Getting Started with Android Development – Zebra Android Link-OS™ SDK – Android Studio

Overview

This Application Note describes the end-to-end process of designing, packaging, deploying and running an Android application using the Zebra Link-OS™ Multiplatform SDK and Smartphone Utility.

The sample code used in this Application Note is from the sample code article Connectivity Demo.

Target Audience

The information in this document assumes that you have technical competence with Microsoft Windows, Java development with the Android Studio Integrated Development Environment (IDE), and core programming concepts and rationales.

For information on joining the ISV Program, contact a Program Manager in your area.

System Prerequisites

This Android SDK requires the following system prerequisites.

Java Development Kit (JDK)

To develop Android applications, you must install the latest version of the JDK. Download the JDK.

Android Studio V1.2.1.1

This guide uses the Android Studio IDE which comes with the Android SDK. The Android SDK is distributed free of charge by Google. It contains sample code, libraries, emulators and tools to help you build applications for the Android platform. Download the IDE.

Android Mobile Device

While the Android Emulator included in the Android SDK satisfies most of the anticipated requirements for developing Android applications, we recommended that you test all development with a physical device.

Note: The Zebra Link-OS Multiplatform SDK supports various printers and mobile devices.

Zebra Link-OS Multiplatform SDK

The Zebra Link-OS Multiplatform SDK contains all the required components to develop applications for Zebra label printers.

The SDK includes a Java library (.jar), which provides the means to scan for, connect to, and print on Zebra label printers. Download the Zebra Link-OS Multiplatform SDK, find system prerequisites, and learn more here.

In order to fully test the application you create, you must have a network-enabled Zebra printer capable of understanding ZPL or CPCL.

Bluetooth connectivity is available through the Zebra Link-OS Multiplatform SDK and covered in other available tutorials.

Check out all of the Zebra label printers.

Zebra Network-Enabled Label Printer

In order to fully test the application created during the course of this tutorial, you must have a network-enabled Zebra printer capable of understanding ZPL or CPCL.

Bluetooth connectivity is available through the Zebra Link-OS Multiplatform SDK and covered in other available tutorials.

Zebra Link-OS Multiplatform SDK supports various printers and mobile devices.

Installations

You must install all of the items listed in the “System Prerequisites” section.

Note: The default installation options for all these prerequisites are satisfactory.

Creating Your Zebra Android Connect and Print Application Project

You will be guided on how to create a mobile application that allows a handheld device to establish a Bluetooth or TCP/IP connection with a Zebra mobile printer.

To begin, launch **Android Studio** and locate your desired workspace path.

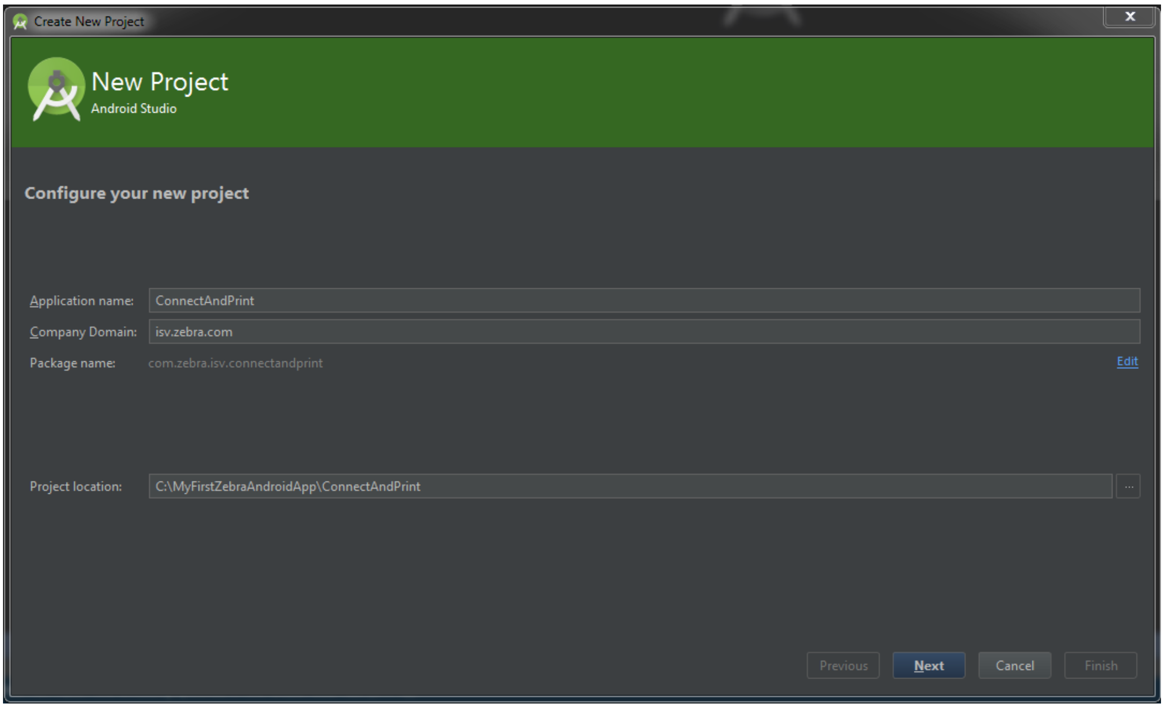
This section includes instructions on how to:

- Create an Android Project
- Design Your Zebra Android Connect and Print Application
- Create Java Helper Classes
- Update ConnectivityDemo.java and Include the Zebra Multiplatform SDK Library for Android
- Run Your Zebra Android Connect and Print Application
- Deploy Your Mobile Application

Create an Android Project

1. Go to File -> New -> Android Application Project.
2. Type **isv.zebra.com** for the Company Domain.
3. Verify that the Package name is "com.zebra.isv.connectandprint".
4. Choose the Project location.

   Note: For the purpose of this application note, we use "C:\MyFirstZebraAndroidApp\ConnectAndPrint".



Configure your new project

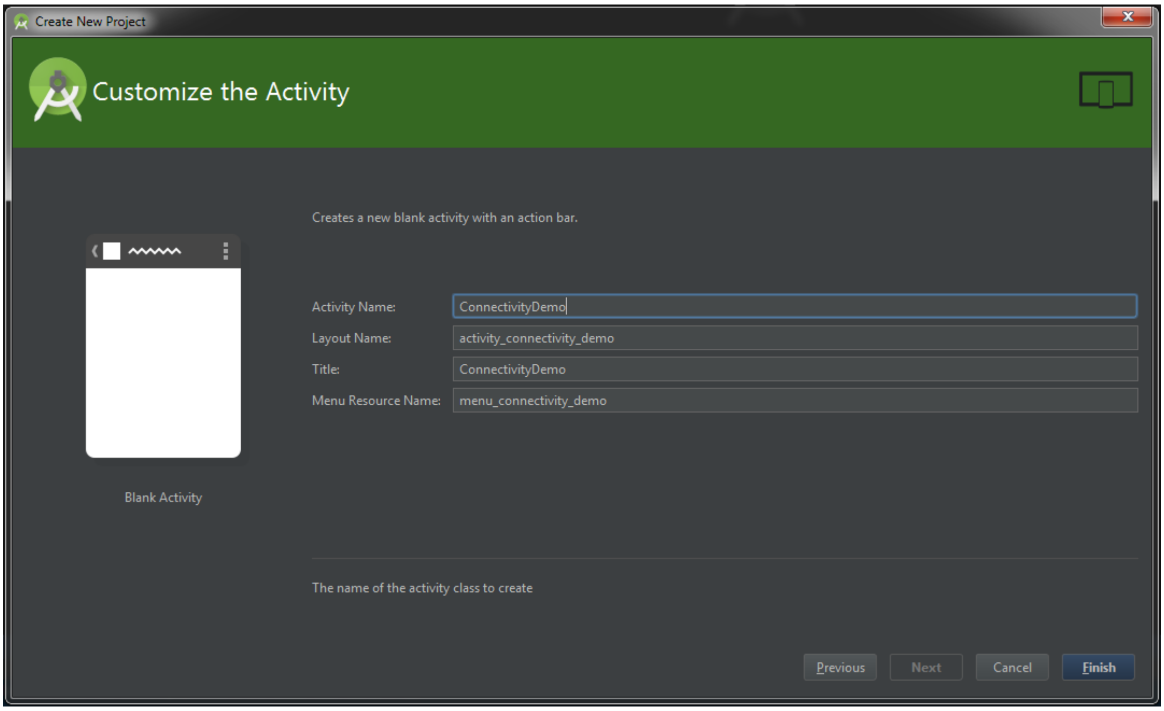
Application name: ConnectAndPrint

Company Domain: isv.zebra.com

Package name: com.zebra.isv.connectandprint

Project location: C:\MyFirstZebraAndroidApp\ConnectAndPrint

5. Click **Next**, leaving the next windows at default, until the Blank Activity window appears.



Creates a new blank activity with an action bar.

Activity Name: ConnectivityDemo

Layout Name: activity_connectivity_demo

Title: ConnectivityDemo

Menu Resource Name: menu_connectivity_demo

The name of the activity class to create

6. In the Customize the Activity window, change the activity name to **ConnectivityDemo** and click **Finish** to exit the window.

Design Your Zebra Android Connect and Print Application

1. Define the variables and values to use in the layout (xml) files later.
   - In the Package Explorer, open strings.xml in the res/values folder.
   - Replace the contents of the “String.xml” file from the “Strings.xml” file from the sample article Connectivity Demo.

   For information on joining the ISV Program, contact a Program Manager in your area.

   This source code defines the Application Name that appears on the GUI, as well as the color Red for the status bar background when no printer is connected. For more information, go to String Resources.

   - Save and close the file.
2. Define the layout.
   - In the Package Explorer, expand the res/layout folder. Delete activity_connectivity_demo.xml.
   - In the same folder, create 2 files, “connection_screen_with_status.xml” and “connection_screen.xml”.
   - Replace the contents of the two files with the contents of the “connection_screen_with_status.xml” and “connection_screen.xml” files from the sample article Connectivity Demo.

     “connection_screen.xml” defines a layout with elements for Bluetooth and TCP/IP input. “connection_screen_with_status.xml” is similar to the above, with the exception of an additional connection status bar at the top.

Note: For more information, click Layout Resource.

- Save and close the files.

3. Update the AndroidManifest.xml.

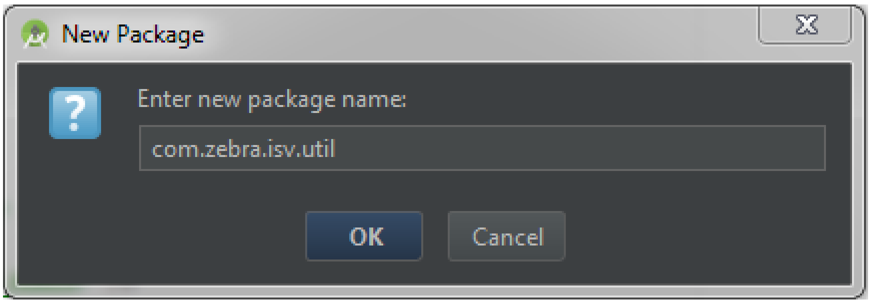
    - Replace the contents of the “AndroidManifest.xml” file with the contents of the “AndroidManifest.xml” file from the sample article Connectivity Demo.

Note: The AndroidManifest.xml contains critical information to build the project correctly,

4. Go to res/menu folder to delete the connectivity_demo.xml file settings menu.

Create Java Helper Classes

1. Right click on src, hover over new and click **Package** to create a new Java Package.
2. Choose ...\app\src\main\java as the destination directory and name the package “com.zebra.isv.util”.



New Package

Enter new package name:

com.zebra.isv.util

3. Create a file “DemoSleeper.java” in the “com.zebra.isv.util” package and replace its contents from the “DemoSleeper.java” file from the sample article Connectivity Demo.

    This file contains a static sleep method that ConnectivityDemo.java uses.

4. Create a file “SettingsHelper.java” in the “com.zebra.isv.util” package and replace its contents from the “SettingsHelper.java” file from the sample article Connectivity Demo.

    This file contains methods used in ConnectivityDemo.java to persist Bluetooth and TCP/IP settings for subsequent recall.

Update ConnectivityDemo.java and Include the Zebra Multiplatform SDK Library for Android

1. Replace the source code in ConnectivityDemo.java with the contents of the “ConnectivityDemo.java” file from the sample article Connectivity Demo

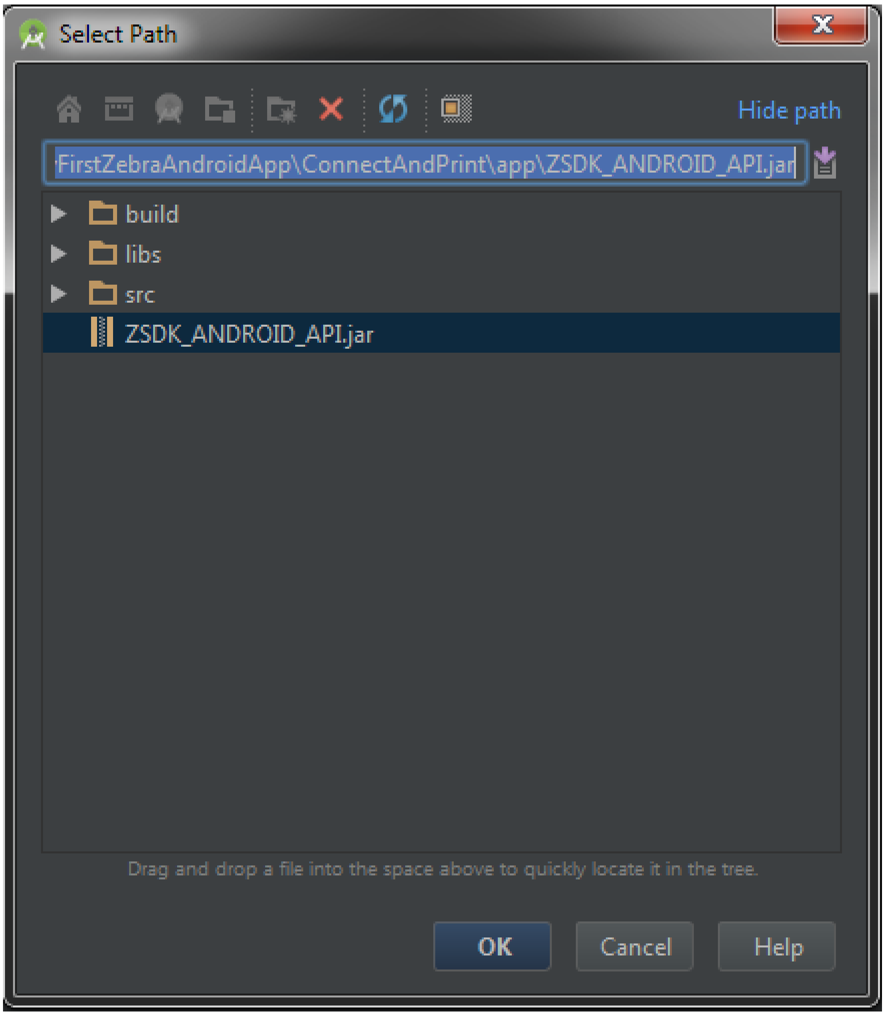
    This class is the starting point of the Android application.

2. Browse to the location of the “ZSDK_ANDROID_API.jar” file and copy it to the 'app folder'.

Note: The default installation path of this file should be “C:\Program Files\Zebra Technologies\link_os_sdk\android\v2.8.2148\lib\ZSDK_ANDROID_API.jar”.

3. At this point, you should see many errors. To resolve the dependencies, go to File -> Project Structure -> app(under Modules) -> Dependencies.

4. Click on the '+' symbol -> File Dependency and choose the ZSDK_ANDROID_API.jar.



FirstZebraAndroidApp\ConnectAndPrint\app\ZSDK_ANDROID_API.jar

Drag and drop a file into the space above to quickly locate it in the tree.

Run Your Zebra Android Connect and Print Application

1. Build and run the application.
   - Click the green play button at the top, or alternatively, Run -> Run app. and click **OK** to launch from emulator.
   - To run the application, unlock the screen on the emulator. To do this, drag the green lock icon to the right of the screen.

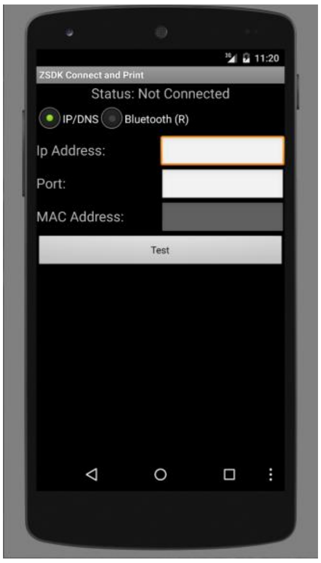
This runs the application in an Android emulator. The emulator might take a while to start up, depending on the available computer resources and hardware.

Congratulations! You now have your first Zebra mobile application running on an emulator!

2. To exit the application, click **Back** on the emulator.

   To continue development, leave the emulator running as it takes considerable amount of time to start up.

   Note: The Bluetooth and TCP/IP connectivity may not be correctly emulated. Test the application on a physical device, which will be covered in the next section.



Status: Not Connected

● IP/DNS ○ Bluetooth (R)

Deploying Your Mobile Application

After your project is complete, you can export it into an Android Application Package (.APK) file for deployment on an Android Device.

1. Go to Build -> Generate Signed APK.

   A dialog appears asking you to enter keystore information.

2. Type the keystore information.

   If you do not have this information, type "C:\AndroidKeyStore" (or any other desired folder) as Location, and then type your desired password.

3. Fill up the Key Creation form and click **Next.**

   Note: You must enter at least the Alias, Password, Confirm, Validity and First and Last Name fields. If you do not, you are not allowed to proceed.

   The .APK file should be in your "C:MyFirstZebraAndroidApp\ConnectAndPrint\app" folder. Copy the .APK file to your device and install your application.

See Also

- For any further information, sample code and solutions or to request further content, visit the Zebra Developer Portal.

Document Control

Version	Date	Description
1.0	July 2015	Initial Release
2.0	2nd October 2017	Updated URLs

Disclaimer

All links and information provided within this document are correct at time of writing
Created for Zebra Global ISV Program by Zebra Development Services.