# Appendix D – Building a Standalone Application

|  |
| --- |
| Note: A BASIC! user, Nicolas Mougin, has created an automated tool for generating standalone applications. This tool can be downloaded from:  <http://mougino.free.fr/rfo-basic-app-builder.zip>  Using Mr Mougin’s tool avoids having to do all of the following.  If you have any questions or problems with this tool you can contact Mr. Mougin and other users of the tool at the BASIC! forum in this thread:  <http://rfobasic.freeforums.org/rfo-basic-app-builder-f20.html> |

## Introduction

This document will demonstrate how to create a standalone application from a BASIC! program. The resulting application does not need to have BASIC! installed to run. It will have its own application name and application ICON. It may be distributed in the Android Market or elsewhere. The process involves setting up the Android development environment and making some simple, directed changes to the BASIC! Java source code.

## License Information

BASIC! is distributed under the terms of the [GNU GENERAL PUBLIC LICENSE](http://laughton.com/basic/license.html). An implication of this is that the source code for your application must be provided to anyone who asks.

## Before You Start

Run the Sample Program, f99\_my\_program.bas.

We are going to turn this program into a practice APK.

## Setting Up the Development Environment

1. Download and install the latest version of the Java Development Kit (JDK). Find this by Goggling “java jdk download” and going to the listed oracle.com download site. Do not download from any other site. Note: The JDK download includes the Java Run Time Environment (JRE) which is also needed.

2. Download the Android development SDK installer. Start at: <http://developer.android.com/sdk/index.html>

3. Continue with: <http://developer.android.com/sdk/installing/index.html>  
Execute the SDK installer. Install the recommended items.

(Windows: If you get messages saying that nothing was installed: close the SDK Manager. Go to Start->All Programs->Android SDK Tools. Right click on SDK Manager and select run as administrator.)

If you get a request to start the ADB, do it.

Close the SDK Manager when all packages have been installed.

4. Download and install the Eclipse Integrated Development Environment (IDE) from: <http://laughton.com/basic/eclipse>. Choose the 32 bit or 64 bit version of depending upon your development computer. You can use other versions of Eclipse but these instructions might not work and thus you are on you own.

5. Continue at: <http://developer.android.com/sdk/installing/installing-adt.html>

Start Eclipse. **Carefully** follow the instructions for installing the ADT.

Accept any security warnings and allow Eclipse to restart as per instructions.

Continue with “Configure the ADT.”

Download the **latest API** level available no matter what API the level of your device is.

Ignore “Updating the ADT” and the rest of the page.

Change API level checking from Error to warning:

Select Window->Preferences->Android->Lint Error Checking

Find "NewApI" and click on it.

In the dropdown list, change the Severity from Error to warning.

Press Apply

## Download the BASIC! Source Code

Go to: <http://laughton.com/basic/versions/index.html> and click on the highest version number.

Look for the section heading, “**Download BASIC! Source Code.**“ Click on the “**here**” to download the latest “Basic.zip”

Unzip this file into a folder named, for example, CatsApp. You should use a different folder for each new APK that you create from Basic.zip.

## Create a New Project in Eclipse

Select: File -> New -> Project…->Android->Android Project from Existing Code

In the Import Projects dialog box, browse to the CatsApp folder(directory) than contains the unzipped Basic folder.

Check the project: com.rfo.myapp.Basic. Do not check any other boxes. Click Finish.

In the Package Explorer window on the left, right click on com.rfo.basic.Basic.

Select Properties (at the bottom of the list).



Check the highest level of the Android OS available in this list. Do this without regard to the level of OS in your device(s).

Click Apply then OK.

From the menu, Select: Project->Clean.

Check “com.rfo.basic.Basic”

Press Ok.

The Basic source is now ready for making an APK.

## Rename the package

The package name is what makes your application different from every other application that runs on Android devices. No matter what you name your application, it is the package name that Android uses to identify your particular application.

In the Package Explorer at the left side of the page tap the name “rfo.com.basic.BASIC”.

Next Select: File->Rename.

Note: In the remainder of this tutorial, we are assuming are application is about cats thus we are using the name “com.rfo.cats” You should, for you own APK, choose a name that matches your application(s).

Enter the new name for the package. Let’s assume that your application will be named “cats.” Change the name of the package to “rfo.com.cats.BASIC”

Make sure the Update References check box is checked.

Press OK.

In the package explorer, click and open the “src” hierarchy as shown:



Select the “com.rfo.basic” as shown.

Select File->Rename and rename it to “com.rfo.cats”. Make sure the Update References box is checked.

From the Menu Bar, select: Search->File.

Fill in the dialog like this:



Press Replace.  
The Replace Text Matches Dialog Box will be shown.  
Enter “com.rfo.cats” in the “With:” field.  
Press Ok

A Launch Configuration Change dialog box will be displayed. Press Yes.

Finally, Select: Project -> Clean  
Press OK on the Clean Dialog Box.  
At this point the package has been successfully renamed. Next we will create a practice APK that you then use to make your own APK.

## Modifications to Basic.java

In the Package Explorer, Expand “com.rfo.cats” and then double click on Basic.java. Basic.java will be opened in the window on the right.



Change: AppPath = “rfo-basic” to: AppPath = “rfo-cats”

“rfo-cats” the directory on the SDCARD where your files will be stored, if you choose to have a directory for file for you application. Even if you do not choose to create directories for your application, you should make this change. It has implications in other parts of the code.

Change: *isAPK* = **false** to: *isAPK* = **true**;

Since this practice application does not need any directories, change:

*apkCreateDataDir* = **true**; -> *apkCreateDataDir* = **false**;

*apkCreateDataBaseDir* = **true**; -> *apkCreateDataBaseDir* = **false**;

Finally, we need to change the files to be loaded onto the SDCARD. Our practice application uses the file “meow.wav.” We need to load this file onto the SDCARD for Standard Basic. We do not need to load it onto the SDARD for an APK. The APK will automatically read the file from res.raw. We do not need to load any files for this APK. To indicate that no files are to be loaded, remove them as shown below.



If you APK actually needs files to be loaded to the SDCARD, name them here. Be sure to allow the creation of the Data directory.

Files with the extension “.db” will be loaded into the /databases/ directory.

Save the changes to Basic.java.

## Changing the APK Name

We are now going to change the name of the Application as it will appear on your Android device. Expand res.values and double click on strings.xml.



Change: <string name=*"app\_name"*>BASIC!</string>

To: <string name=*"app\_name"*>Cats</string>

Note: If you application uses the version$() function, this is where that function gets that value.

Save the changes.

## Testing the APK

We are now ready to test this practice APK.

The first thing you will do is to create a Keystore. The Keystore is used to sign your application. Google Play requires this signing. Android devices will not install unsigned APKs. You will use this one Keystore for all you APKs. Preserve and protect it. You will not be able to update you APK without it.

For more information about the Keystore and signing, see http://developer.android.com/tools/publishing/app-signing.html

On the left hand side of the screen, right click on “com.rfo.cats.Basic”  
Select: Android Tools->Export Signed Application Package  
Select Next in the Project Checks dialog box.  
Select “Create New Keystore.”

Provide a name and location for the Keystore.

Provide a pass word and confirm it.

Click next.

Fill out the Key Creation dialog.

Pick any name for an Alias.

Enter 25 for Validity (Years)

Click Next

In the Destination and Key/Certificate Checks dialog,

browse to the folder where you want to put the APK.

Name the APK “Cats.apk”

Click Finish

Now, install and run Cats.apk

The APK will have the BASIC! icon. The name below the icon your device will be Cats. Double click Cats to run it.

If you have reached this point successfully then you are ready to customize to create the APK for your application.

Start over with a new copy of Basic.zip but use names and information particular to your application and then continue below.

## Installing A BASIC! Program Into the Application

Outside of Eclipse  
    Open you BASIC! program in a text editor  
    If your program uses INCLUDE files, you should text-merge the included files into a single text file.  
    Copy the entire program into the Clipboard  
  
In the Package Explorer on the left side,  
    Open res  
    Open raw  
    Double click on f99\_my\_program  
    The file will be opened  
  


Tap inside the window on the right.  
Press Edit - > Select All  
Press Edit -> Paste  
Press the "X" on the f99\_my\_program tab to close the file.  
Press "Yes" on the Save Resource dialog box.

### Adding Your Image and Audio Files

Looking into the Package Explorer under res.raw you will see a lot of files. The only one of these files that you will need in your APK is f99\_my\_app which we changed above. To delete the files, use shift-click to select the two blocks of files around f99\_my\_program and press the delete key for each block. If you have done this correctly, you will have only f99\_my\_program left in res.raw.

You can now proceed to add your application specific image, audio or other files.

Outside of Eclipse, right click on the file you want import. Select copy. Back in Eclipse, right click on raw. Select paste. The file is now installed in your APK. When you open the file in your program, it will be read directly from res.raw.

## Application ICONs

Android specifies that Application icons must be provided in three specific sizes: low dpi (36x36 pixels), medium dpi (48x48 pixels) and high dpi (72x72 pixels). The icons must also be .png files. There are tens of thousands of free icons available on the web. Most image editing programs can be used to re-sample the icons to the proper sizes and to save them as .png files. If you are not going to put your application on the Android market then you do not really need to worry about getting this exactly right.  
  
To get your icon into your application, in res, open drawable-ldpi, drawable-mdpi, drawable-hdpi.  
  
  
  
For each of the icon sizes:  
  
    Outside of Eclipse, copy the icon file  
    In Eclipse, right click on the appropriate drawable- for the copied icon's size  
    Select Paste  
    Right click on the icon.png file and delete it.  
    Select the newly pasted icon and rename it to "icon.png" by selecting File -> Rename.  
  
Yes, it is tedious work.

## Setting The Version Number and Version Name

If you are going to put the application on the Android Market, you will need to change the version number and name for each new release  
  
Change the Versions information by double clicking on the AndroidManifest.xml file.  
  
  
  
Make the appropriate changes to android:versionCode and android:versionName, click the X in the tab to close and save the changes.

If you want to use the BASIC! version$() function to have your program read your version number, you will also have to change the version number in res->values->strings.

## Permissions

BASIC! uses many features about which the APK user is warned and must approve. Your particular APK may not need all or any of these permissions. The permission notifications are contained in the AndroidManiest.xml.

The permission notifications look like:

<uses-permission android.name=”…….

Look them over. If you feel that your APK does not need them then delete or comment them out.

Please do keep the vibrate permission. If you do not have this permission, you APK may crash when it exits. This is what that permission looks like.

<uses-permission android:name=*"android.permission.VIBRATE"* android:required=*"false"*></uses-permission>

If your application uses the SDCARD, do not comment out:

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

Be sure to test your APK after doing changing permission.

## Launch at device boot

You APK can be set up to automatically launch just after the Android device has booted. This is accomplished by changing a parameter in AndroidManifest.xml.

Find the code line (around line 72) :

<receiver android:enabled="false" android:name=".BootUpReceiver">

and change it to:

<receiver android:enabled="true" android:name=".BootUpReceiver">

## Finished

Create you finished APK in the same way we created the practice APK.

Now that was not too bad, was it?