
        <FpsList>
            <!-- width/height must be increasing, fpsBound must be decreasing-->
            <Limit width="640" height="480" fpsBound="30.0"/>
            <Limit width="1280" height="720" fpsBound="15.0"/>
            <Limit width="1920" height="1080" fpsBound="10.0"/>
            <!-- image size larger than the last entry will not be supported-->
        </FpsList>
```

```
    </Device>
</ExternalCamera>
```

Customization

You may enhance the Android camera either through general customization options or device-specific optimizations.

General customizations

You can customize the external camera provider by modifying the `external_camera_config.xml` file. Specifically, clients can customize the following parameters:

- Excluding video nodes of internal camera(s)
- Supported image size and frame rate upper bound
- Number of inflight buffers (jank vs memory tradeoff)

In addition to these parameters, you can add your own parameters or develop your own configurations.

Device-specific optimizations

You can also improve performance by adding device-specific optimizations.

Buffer copy/scaling and JPEG decode/encode

Generic implementations use CPU (libyuv/libjpeg) but you can replace this with device-specific optimizations.

HAL output format

Generic implementations use the following output formats:

- YUV_420_888 for video IMPLEMENTATION_DEFINED buffers.
- YV12 for all other IMPLEMENTATION_DEFINED buffers.

To improve performance, you can replace output formats with device-specific efficient formats. You can also support additional formats in a customized implementation

Validation

Devices with external camera support must pass camera CTS. The external USB webcam must remain plugged in the specific device during the entire test run, otherwise some test cases will fail.

Note: `media_profiles` entries are not available for external USB webcams, so camcorder profiles are absent too.