



Foxit PDF Android SDK

Tutorial – View Demo

Contents

Prerequisites	3
Developer Audience	3
Supported Environments	3
Overview	3
Purpose.....	3
Setup.....	3
Demo Functionalities.....	5
Initializing the environment.....	5
Initializing data	6
Create the bitmap buffer	6
Render the bitmap buffer	6
Generate system compatible bitmap.....	6
Render the Android view	6

Prerequisites

Developer Audience

This document is targeted towards Android developers using the SDK to add PDF functionality to Android applications. It assumes that the developer is familiar with C/C++ and Java.

Supported Environments

Platform	Operating System	Compiler
Android	Android 2.2 and newer.	android-ndk-r7 and newer.

Overview

Purpose

This document shows how to render a PDF page with Android view demo. It also contains necessary steps to start all the other Android demos.

Setup

- 1) Download and install the Eclipse IDE (<http://www.eclipse.org/>), the Android SDK, ADT plugin for Eclipse, and the Android NDK (<http://developer.android.com/sdk/index.html>).
 - a) For Windows use, also download and install Cygwin (<http://www.cygwin.com/>). During Cygwin setup, make sure to include the "Devel -> make" package.
- 2) Download the Foxit embedded SDK Package.
- 3) Extract the provided fpdfemb_android_examples.zip to any directory.

- 4) Place the Foxit embedded SDK library and header files in
fpdfemb_android_examples/demos/bin and include directory.
- 5) Build the NDK layer.
- a) Open the Android.mk makefile in fpdfemb_android_examples/demos/demo(like
"demo_view")/jni/ in a text editor and fill in the Foxit library name in the area
designated for LOCAL_LDLIBS, dropping the lib prefix:

The demo is shipped as:

LOCAL_LDLIBS += \$(LOCAL_PATH)/../bin/**# fill in library name here**

To add downloaded **libfoxit.a** from step 1, fill in as:

LOCAL_LDLIBS := \$(LOCAL_PATH)/../bin/**libfoxit.a**

If the library provide is not named "libfoxit.a" please adjust accordingly.

- b) Open Cygwin (Windows), or a terminal (Linux based), and navigate to the
fpdfemb_android_examples/demos/demo(like "demo_view") directory. Run
"ndk-build -B" to build the NDK/JNI layer.

Example:

me@myStation /myProjectPath/ > ndk-build -B

This assumes that the ndk directory is part of the \$PATH environment variable. The command can also be qualified with the path to the NDK directory.

- c) The "ndk-build" script will automatically place the finished NDK layer in the form of a
shared object (.so) in the fpdfemb_android_examples/demos/demo(like
"demo_view")/libs/armeabi/ directory.

- 6) Import the project into Eclipse through File->Import->Existing Project into
Workspace, and choose the directory where the demo was extracted.

- 7) Eclipse builds automatically.

- a) If the NDK/JNI code is changed, it will need to be rebuilt by following steps 5b
and 5c. After rebuilding the Eclipse project must be cleaned (Project -> Clean) to
allow Eclipse to rebuild your sample. Hairy and unwanted things can occur if this
is not done.

NOTE: If you encounter this error message in the "Console" tab in Eclipse,

**ERROR: Unable to open class file [full path to extracted demo files]\gen\com\[foxit
demo]\frontend\R.java: No such file or directory.**

Try regenerating the entire \gen folder by making a change to one of the files. For example,

- a) In Eclipse, click on /res/layout/main.xml
 - b) Make the following change and save it,
Android:layout_height="fill_parent" to
Android:layout_height="fill"
 - c) Now change it back to "fill_parent" and save it. This results in no change to main.xml but you should have generated a new /gen folder.
 - d) The demo project should build now.
 - e) If the project does build try Step 7a to clean the project resources.
- 8) Push the finished foxitSample apk to a device/emulator.
- a) Make sure you have a device/emulator ready either by firing off an Android Virtual Device or having an Android phone/tablet plugged in with Settings->Applications->Development->USB Debugging enabled.
 - b) In Eclipse, choose Run->Run to push the foxitSample onto the device. The sample will automatically launch.

Note: If you encounter this error message in the "Console" tab in Eclipse,
"Android requires .class compatibility set to 5.0. Please fix project properties"

Try fixing the project properties. Right click on the demo project, select Android Tools, and then select Fix Properties.

Demo Functionalities

Render the page as an FS_BITMAP object and get the buffer, size and stride data of the object. Then draw the object in the client areas of the platforms.

Initializing the environment

//Initialize the memory, initialize the library and unlock the SDK with given s/n.

```
EMBJavaSupport.FSMemInitFixedMemory(initMemSize);  
EMBJavaSupport.FSInitLibrary(0);  
EMBJavaSupport.FSUnlock("XXXXXXXXXX",  
"XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX");
```

Initializing data

//Make sure the document handler and the page handler exist.

```
docHandle = EMBJavaSupport.FPDFDocLoad(fileAccessHandle,password);
pageHandles[pageIndex]=
EMBJavaSupport.FPDFPageLoad(docHandle,pageIndex);
```

Create the bitmap buffer

//Create a bitmap using SDK function and whiten its background,

```
int dib = EMBJavaSupport.FSBitmapCreate(bm.getWidth(),
bm.getHeight(), 7, null, 0);
EMBJavaSupport.FSBitmapFillColor(dib,0xff);
```

Render the bitmap buffer

//Calculate the size of rendering area and render this area,

```
float scaledWidth = bm.getWidth() * xScale;
float scaledHeight = bm.getHeight() * yScale;
EMBJavaSupport.FPDFRenderPageStart(dib,pageHandles[pageIndex],sta
rtX,startY,(int)scaledWidth,(int)scaledHeight,rotate,flags,rect,p
auseHandler);
```

Generate system compatible bitmap

//Achieve byte string from the system bitmap so that the Android system can use it.

```
byte[] bmpbuf=EMBJavaSupport.FSBitmapGetBuffer(dib);
ByteBuffer bmBuffer = ByteBuffer.wrap(bmpbuf);
bm.copyPixelsFromBuffer(bmBuffer); //Bitmap bm;
```

Render the Android view

//Render the page.

```
imageView.setImageBitmap(myDoc.getPageBitmap(0, display.getWidth(),
```

```
display.getHeight(), xScaleFactor, yScaleFactor, 0)); //gets the  
image from the previous step and attaches the image with the screen  
imageView.invalidate();
```