**Camera API**

**Table of Contents :**

* Introduction
* Manifest declaration
* Building a Camera APP

### Detecting camera hardware

### Accessing cameras

### Checking camera features

### Creating a preview class

### Capturing pictures

### 

**Introduction**

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 and outlines an advanced approach for creating custom camera experiences for your users.

* The Android framework supports capturing images and video through the [android.hardware.camera2](https://developer.android.com/reference/android/hardware/camera2/package-summary) API
* [android.hardware.camera2](https://developer.android.com/reference/android/hardware/camera2/package-summary)

This package is the primary API for controlling device cameras. It can be used to take pictures or videos when you are building a camera application.

* [SurfaceView](https://developer.android.com/reference/android/view/SurfaceView)

This class is used to present a live camera preview to the user.

* [MediaRecorder](https://developer.android.com/reference/android/media/MediaRecorder)

This class is used to record video from the camera.

## Manifest declarations

Before starting development on your application with the Camera API, you should make sure your manifest has the appropriate declarations to allow use of camera hardware and other related features.

* **Camera Permission**- Your application must request permission to use a device camera.

<uses-permission android:name="android.permission.CAMERA" />

* **Camera Features** - Your application must also declare use of camera features, for example:

<uses-feature android:name="android.hardware.camera" />

* android.hardware.camera.flash

The app uses the flash feature that the device's camera supports. By using this feature, an app implies that it also uses the android.hardware.camera feature, unless this parent feature is declared with android:required="false".

* android.hardware.camera.front

The app uses the device's front-facing camera.

* android.hardware.camera.autofocus

The app uses the autofocus feature that the device's camera supports.

* android.hardware.camera

The app uses the device's back-facing camera. Devices with only a front-facing camera do not list this feature, so use the android.hardware.camera.any feature instead if your app can communicate with any camera, regardless of which direction the camera faces.

* **Storage Permission** - Your application can save images or videos to the device's external storage (SD Card) if it targets Android 10 (API level 29) or lower and specifies the following in the manifest.

<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" />

* **Audio Recording Permission** - For recording audio with video capture, your application must request the audio capture permission.

<uses-permission android:name="android.permission.RECORD\_AUDIO" />

* **Location Permission** - If your application tags images with GPS location information, you must request the ACCESS\_FINE\_LOCATION permission. Note that, if your app targets Android 5.0 (API level 21) or higher, you also need to declare that your app uses the device's GPS:

## Building a camera app

The general steps for creating a custom camera interface for your application are as follows:

* **Detect and Access Camera** - Create code to check for the existence of cameras and request access.
* **Create a Preview Class** - Create a camera preview class that extends [SurfaceView](https://developer.android.com/reference/android/view/SurfaceView) and implements the [SurfaceHolder](https://developer.android.com/reference/android/view/SurfaceHolder) interface. This class previews the live images from the camera.
* **Build a Preview Layout** - Once you have the camera preview class, create a view layout that incorporates the preview and the user interface controls you want.
* **Setup Listeners for Capture** - Connect listeners for your interface controls to start image or video capture in response to user actions, such as pressing a button.
* **Capture and Save Files** - Setup the code for capturing pictures or videos and saving the output.
* **Release the Camera** - After using the camera, your application must properly release it for use by other applications.

### Detecting camera hardware

\* Check if this device has a camera \*/  
private boolean checkCameraHardware(Context context) {  
    if (context.getPackageManager().hasSystemFeature(PackageManager.FEATURE\_CAMERA)){  
        // this device has a camera  
        return true;  
    } else {  
        // no camera on this device  
        return false;  
    }

Android devices can have multiple cameras, for example a back-facing camera for photography and a front-facing camera for video calls. Android 2.3 (API Level 9) and later allows you to check the number of cameras available on a device using the [Camera.getNumberOfCameras()](https://developer.android.com/reference/android/hardware/Camera" \l "getNumberOfCameras()) method.

public static class CameraInfo {

/\* The facing of the camera is opposite to that of the screen.\*/

public static final int CAMERA\_FACING\_BACK = 0;

/\* The facing of the camera is the same as that of the screen.\*/

public static final int CAMERA\_FACING\_FRONT = 1;

### Accessing cameras

If you have determined that the device on which your application is running has a camera, you must request to access it by getting an instance of [Camera](https://developer.android.com/reference/android/hardware/Camera)

/\*\* A safe way to get an instance of the Camera object. \*/  
public static Camera getCameraInstance(){  
    Camera c = null;  
    try {  
        c = Camera.open(); // attempt to get a Camera instance  
    }  
    catch (Exception e){  
        // Camera is not available (in use or does not exist)  
    }  
    return c; // returns null if camera is unavailable  
}

here example code above will access the first, back-facing camera on a device with more than one camera.

### Checking camera features

Once you obtain access to a camera, you can get further information about its capabilities using the [Camera.getParameters()](https://developer.android.com/reference/android/hardware/Camera" \l "getParameters()) method

Use the [Camera.getCameraInfo()](https://developer.android.com/reference/android/hardware/Camera" \l "getCameraInfo(int,%20android.hardware.Camera.CameraInfo)) to determine if a camera is on the front or back of the device, and the orientation of the image.

## Camera features are :

Android supports a wide array of camera features you can control with your camera application, such as picture format, flash mode, focus settings, and many more.

|  |  |
| --- | --- |
| Feature | Description |
| [Face Detection](https://developer.android.com/guide/topics/media/camera#face-detection) | Identify human faces within a picture and use them for focus, metering and white balance |
| [Focus Areas](https://developer.android.com/guide/topics/media/camera#metering-focus-areas) | Set one or more areas within an image to use for focus |
| [White Balance Lock](https://developer.android.com/reference/android/hardware/Camera.Parameters#setAutoWhiteBalanceLock(boolean)) | Stop or start automatic white balance adjustments |
|  |  |
| [Zoom](https://developer.android.com/reference/android/hardware/Camera.Parameters#setZoom(int)) | Set image magnification |
| [Flash Mode](https://developer.android.com/reference/android/hardware/Camera.Parameters#setFlashMode(java.lang.String)) | Turn flash on, off, or use automatic setting |
| [Color Effects](https://developer.android.com/reference/android/hardware/Camera.Parameters#setColorEffect(java.lang.String)) | Apply a color effect to the captured image such as black and white, sepia tone or negative. |

// get Camera parameters  
Camera.Parameters params = camera.getParameters();  
// set the focus mode  
params.setFocusMode(Camera.Parameters.FOCUS\_MODE\_AUTO);  
// set Camera parameters  
camera.setParameters(params);

Your application must start the face detection function each time you start (or restart) the camera preview. Create a method for starting face detection so you can call it as needed, as shown in the example code below.

public void startFaceDetection(){  
    // Try starting Face Detection  
    Camera.Parameters params = mCamera.getParameters();  
  
    // start face detection only \*after\* preview has started  
    if (params.getMaxNumDetectedFaces() > 0){  
        // camera supports face detection, so can start it:  
        mCamera.startFaceDetection();  
    }  
}

### Creating a preview class

For users to effectively take pictures or video, they must be able to see what the device camera sees. A camera preview class is a [SurfaceView](https://developer.android.com/reference/android/view/SurfaceView) that can display the live image data coming from a camera, so users can frame and capture a picture or video.

If you want to set a specific size for your camera preview, set this in the surfaceChanged()

### Capturing pictures

Once you have built a preview class and a view layout in which to display it, you are ready to start capturing images with your application. In your application code, you must set up listeners for your user interface controls to respond to a user action by taking a picture.

In order to retrieve a picture, use the [Camera.takePicture()](https://developer.android.com/reference/android/hardware/Camera" \l "takePicture(android.hardware.Camera.ShutterCallback,%20android.hardware.Camera.PictureCallback,%20android.hardware.Camera.PictureCallback)) method. This method takes three parameters which receive data from the camera. In order to receive data in a JPEG format, you must implement an [Camera.PictureCallback](https://developer.android.com/reference/android/hardware/Camera.PictureCallback) interface to receive the image data and write it to a file.

The following code shows a basic implementation of the [Camera.PictureCallback](https://developer.android.com/reference/android/hardware/Camera.PictureCallback) interface to save an image received from the camera.

/ Add a listener to the Capture button  
Button captureButton = (Button) findViewById(R.id.button\_capture);  
captureButton.setOnClickListener(  
    new View.OnClickListener() {  
        @Override  
        public void onClick(View v) {  
            // get an image from the camera  
            mCamera.takePicture(null, null, picture);  
        }

## Saving media files

Media files created by users such as pictures and videos should be saved to a device's external storage directory (SD Card) to conserve system space and to allow users to access these files without their device. There are many possible directory locations to save media files on a device, however there are only two standard locations you should consider as a developer:

[Environment.getExternalStoragePublicDirectory](https://developer.android.com/reference/android/os/Environment#getExternalStoragePublicDirectory(java.lang.String))([Environment.DIRECTORY\_PICTURES](https://developer.android.com/reference/android/os/Environment#DIRECTORY_PICTURES)) –

This method returns the standard, shared and recommended location for saving pictures and videos. This directory is shared (public), so other applications can easily discover, read, change and delete files saved in this location.

\*\* Create a File for saving an image or video \*/  
private static File getOutputMediaFile(int type){  
    // To be safe, you should check that the SDCard is mounted  
    // using Environment.getExternalStorageState() before doing this.  
  
    File mediaStorageDir = new File(Environment.getExternalStoragePublicDirectory(  
              Environment.DIRECTORY\_PICTURES), "MyCameraApp");  
    // This location works best if you want the created images to be shared  
    // between applications and persist after your app has been uninstalled.  
  
    // Create the storage directory if it does not exist  
    if (! mediaStorageDir.exists()){  
        if (! mediaStorageDir.mkdirs()){  
            Log.d("MyCameraApp", "failed to create directory");  
            return null;  
        }  
    }