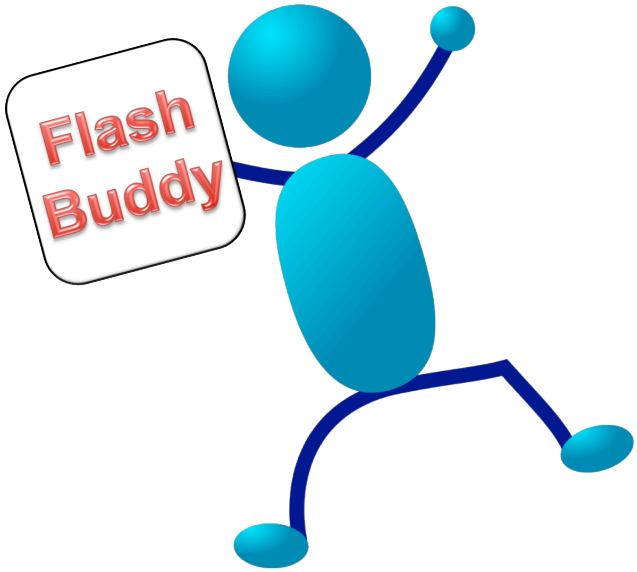


FlashBuddy Flash Card Build & Execution Instructions

Version 1.1

Changelog

Date	User	Change
04/24/14	JLeidel	v.1.1: Adding AVD requirement and setup details for SDCard
03/01/14	JLeidel	v.1.0: Initial Printing

Contents

1 Introduction

1.1 Overview

The following document describes the file format specification for the FlashBuddy card decks. Card decks are described using a predefined XML specification such that we may describe card deck subject matter, individual card layout [front versus back] as well as any plain text data to be included in the card's content. We selected XML in order to permit simple text compression for transferring over low bandwidth connections as well as the ubiquity of parsing on a multitude of platforms.

1.2 Document Assumptions

This document assumes basic knowledge of the Java programming language, the Eclipse Integrated Development Environment and the Android build and execution environment.

1.3 Software Requirements

This document relies upon the latest version of the Android Development Toolkit [ADT]. We rely upon the ADT version with the bundled Eclipse IDE. This ensures that the build paths and emulation environment are correctly installed and initialized.
One may obtain the latest version of the Google ADT from the following website for a variety of platforms:

http://developer.android.com/sdk/index.html

1.4 Typographical Conventions

This document contains the following typographical conventions:
`Bold Fixed Width Font` is used for system-generated output and source code examples.

2 FlashBuddy Installation

2.1 Unpacking FlashBuddy

The FlashBuddy application deliverable is packaged as a gzipped tarball. The deliverable contains all the necessary project files, documentation and source code necessary to rebuild and execute the FlashBuddy application. The basic directory structure for the FlashBuddy application is as follows:

Directory	Description
~/flashbuddy/	The FlashBuddy main directory
~/flashbuddy/docs/	The FlashBuddy documents directory
~/flashbuddy/FlashBuddy/	The FlashBuddy project and source directory. This is the required directory to import into Eclipse.
~/flashbuddy/LICENSE	The FlashBuddy license file
~/flashbuddy/README	A basic FlashBuddy readme file

Users are expected to have or obtain a method by which they may unpack a gzipped tarball. Typical UNIX or Linux platforms will generally have the **`tar`** application available. An example of unpacking the deliverable package using tar is as follows:

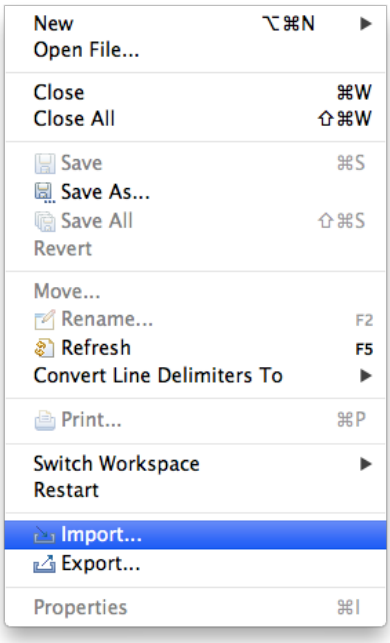
```
$> tar xzvf flashbuddy.project-N.tgz
```

where N is one of 2-5, correlating to the project deliverable 2-5.

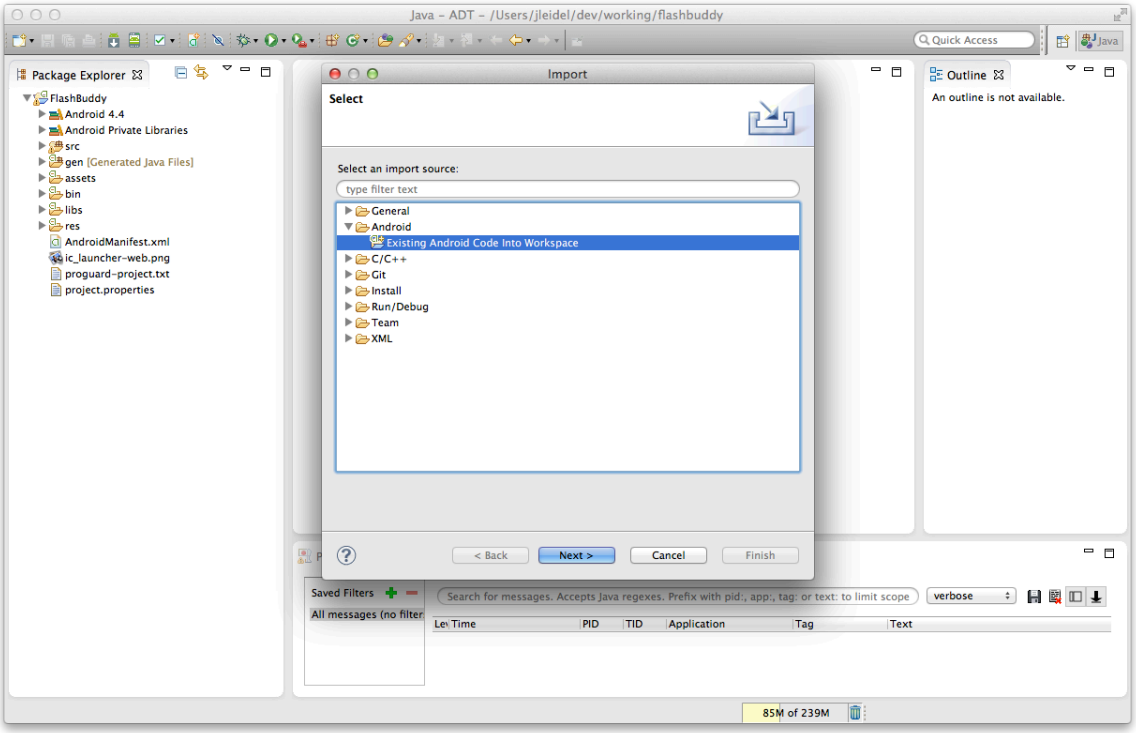
2.2 Importing into Eclipse/ADT

In order to import the project into Eclipse, open the Eclipse Android ADT instance. It is highly recommended to utilize the bundled Eclipse instance with the Android ADT package. Raw Eclipse packages lack specific Android Java required classes and build properties.

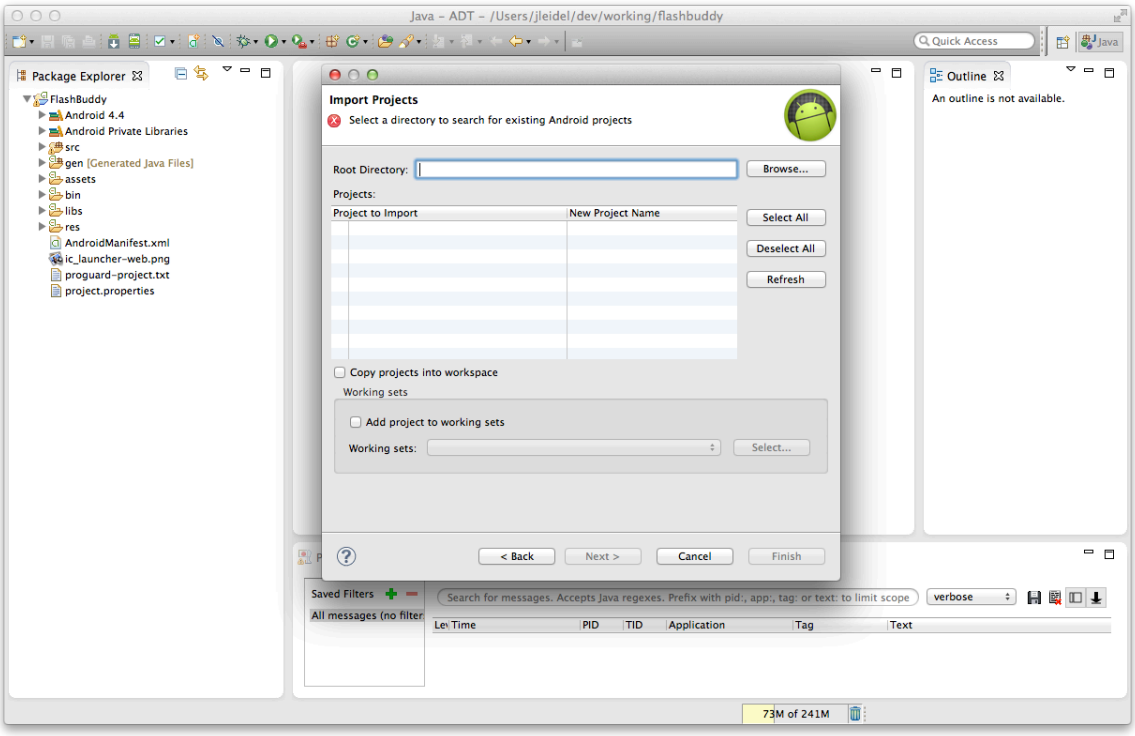
Once the IDE is open to a blank project screen, click the **`File`** menu on the top left. Near the bottom of the **`File`** menu is an option to **`Import`** a new project. Select the **`Import`** option.



Next, select under the Android menu to import **`Existing Android Code into Workspace`** and select **`Next`**.



The next menu will permit you to browse to the Root Directory of the project and select it to import. In this case, the root directory is **`~/flashbuddy/FlashBuddy`**.
Note: The "~/" denotes the directory where you unpacked the gzipped tarball.

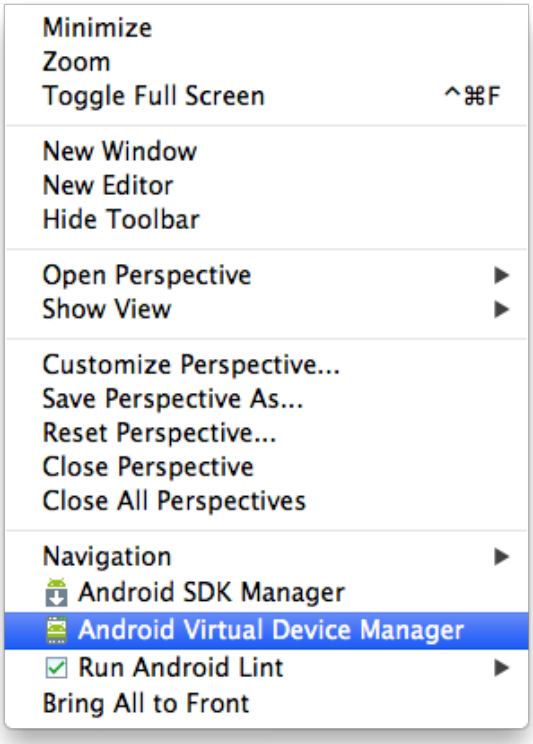


The project should now successfully import into Eclipse.

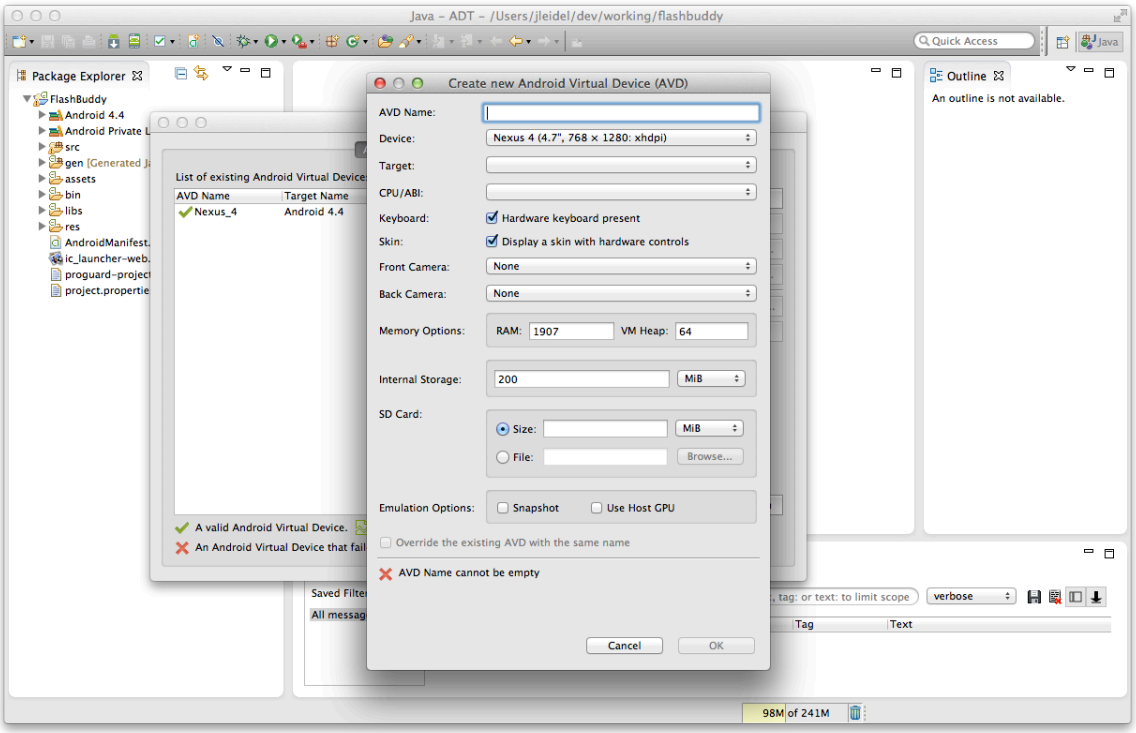
3 FlashBuddy Execution

3.1 Supported FlashBuddy Android Emulators

Once you have imported the project into the Eclipse IDE, you need to initialize a Google Nexus 4 emulation instance. This can be done by selecting the **`Android Virtual Device Manager`** under the **`Window`** menu.



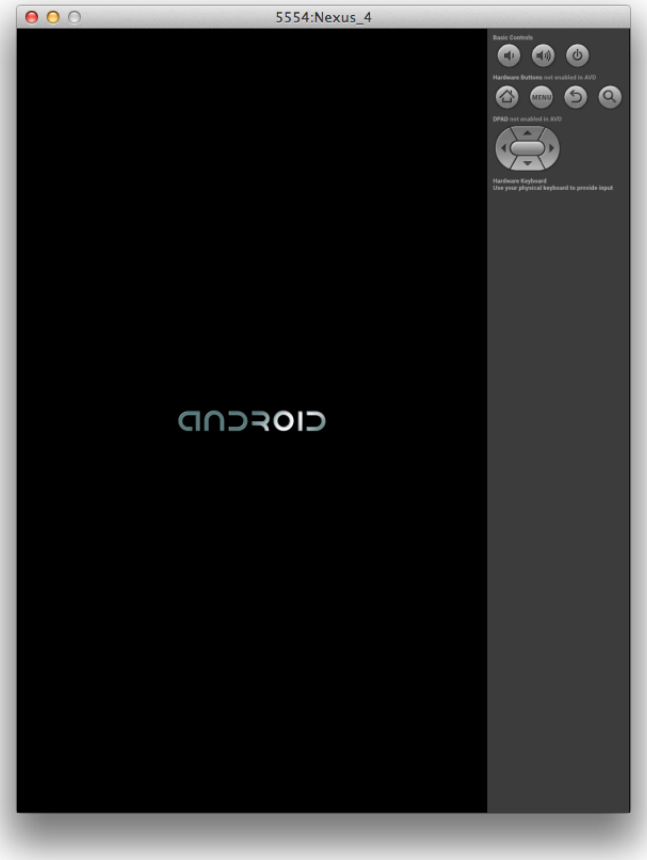
Once in the virtual device menu, create a new Android Virtual Device (AVD) using the Nexus 4 device specification. Enter a simple name in order to delineate this from other AVD's. You must also create an SDCard instance for the target AVD. In the SDCard "size" field, enter a value greater than or equal to 64 [MiB]. The AVD manager will prompt you pre-initialize this SDCard storage. This is desired in order to reduce the overall latency of the emulator. Select "OK" to save the AVD. Once saved, close the virtual device menu.



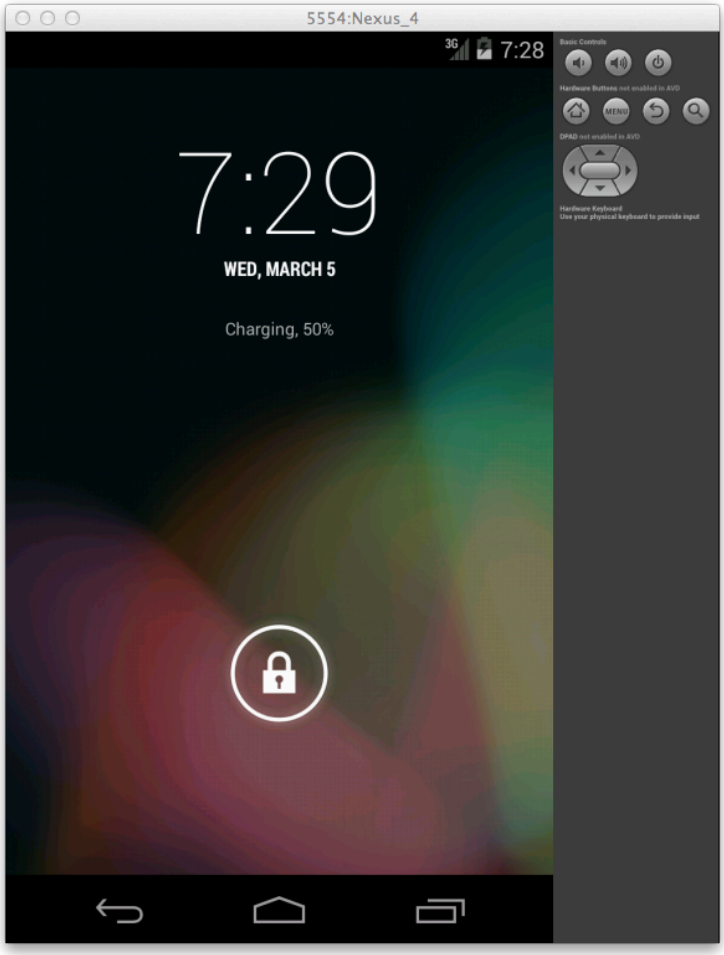
3.2 Executing FlashBuddy

Once the AVD device is in place, select the **Run** command from the **Run** menu. This will build and instantiate a new AVD emulator. The Eclipse IDE will automatically install the FlashBuddy application on the emulated device. Once the emulator has booted [this may require several minutes], you may need to unlock the emulator by swiping the lock button at the center of the screen to the right-hand outer edge of the emulated device screen. At which time, the FlashBuddy application will start automatically.

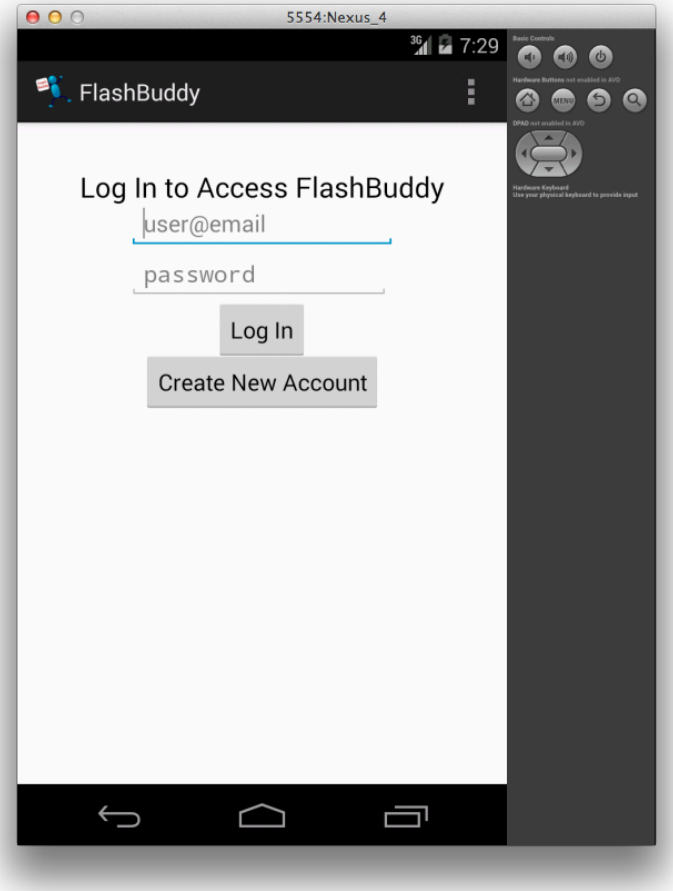
Android emulator booting:



Android emulator lock screen:



Android emulated FlashBuddy application:



3.3 Authenticating to FlashBuddy

The FlashBuddy application has a default username and password for authentication. They are as follows:

Username: **`admin`**
Password: **`password`**

It is expected that we will remove this as we move forth in the project.