Contents

1. Prerequisites

To implement printing functionality in an Android app, an API level 19 or higher is required.

However, if you application deals with PDF documents and need to open or render them, you need to **specify an API level 21** or higher.

```
apply plugin: 'com.android.application'
android {
    compileSdkVersion 24
    buildToolsVersion "24.0.1"

    defaultConfig {
        applicationId "com.example.myapp"
        minSdkVersion 21
        targetSdkVersion 24
        versionCode 1
        versionName "1.0"
    }
    (…)
}
```

In order to let the Android Print Service find HP printers, you must install HP Print Service Plugin and ensure it is enabled it on your Android device (Settings -> Printing).

HP Print Service Plugin can be found on Google Play Store:
https://play.google.com/store/apps/details?id=com.hp.android.printservice

2. API Details

The **PrintDocumentAdapter** abstract class provides the content of a document to be printed. When extending it, you have to implement a set of callbacks to make the necessary actions when your application receives the OS calls for ***onStart()***, ***onLayout()***, ***onWrite()***, ***onFinish()***.

The **PrintAttributes** class describes the attributes of a print job and how the printed content should be laid out. It defines:

- The **color mode** to be applied to the print job (Color or Monochrome).

- The **media size** in which the job will be printed.
- If the job should be printed in **duplex mode** and, if so, which type (short or long edge).
- The **minimal margins** that the developer must respect when displaying the content into the media size.
- The print **resolution**.

The developer must rely on these **PrintAttributes** information to print the application content.

3. Implementing your own PrintAdapter

Android.print API contains the class PrintDocumentAdapter, an abstract class that one may extend in order to implement its abstract methods ***onLayout()*** and ***onWrite()***.

```
onLayout(PrintAttributes oldAttributes, PrintAttributes newAttributes, Can
cellationSignal cancellationSignal, PrintDocumentAdapter.LayoutResultCallb
ack callback, Bundle extras)
```

- The *onLayout()* method is called every time any print attribute changes (eg, when the user selects a different media size or orientation from the print menu). Here you must lay out the content you want to print according to the values received in the *PrintAttributes* class.
  This *onLayout()* call must perform a call to any of the methods the parameter *LayoutResultCallback* before ending. For a successful ending, you must call *LayoutResultCallback.onLayoutFinished*, specifying whether the content changed or not (in order to force or skip a subsequent *onWrite()* call).

```
onWrite(PageRange[] pages, ParcelFileDescriptor destination, CancellationS
ignal cancellationSignal, PrintDocumentAdapter.WriteResultCallback callbac
k)
```

- The *onWrite()* method is called when it is time to generate the pages (in the form of PDF files) that will be printed. It must be called each time there are changes in the layout.
  Same as before, this method implementation must perform a call to any of the methods of the *WriteResultCallback* parameter, indicating a valid page range in case of performing a call to *WriteResultCallback.onWriteFinished*.

4. The LFPrintAdapter class

The LFPRintAdapter class provides an implementation of Android's PrintAdapter interface. It implements three ways to send a job to print:

- *PRINT_CLIP_CONTENT:* When selecting this option, the content to print is placed in the center of the media selected. However, if the content is bigger than the media size, the content will be clipped.

- *PRINT_FIT_TO_PAGE:* This option will place the content to print in the center of the media selected and then scaled down to fit page dimensions, if necessary. There won't be clipping, but there may be white spaces in top/bottom or left/right edges if media aspect ratio is different from content aspect ratio.

- *PRINT_PDF_AS_IS:* This one is similar to *PRINT_CLIP_CONTENT* but rather than placing it in the center of the media selected, it just forwards the original PDF file to the print system. The print service may take some decisions on how to scale or place the content in the media selected.

In addition, you may decide to extend the current PrintJob and/or LFPrintAdapter classes in order to paint your own content direcly on the **Canvas** object of each **PdfDocument.Page** you want to print:

```
PdfDocument.Page page = mDocument.startPage(i);

Canvas c = page.getCanvas();
// Paint whatever you want on the canvas.
mDocument.finishPage(page);

writtenPages.append(writtenPages.size(), i);
```

5. More Info

There are some useful webpages where you can find additional information:

- Android Developer Trainings – Printing Content:
  https://developer.android.com/training/printing/index.html

- Android.print overview:
  https://developer.android.com/reference/android/print/package-summary.html

  - PrintDocumentAdapter class detail:
    https://developer.android.com/reference/android/print/PrintDocumentAdapter.html

  - PrintAttributes class detail:
    https://developer.android.com/reference/android/print/PrintAttributes.html