# PART I
# Getting Started

- **CHAPTER 1:** Introducing Flash Development for Mobile Devices
- **CHAPTER 2:** Setting Up Your Development Environment
- **CHAPTER 3:** Building and Installing VanillaApp

# 1

# Introducing Flash Development for Mobile Devices

**WHAT'S IN THIS CHAPTER?**

- Discovering Adobe AIR for Android
- What you can do on Android devices
- What you cannot do on Android
- A look at the Application Security Model

Not long after my wife and I got married, we moved 500 miles away to a new city. We were still the same couple as before, but we had to get used to our new environment — living in a new apartment, working in a new metro area, and finding new friends.

Developing Flash/ActionScript (AS3) apps for Android and iOS devices is quite similar. You already know the tool and the language that you've worked with for web and desktop-based Adobe Integrated Runtime (AIR) environments. Yet, you find yourself in a completely different runtime environment, with different capabilities and constraints that you never have to consider when working with desktop computers.

This chapter introduces these new two mobile environments and highlights some of the things you need to consider as you get started developing Flash-based applications for Android and iOS devices.

## EXPANDING TO THE MOBILE WORLD

Ever since its early days at Macromedia in the 1990s, Flash has been synonymous with interactive media, animations, and games that run embedded inside a Web page. And it has been Flash's ability and power to provide what HTML and JavaScript alone could not that has awarded the Flash plug-in a 99 percent installation rate among all Internet users.

Fast forward several years. Although Flash is still utilized predominately for browser-based purposes, the overall Flash landscape is becoming more diversified. Flash isn't just for interactive media and light apps; you can use it to deploy full-fledged mission-critical applications. In addition to Flash, its ActionScript "brother" Flex offers a more traditional application development environment that utilizes both AS3 and Flash run time.

Flash is no longer constrained to a browser window. With the release of AIR in 2007, Flash and Flex developers could, for the first time, create standalone, cross-platform, rich Internet applications (RIAs) for Windows, Mac OS X, and Linux platforms. These AIR desktop applications not only had the look and feel of native apps but could take advantage of native operating system capabilities, such as local file access, native menus and UI elements, and OS-specific events.

Although Flash's dominance on the desktop is unquestioned, its entry into the rapidly emerging mobile phone world has been far more problematic. Apple's refusal to support the Flash plug-in in the iPhone in its Mobile Safari browser left Flash Web developers out in the cold. In response, Adobe engineers came up with a different plan to get Flash-created content and applications onto iOS devices (iPhone, iPad, iPod touch): bypass the browser and go native. In other words, Adobe engineers figured out a way to package Flash apps as native iPhone apps — yes, the same apps that you can download and install from the App Store (see Figure 1-1). Adobe made Packager for iPhone available in Flash Professional CS5.

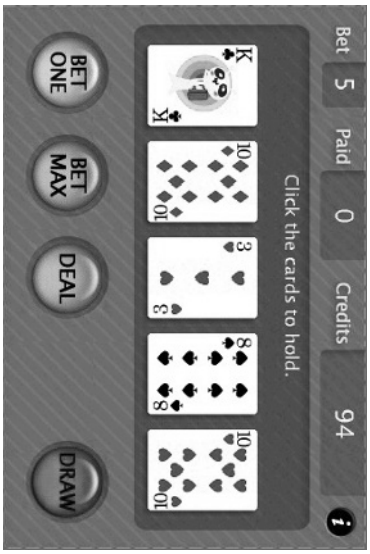


**FIGURE 1-1**

Beyond the iOS universe, Adobe also is expanding support for Flash onto other mobile platforms, particularly Android and BlackBerry. But, unlike the roadblocks that Adobe encountered with Apple, Adobe has been providing support for both Flash Player and AIR on these other mobile devices (as shown in Table 1-1). However, a strategic goal for Adobe has been to ensure that you will be able to take the same Flash project that you use for deploying on the iPhone and outputting it as an AIR app on Android or BlackBerry.

**TABLE 1-1:** Flash Platform Support

| PLATFORM | BROWSER | NATIVE |
|---|---|---|
| Windows | Flash Player | AIR |
| Mac OS X | Flash Player | AIR |
| Linux | Flash Player | AIR |
| iOS | None | Packager for iPhone |
| Android | Flash Player | AIR |
| BlackBerry | Flash Player | AIR |

# DISCOVERING ADOBE AIR

Before you begin to tackle Flash-based mobile development, it's important to have some basic understanding of the runtime environment on which you will be developing applications.

## Building for Android

Before you begin to develop Flash-based mobile apps, I wanted to "peek under the hood" for a moment and explain to you just exactly how Adobe can take a Flash file (`.fla`) and publish it as an `.apk` for Android or `.ipa` for iOS.

For Android apps, the process is not much different than AIR apps for the desktop. The AIR for Android run time provides an environment on which developers can build applications using Flash technologies and deliver it as a standalone application, outside of any browser. Users need to install the AIR for Android run time on their Android devices, and then Flash-based Android apps run on top of it.

AIR for Android embeds the cross-platform virtual machine Flash Player used to run media and apps created using Adobe Flash or Flash Builder. Inside of an AIR app, you have programmatic access to existing Flash Player API calls as well as some enhanced functionality for vector-based drawing, multimedia support, and a full networking stack.

AIR for Android also embeds SQLite, a database engine for enabling local database access. It is an extremely lightweight, open source, cross-platform SQL database engine that is embedded in many desktop and mobile products. Unlike most SQL databases, it does not require a separate server process and uses a standard file to store an entire database (tables, indexes, and so on). For more information on SQLite, go to `www.sqlite.org`.

When you publish a Flash file for Android, your `.fla`, .as source code, and other source files are transformed by the ActionScript compiler into a binary format called ActionScript Byte Code (ABC). The ABC is packaged inside a `.swf` file (see Figure 1-2). The `.swf` and supporting resource files are then packaged together as an Android package (`.apk`) ready for installation onto a device.

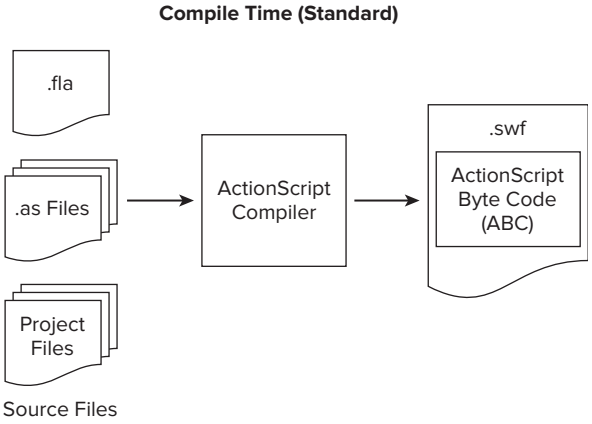


**FIGURE 1-2**

At run time, the `.swf` file inside of the Android app is processed by the ActionScript Virtual Machine (AVM2), which is part of the AIR for Android run time. The AVM2 loads the ABC file into memory and decodes it. The bytecodes are then run through an interpreter and executed as native machine code (see Figure 1-3) by the AIR run time. This process of bytecode compilation by the AVM2 is specific to the Android platform.

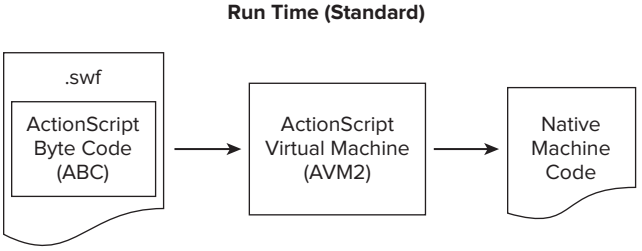


**FIGURE 1-3**

## Building for iOS

So, while Android apps run on top of an AIR runtime environment that is installed separately, iOS apps wrap the runtime code inside of the app itself, making it self-contained. Not surprisingly, then, the process in which an iOS app is created, is quite distinct. Let me explain.

When you compile a Flash project for iPhone, the ABC code is compiled by the Low Level Machine Compiler (LLVM), which is an open source compiler infrastructure that is used to generate machine code for iOS. (Apple itself uses the LLVM). However, as shown in Figure 1-4, in its Packager for iPhone, Adobe provides an ActionScript front-end to the LLVM for handling Flash files.

While the AVM2 supports JIT for Web and AIR, LLVM uses Ahead-Of-Time (AOT) compilation to produce native ARM assembly code wrapped inside of an iPhone executable file. The `.ipa` also contains a `.swf` containing assets and a configuration file.

During the publishing process, the AIR Developer Tool (ADT) is used to add the `.p12` certificate to the `.ipa` application file for developer authentication. The resulting `.ipa` is a native iPhone application ready for installation on your iPhone device.

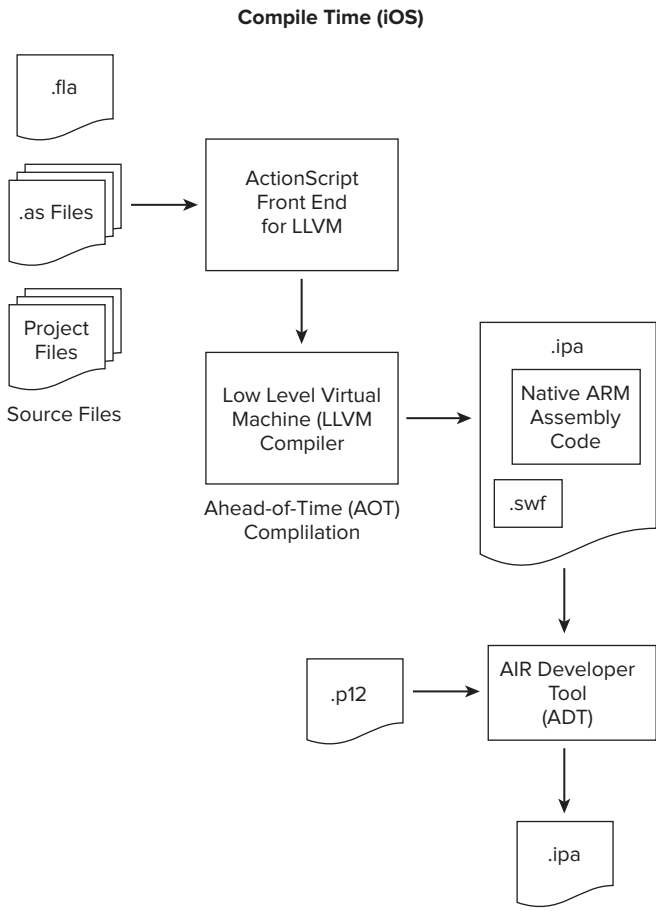


**FIGURE 1-4**

## WHAT YOU CAN AND CANNOT DO

Flash CS5 and Flash Builder allow you to create native Android and iOS apps, but it is important to understand from the get-go the capabilities and limitations of the types of functionality you develop.

### Device Support

Support for Android and iOS APIs are similar, but at the time of writing, not identical. Table 1-2 summarizes the API capabilities of Flash apps running on Android and iOS.

**TABLE 1-2:** API Support

| API | ANDROID | IOS |
|---|---|---|
| Touch/Gestures | X | X |
| Accelerometer | X | X |
| Geolocation Sensor | X | X |
| Camera | X | |
| Microphone | X | |
| StageWebView | X | |
| CameraRoll/Photo Library | X | Add to library only |
| File I/O | X | X |
| SQLite database | X | X |
| Ability to launch mobile services via URL protocol, such as Phone (`tel:`) and E-mail (`mailto:`) and SMS (`sms:`) | X | X |

As you can see, some core mobile services are unsupported:

- Native UI controls
- Music player and library
- Bluetooth
- Contacts
- Calendar
- Preferences

The areas of strongest support center on multitouch and gesture events, Accelerometer, and persistent file and database storage. Both AIR for Android and Packager for iPhone are weaker than their native SDK counterparts in being able to integrate with other parts of the respective devices — both hardware and system services. Therefore, as you architect your apps, factor in those constraints.

## Unsupported AS3 API Objects

When creating mobile applications, you have access to many parts of the core AS3 library and AIR API extensions. However, not all core and AIR functionality is supported on Android and iOS. The following is a list of AS3 API objects or members that are *not* supported:

- `Accessibility`
- `DNSResolver`

- `DockIcon`
- `DRMManager`
- `EncryptedLocalStore`
- `HTMLLoader`
- `LocalConnection`
- `NativeApplication exit()`, `isSetAsDefaultApplication()`, `menu`, and `startAtLogin`
- `NativeMenu`
- `NativeProcess`
- `NativeWindow` and `NativeWindow.notifyUser()`
- `NetworkInfo`
- PDF support
- `PrintJob`
- Socket support (`DatagramSocket`, `SecureSocket`, `ServerSocket`)
- `Shader`
- `ShaderFilter`
- `Socket.bind()`
- `StorageVolumeInfo`
- `XMLSignatureValidator`

What's more, although you can use Flash Builder to create AIR for Android apps, the Flex MXML framework is not officially supported on Android and incompatible with iOS.

## UNDERSTANDING THE AIR FOR ANDROID SECURITY MODEL

AIR for Android carries over the same basic security model that Adobe created for the desktop version of AIR. In the traditional desktop environment, desktop apps get permission in terms of what they can do and cannot do from the OS and the available permissions of the currently logged in user. They receive this level of access because the users need to explicitly install the app — effectively telling their computer that they trust the app they are about to launch. As a result, native apps have access to read and write to the local file system and perform other typical desktop functions.

Web apps, however, are far more restrictive because of the potentially malicious nature of scripting. As a result, web apps limit all local file access, can only perform web-based actions inside the context of a browser, and restrict data access to a single domain.

The hybrid nature of an AIR for Android application puts it somewhere between the traditional desktop and restrictive web security models. On the one hand, you can create an Android application

that runs on top of the normal Android OS security layer. Therefore, it is able to read and write from the local file system. However, because AIR utilizes web technologies that, if unchecked, could be hijacked by a malicious third party and used in harmful ways when accessing the local system, AIR has a security model to guard against that happening. Specifically, AIR for Android grants permissions to each source or data file in an AIR application based on their origin and places them into one of two kinds of sandboxes.

The application sandbox contains all content that is installed with the app inside of the home directory of an application. Only these resources have access to the AIR for Android API and its runtime environment.

Adobe AIR does allow you to link in other local and remote content not inside the root directory of the application, but it places this content in a non-application sandbox. Content inside the non-application sandbox is essentially handled from a security standpoint just like a traditional web app and is not granted access to the AIR APIs (see Figure 1-5).

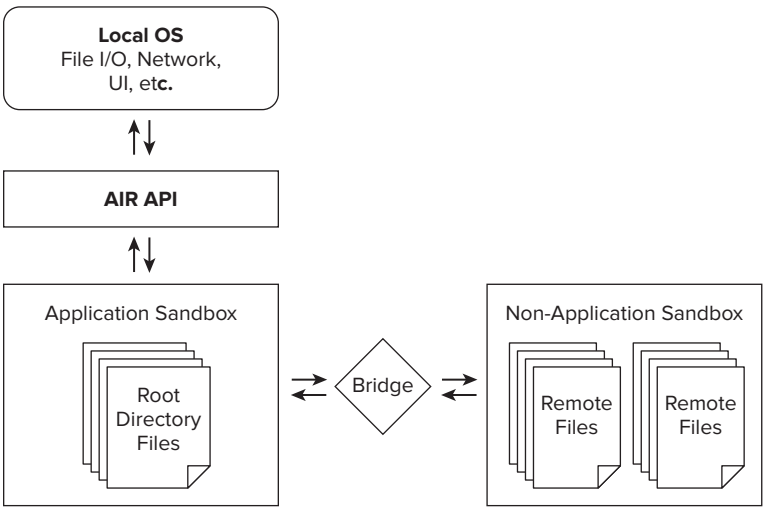


**FIGURE 1-5**

## GETTING TO KNOW THE ANDROID SDK

Apart from using Flash CS5, Flash Builder, or the AIR command-line utilities, the only way to create Android applications is by working with the Android Software Developer Kit (SDK). The SDK is a set of APIs and development tools that developers use to create native Android apps. Although much of the Android SDK is not directly useful to Flash developers, you will still utilize some of its tools during your app development process. Therefore, you'll want to begin by downloading and installing the latest version at `http://developer.android.com` before continuing.

## GETTING TO KNOW THE IOS SDK

Apart from Flash CS5, the only way to create iOS applications is by working with Apple's iOS Software Developer Kit (SDK). The SDK is a set of APIs and development tools that are used by Objective-C developers to create native iOS apps. While much of the iOS SDK is not useful to Flash developers, you can still utilize some of its profiling and diagnostic tools to debug your apps. Additionally, it is also a good idea to know what's in the SDK, particularly as you read Apple reference materials pertaining to iOS app development.

The core API frameworks include:

- Cocoa Touch framework is the core API used for developing iPhone apps. It includes support for multi-touch and gestures, accelerometer, and camera.
- The Media API provides support for video, audio, and core animation processes.
- Core Services are lower level services that provide networking, database, and thread support.
- OS X Kernel is the lowest level of services providing basic File I/O, TCP/IP, security, and power management.

iOS SDK apps are built using Xcode developer tools. Xcode consists of the following:

- Xcode IDE is the Objective-C based development environment.
- Interface Builder is used to create user interfaces in a visual environment and then link them into the Xcode project.
- Instruments is a diagnostic tool that collects disk, memory, and CPU data of an app in real time.
- Shark is a companion tool for profiling your app.

As Chapter 13 explains, you can use Shark and Instruments with your Flash-built apps.

## SUMMARY

In this chapter, you were introduced to Android and iOS application development using Flash CS5 and Flash Builder. You reviewed the API support for both mobile operating systems and explored what you can and cannot do in your Flash app. After that, you read aspects of the general AIR API that are available to you as you develop for Android devices. Finally, you took a quick survey of the Android SDK and iOS SDK. Although you do not need to use many parts of it for Flash apps, it is helpful to know that it contains support tools that can make your life easier.