# 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. Since Google changes the Development Environment every so often, this procedure does not necessarily reflect the latest version of the ADT.

## License Information

BASIC! is distributed under the terms of the [GNU GENERAL PUBLIC LICENSE](http://laughton.com/basic/license.html). The license requires that the source code for "derivative works" be made available to anyone who asks. The author of BASIC! interprets this to mean that the license applies only to derivatives of the BASIC! *source interpreter code*. It does not apply to source code for BASIC! applications, i.e., code that you have written using the BASIC! language.

## Before You Start

Run the sample program **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 Googling "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.

1. 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.
2. Continue at: <http://developer.android.com/sdk/installing/installing-adt.html>
3. Start Eclipse. **Carefully** follow the instructions for installing the ADT.
4. Accept any security warnings and allow Eclipse to restart as per instructions.
5. Continue with "Configure the ADT."
6. Download the **latest API** level available no matter what API the level of your device is.
7. Ignore "Updating the ADT" and the rest of the page.
8. Change API level checking from Error to warning:
   1. Select Window->Preferences->Android->List Error Checking
   2. Find "NewApI" and click on it.
   3. In the dropdown list, change the Severity from Error to warning.
   4. Tap Apply

## Download the BASIC! Source Code from the GitHub Repository

1. Go to: <https://github.com/RFO-BASIC/Basic>.
2. Find the branch/tag selection button:to select the tagged release version you want. If you do not select a tag, you will get the source for the latest development build. The latest development is not guaranteed to be stable.
3. Click the “Download ZIP” button: The name of the downloaded ZIP file depends on which tag you selected in the previous step.
4. Put the unzipped source files into a folder. For this exercise, name the folder “Cats”. You should use a different folder for each new APK that you create.

– OR –

## Download the BASIC! Source Code from the Legacy Archive

1. Go to: <http://laughton.com/basic/versions/index.html> and click on the highest version number.
2. Look for the section heading, "**Download BASIC! Source Code.**" Click on the "**here**" to download the latest "Basic.zip"
3. Older versions of this file contain only the BASIC! source code. More recent versions also contain a copy of the Android-loadable Basic.apk file and a copy of this manual.
4. Put the unzipped source files into a folder. For this exercise, name the folder “Cats”. You should use a different folder for each new APK that you create.

Note: In the remainder of this tutorial, we assume our application is about cats, thus we are using the name "Cats". You should, for you own APK, choose a name that matches your application.

## Create a New Project in Eclipse

1. Select: **File -> New -> Project…->Android->Android Project from Existing Code**.
2. In the **Import Projects** dialog box, browse to the folder where you put the BASIC! source files.
3. Under **Projects to Import**, click the project you are importing. Make sure it is checked.
4. Under **New Project Name**, change the project name from “Basic” to “Cats”. Click **Finish**.
5. In the **Package Explorer** window on the left, right-click on **Cats** and select **Properties** (near the bottom of the list).
6. In the Properties dialog box, select **Android**: 
7. 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).
8. Click **Apply** then **OK**.
9. From the menu, select: **Project->Clean**.
10. Check **Cats**.
11. Click **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.

There is an Android tool that does most of the work for you:

1. In the **Package Explorer**, right-click on **Cats** and select  
   **Android Tools -> Rename Application Package**.
2. Under **Enter new package name:** change “com.rfo.basic” to “com.rfo.cats”.
3. A **Refactoring** window opens. Click **Finish**.
4. A **Launch Configuration Change** dialog box is displayed. Click **Yes**.

You still need to manually change some things the Android tool missed. First, tell Eclipse about the package change.

1. In the **Package Explorer**, click and open the **Cats -> src** hierarchy as shown below.
2. Select the **com.rfo.basic** package.
3. From the Menu Bar, select **File->Rename** to open the **Rename Package** dialog.
4. In the **New name:** field, enter “com.rfo.cats”.
5. Make sure the **Update references** box is checked and click **OK**.  
   
6. You may see a warning that the package already exists in folder ‘gen’. Click **Continue**.
7. From the menu, select: **Project->Clean**.
8. Check **Cats** and click **OK**.

Finally, use the search-and-replace dialog to update any remaining internal references to the package name.

1. From the Menu Bar, select: **Search->File**.
2. Fill in the dialog like this:  
   
3. Click **Replace**. The Replace Text Matches Dialog Box opens.
4. Enter “com.rfo.cats” in the **With:** field, and click **OK**.

The project should build automatically. 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 setup.xml

In the **Package Explorer**, expand **Cats/res/values** and then double-click **setup.xml**. The file opens in the window on the right. The Eclipse Editor does not know how to modify all of the properties in this file, so click on the **setup.xml** tab to display the actual XML text, as shown below. In this view, you can also see the comments describing the values you can change for your application.

Be sure you change only values, not names. Names are shown as blue text in quotes. Values are shown as black text with no quotation marks. If you change a name, Java can not find the item to get its value.

Change the value of **“app\_name”** from **BASIC!** to **Cats!**. This is the name of the application as it will appear on your Android device.

The VERSION$() function gets its value from the item named **“version”**. If your application uses VERSION$(), this is where you set the value you want it to return.

Change the value of **“app\_path”** to **rfo-cats**. This will be the directory on the SDCARD where your files are stored, if you choose to have a directory for files for your application. Make this change even if you do not choose to create directories for your application. It has implications in other parts of the code.

Change the value of **“is\_apk”** to **true**.

The items **“apk\_create\_data\_dir”** and **“apk\_create\_database\_dir”** are flags that control whether directories are created under **“app\_path”** for your application’s files. Since this practice application does not need any directories, change the values of both to **false**.

The item **“load\_file\_names”** is a list of files that you want loaded to the SDCARD. This practice application uses the sound clip **meow.wav**. Running under standard BASIC!, the program would use the file rfo-basic/data/meow.wav. As a standalone application, it can use a file image built into APK. Since the practice application is not using a real file, you can leave the **“load\_file\_names”** list empty.

The item **“my\_program”** is the name of the BASIC! program you want your application to run. This will be explained in the next section.

The item **“run\_name”** is the default title of the Console. Your program can set the Console title, but you may want to change the default title here.

Save the changes to setup.xml. 

Note: If you do want to load files to the SDCARD, you must allow the creation of the data directory (set **“apk\_create\_data\_dir”** to **true**) and put the file names in the **“load\_file\_names”** list like this:   
  
The files will be loaded from the assets folder. This will be explained in the next section.

At this point, you have modified the source files of the BASIC! interpreter so it can be packaged into a standalone application. You can build and run it, if you like, but it will display a blank Console. That’s result you get when you build BASIC! into an application but you don’t give it a program to run.

## Files and Resources

Standard BASIC! loads its sample programs and the data files they need from the **assets** folder of the Eclipse project. Android treats the **assets** folder like a file system. At startup, BASIC! simply copies the entire **assets/rfo-basic** folder to the SDCARD.

Your application uses the **assets** folder, too.

1. In the **Package Explorer**, expand **Cats/assets/rfo-basic** and its **data** and **source** folders.
2. Right-click **assets** and select **New -> Folder**.
3. In the **Folder name:** field, enter “rfo-cats/data”.
4. Click **Finish**.
5. In the same way, create “rfo-cats/source”.
6. Drag **assets/rfo-basic/data/meow.wav** to **assets/rfo-cats/data/**.
7. Drag **assets/rfo-basic/source/my\_program.bas** to **assets/rfo-cats/source/**.

Note: the top folder in **assets** must exactly match what you put in the **“app\_path”** item in **res/values/setup.xml**. For this practice program, it is **rfo-cats**.

If you expand your new folders, your **Package Explorer** should look like this:



Your APK does not need anything in **assets/rfo-basic**. Delete the entire folder:

1. In the **Package Explorer**, right-click **Cats/assets/rfo-basic** and select **Delete**.
2. In the confirmation window that appears, click **OK**.
3. You are done making changes! It wouldn’t hurt to do another **Project->Clean** here.

## 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 your APKs. Preserve and protect it. You will not be able to update your APK without it.

For more information about the Keystore and signing, see:

<http://developer.android.com/tools/publishing/app-signing.html>

1. In the **Package Explorer**, right-click **Cats**.
2. Select: **Android Tools -> Export Signed Application Package**
3. In the **Project Checks** dialog box click **Next**.
4. Select **Create New Keystore**.
   1. Provide a name and location for the Keystore.
   2. Provide a password and confirm it.
   3. Click **Next**.
5. Fill out the **Key Creation** dialog.
   1. Pick any name for an Alias.
   2. Enter 25 for Validity (Years)
   3. Click next.
6. In the **Destination and Key/Certificate Checks** dialog,
   1. Browse to the folder where you want to put the APK.
   2. Name the APK "Cats.apk"
   3. Click next.
7. 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 the icon to run your application.

If you have reached this point successfully then you are ready to customize 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

You must build your BASIC! program into your application my putting it into the **assets** folder, just as you saw in the practice program. One very simple way is to copy your text into **assets/<your-app>/my\_program.bas**. “**<your-app>**” is the path you named in the **setup.xml** item **“app\_path”**.

1. In the Eclipse **Package Explorer**, expand the **assets** folder.
2. Double-click **assets/<your-app>/source/my\_program.bas**.
3. The file will open in the edit window to the right: 

Now you can edit the file directly in Eclipse. If you prefer, you can open your program outside of Eclipse, copy its contents to the clipboard, and paste it into the Eclipse editor. When you are finished, select **File->Save** from the menu bar, or just close the file by clicking the **X**  on the file tab and **Yes** in the **Save Resource** dialog box.

For more complex projects, here is another way to do it.

1. In the Eclipse **Package Explorer**, expand your program’s **assets** folder.
2. If you have a **my\_program.bas** in your **assets** folder, delete it.
3. In a file browser, browse to the file containing your program.
4. Drag your program from your file browser to your **assets/<your-app>/source** folder in Eclipse.
5. If your program uses INCLUDE files, drag them to the **source** folder, too.

With either method, you now have a program to run. If your program uses graphics, audio files, or other resources, you must put them in your Android **assets** folder, too.

If your program uses data files, drag them to your **assets/<your-app>/data** folder.

If your program uses data base files with the “.db” extension, create a new folder for them called **assets/<your-app>/databases** and drag your file into the new folder.

The result might look something like this:



Notice that the top folder in **assets** matches the **“app\_path”** value, and there is a program in its **source** folder named the same as the **“my\_program”** value.

## Application ICONs

Android specifies that Application icons must be provided in specific sizes: low dpi (36x36 pixels), medium dpi (48x48 pixels), high dpi (72x72 pixels), x-high dpi (96x96), and so on. 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:

1. Outside of Eclipse, copy the icon file
2. In Eclipse, right click on the appropriate drawable- for the copied icon's size
3. Select Paste
4. Right click on the icon.png file and delete it
5. 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/setup.xml**.

## 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, your 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.

## Preferences

There are certain preferences such as screen colors and font sizes that you have set for your application. The preferences that you will get with an APK will be BASIC! default preferences. You can change the default preferences if you wish.

Some preferences are simple check boxes. Other preferences are multiple-choice lists. The one check box preference whose default value that you might wish to change is the Console Lines preference. To change the default from lined console to unlined console:



Open the res.xml hierarchy and double click on settings.xml. In the opened file scroll down to the indicated line and change the "true" to "false". Save the changes.

To change the multiple-choice preferences, open the res.values hierarchy and double click on arrays.xml. Each preference has two blocks. The top block lists the words that will be seen on the Android screen. The second block lists the internal names that correspond to the displayed words.

In the image below:

  
The section marked contains the names and values for the Screen Orientation preference. The top block is the display names. The bottom block is the internal values that correspond to the display name. For example the internal value of "Fixed Reverse Landscape" is 2.

To set a default value for Screen Orientation, we need to go back to settings.xml.



Find the block with the "android:title" (in the "Screen Orientation" section of the ListPreferences). That is the preference name that you see on the Android screen. The default value is in the "android:defaultValue =" line. Here we see the default value for the screen orientation is "0". Looking at the Array.xml file we can see the 0 is the internal name for "Variable By Sensors". To change the default value to "Fixed Reverse Landscape", change the 0 to 2.

The other list preferences follow the same logic.

Note: Be sure to test your application with your chosen preferences before burning them into the APK here.

## Launch at device boot

Your 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 your finished APK in the same way we created the practice APK.

Now that was not too bad, was it?