Introducing Android Devices

It is an exciting time to be a Flash developer. Adobe has taken big steps in making the Flash platform available on as many devices as possible. The Open Screen Project is an Adobe-led initiative whose goal is to "enable consumers to engage with rich Internet experiences seamlessly across any device, anywhere," as it says on its Web site, at www.openscreenproject.org.

Flash Player 10.1 will be available for multiple mobile platforms, such as Google Android, RIM's BlackBerry, Palm Pre, and Nokia, as well as numerous other devices such as TVs, set top boxes, tablets, and netbooks. Adobe is working with these and over 50 other partners to optimize Flash Player 10.1 in order to work better with the different devices.

Flash Player 10.1

There are two very different parts to the Flash platform on mobile devices. Flash Player 10.1 has been optimized for use in mobile browsers. The intent is to have all the content that you would normally see on a desktop browser work just as you would expect on a mobile device. Flash Player 10.1 also takes advantage of some of the new mobile device APIs, such as multitouch and geolocation. If you currently have Flash content on the Web, it is a good idea to test it on an Android device with Flash Player 10.1 installed. Consider updating the content to better support mobile devices if possible. As mobile devices become more popular, users will demand that content work seamlessly across all platforms. The Flash Player 10.1 plug-in is available now for supported Android 2.2 devices in the Android Market.

AIR for Mobile Devices

The other piece of the Flash platform mobile story is AIR for mobile devices. AIR (Adobe Integrated Runtime) was originally designed and developed for the desktop, and some features do not translate well to mobile platforms. To help with this, Adobe has created the mobile profile for AIR, which is a subset set of AIR. To find out which APIs are available, see the sections "Introducing the Available APIs" and "Check What APIs Are Not Available" later in this chapter.

Currently, the AIR mobile profile is available for iPhone OS and Android devices. However, because of the new terms of use by Apple, you can no longer use Flash CS5 to create and submit applications to the Apple iTunes App Store. The AIR Runtime is available on Android devices and needs to be installed on the device in order to run any AIR applications. Similarly to the desktop runtime, the mobile runtime will be installed if the user tries to install an AIR for Android application without the runtime.

The AIR Runtime is currently supported on Android devices with Android 2.2, also known as *Froyo,* installed or higher. It also is currently supported only on Android devices with an ARMv7 processor. Most new devices should support the AIR Runtime; however, you will want to double check before you purchase a device.

Because the Android platform is open source, many device manufacturers ship their devices with custom Android skins. As Google releases new versions of Android, each manufacturer must convert its skins to the new version before the device can be updated. Because of this, it is a good idea to stay away from devices that have highly customized skins, such as the Sony Ericsson Xperia X10, which is capable of running AIR except that it currently ships with Android 1.6.

Android AIR Devices

The following are a few of the devices that are currently available that support the AIR Runtime and Android AIR applications. You may still need to update them to Android 2.2, but after they are updated, they should run all applications.

Google Nexus One

The Nexus One is manufactured by HTC for Google and can be purchased from Google unlocked. Being able to purchase an unlocked device is attractive for developers, as they are not restricted to a specific network provider. This also means that you can purchase one if you plan to use it only as a development device and not as an everyday mobile phone. There are two versions available that support the different cell network providers and can be purchased for the United States, Canada, and Europe.

The Nexus One ships with Android 2.1, a 1GHz Snapdragon processor, and 512MB of memory. It also has 512MB of Flash memory storage and a 4GB microSD card. The SD card can be upgraded to a 32GB card. The Nexus One has a screen resolution of 480 x 800 pixels and has a 5.0 megapixel camera, which can shoot 720 x 480 video at 20fps or higher.

Motorola Droid

Outside of the United States, the Motorola Droid is referred to as the Motorola *Milestone*. Both devices currently support Android 2.1. There are two big differences between the Droid and the Nexus One: The Droid has a slide-out keyboard and a screen resolution of 480 x 854 pixels.

HTC Desire

The HTC Desire is one of the more popular Android phones on the market today, and it is very similar to the Nexus One. The biggest difference between the two devices is the Desire's HTC Sense Android user interface. The Desire also has tactile buttons across the bottom as opposed to the Nexus One's touch-sensitive buttons.

Android Tablets

There are a number of Android tablet devices that are set to release in the last half of 2010, and there are too many to discuss here. However, the most important thing to understand is the difference in screen resolution. 1024 x 768, 1366 x 768, 800 x 600, and 1024 x 600 are just some of the different screen resolutions for tablets that will support AIR Android applications. Testing how your application will look and respond on these different screen resolutions will ensure that users have a good user experience across devices.

Test for Multiple Devices

As you can see, there are many differences between all the different models, and the tablets are a game changer. There are lots of tips and things to think about in this book when developing your applications to support multiple platforms. It is a good exercise to try and take all of these into consideration early on in development. Some things to ask yourself are, "What does my application look like in multiple resolutions?" and "How does a user interact with my application on a non–touch-enabled device?." Even if you ever plan to support only one device today, allowing for multiple platforms in the future will prove to be worthwhile. The iPad is a great example of developers never planning for a different resolution or platform. When the iPhone SDK first came out, developers were fortunate and had to design for only one screen size and resolution. Now with the release of the iPad and the fourth generation iPhone, developers have to redesign and even rewrite their applications in order to support these new platforms.

Introducing the Development Tools

If you have developed Flash applications before, you will already be familiar with some of the tools that will be explored throughout this book. However, some of them may be new to you. The following are all the different tools that are discussed throughout the book.

Flash CS5 Professional

Flash CS5 is the main integrated development environment (IDE) for developing Flash applications for the Web, desktop, and Flash Lite–enabled mobile devices. In this 11th version of Flash, Flash CS5 introduced us to the ability to publish Flash applications to native iPhone applications, which ships with the product. In the fall of 2010, Adobe released an update for Flash CS5 and AIR, which enables you to develop and publish applications for Android devices. The update can be downloaded through the Adobe CS5 Updater application or from the Adobe Web site. If you are not sure if you have the latest version, select Updates from the Help menu in Flash CS5.

Flash Builder

Flash Builder, formerly know as *Flex Builder*, is an Eclipse-based IDE for creating Flex and AS3 projects. Flash Builder is Adobe's main ActionScript coding application. Flash CS5 does have the capability to write separate ActionScript code and classes; however, Flash Builder provides a much more feature-rich development environment. With the newest version of Flash Builder, Adobe has also integrated a better workflow between it and Flash CS5. You are now able to publish .fla files directly from Flash Builder without having to switch between applications. One of the benefits of Flash Builder is that it is built on top of Eclipse, a popular open source IDE. This enables you to take advantage of the many plug-ins built for Eclipse, which provide additional functionality that you do not get in Flash CS5. There are many plug-ins for managing source control, build integration, and support for other programming languages.

FDT

FDT (Flash Development Tool) is an Eclipse-based IDE similar to Flash Builder. FDT provides many features that speed up development, such as code templates, quick fixes, quick assist, and organized imports — just to name a few. FDT 4 is currently in beta and is quickly becoming one of the most popular ActionScript editors among many of the best Flash developers in the community. FDT 4 gives developers the ability to create their own plug-ins for FDT. This allows community members and developers to create a workflow that is best suited for them. It is only a matter of time before a FDT Android plug-in is released to help with publishing and installing Android applications.

Android SDK

The Android SDK is developed and released by Google, and it enables developers to create native Android applications. There are several tools that come with the Android SDK that are extremely helpful in developing Android applications. Before you begin to develop your application, make sure to download this SDK from the Google Web site. For more details, see the section "Get the Android SDK" later in this chapter.

Android Emulator

The Android emulator comes included in the Android SDK and can be used to test your AIR Android applications on your development computer. The emulator can be configured in a number of different ways, which is extremely useful for testing your application on multiple types of devices. You can also simulate a number of device-specific actions that would normally be available only on a device, such as a voice call, SMS message, and geolocation events.

AIR Runtime for Android

In order for your AIR Android applications to run on an Android device, you must first have the AIR Runtime installed. If an AIR Android application attempts to install without the runtime, it will prompt the user to download the runtime first. This process is very similar to the process of installing an AIR application on the desktop. Having the runtime installed is true for both Android devices and any Android emulators you have created. The AIR Runtime is installed the same way as any other Android application.

Android adb Tool

The Android Debug Bridge (`adb`) is a tool that comes with the Android SDK. It enables you to manage the state of an Android emulator instance or a connected device. It can be found in the tools folder of the Android SDK and is used by the Flash IDE to install applications onto your connected devices. If you are comfortable with using the command line, you can use `adb` to drop into a remote shell on an emulator or device instance, and you can issue shell commands on these instances. You can also use the `adb` tool to push or pull files from your device. This can be extremely useful if you want to better examine a file that your application has created on the device. For more details and other `adb` commands, check out the `adb` page on the Android Developers site, http://developer.android.com/guide/developing/tools/adb.html.

Adobe ADT Tool

The AIR Developer Tool (ADT) is used to compile your Android applications from your Flash and AIR applications. The ADT tool can be found in the bin directory of the AIR SDK, as well as the AIK2.5/bin folder in the Adobe Flash CS5 directory. ADT can be used by the command line to package your applications. For more details on compiling Android applications with the command line, see Chapter 3, “Developing Your First Application.” You can also create a self-signed certificate, which can be used to digitally sign your AIR Android applications. It is important to note that applications uploaded to the Android Market must be signed with a certificate that have a validity period ending after October 22, 2033.

Android Market

The Android Market is a place where you can download and install applications to your Android device. It is similar to the iTunes App Store for the iPhone, except it does not have a desktop version and is available only on your device. If you want to submit your application to the Android Market, you need to register with Google for an Android developer account, which costs $25 US. After registration, you will be able to submit applications through the Android Market Web site. For more details on preparing and submitting your applications, see Chapter 17, “Deploying Your Application.” Both the Flash Player 10.1 plug-in for browsers and the AIR Runtime can be downloaded from the Android Market, and you should install them on your device if they are not already.

Introducing the Available APIs

With the ability to publish Android applications from Flash CS5 comes a set of new APIs that enable you to take advantage of some of the features the Android platform has to offer. However, Adobe's strategy is not to support only the Android platform but as many platforms as possible. This is the reason you may not see as many Android-specific features as you may like or think. Adobe is being very pragmatic about what new features it introduces and how its APIs will look on future platforms, mobile or otherwise. Adobe's goal is to provide one consistent API for all platforms. For example, the ActionScript code should be the same for accessing a camera whether you are developing applications for the Web, desktop, Android, iPhone, or any other future supported platform.

Accelerometer

The new `Accelerometer` class, which can be found in the `flash.sensors` package, gives you the ability to interact with the accelerometer that is built into the device. The Android accelerometer is a three-axis accelerometer capable of measuring both acceleration and gravity. The accelerometer is used to detect the device's rotation as well as any movements such as shakes.

Camera Roll

The `flash.media.CameraRoll` class enables you to save a `BitmapData` instance to the device's camera roll. You can also use the `CameraRoll` class to enable the user to select an image from the Gallery application on the device.

Touch Event

The `flash.events.TouchEvent` class is used to detect when a user touches the screen with his or her finger. The `TouchEvent` class is very much the alternative to the `MouseEvent` class but for touches. As well as being available on Android devices, the `TouchEvent` class is also available on AIR applications in Windows 7 with a touch-enabled screen and Flash CS5 iPhone applications.

Geolocation

The `flash.sensors.Geolocation` class enables you to interact with the device's location sensor. With this class, you can retrieve the location of your device anywhere in the world. Coordinates are reported to you in the form of latitude and longitude. There are differences in the ways each device figures out your location. It is important to understand that every device that has the Android OS installed does not necessarily have the same location sensor and that accuracy will differ greatly between the devices.

Stage Orientation

There are a few new classes and methods to help handle stage orientation changes and updates. The `flash.display.StageOrientation` class defines a set of valid orientations in which the Stage can be set. The `flash.display.Stage.setOrienation` method, which is new in AIR 2.0, enables you to set the orientation of the Stage based on one of the static properties in the `StageOrientation` class. And finally, there is a `flash.events.StageOrienationEvent` class, which allows you to listen for when the Stage orientation is changing and has changed.

Gesture Transform Event

The `flash.events.GestureTransformEvent` class is used to detect specific user interactions with multiple fingers. Touch-enabled interfaces are still very much in their infancy; however, there is already a standard set of gestures that a user understands and expects when interacting with your applications. The `GestureTransformEvent` class can detect four different types of gestures: swipe, rotate, pinch and zoom, and pan. Swipe detects a single finger swiping across the screen in the left, right, up, or down direction. The rotate gesture allows you to place two fingers on an object and rotate one finger around the other to rotate the object. The pinch and zoom gesture enables you to zoom in and out of objects by moving your fingers closer or farther apart from each other. Finally, the pan gesture lets you pan an object in any direction with two fingers. Some of these gestures are supported in Windows 7, Mac OS X 10.5.3, and Windows Mobile 6.5, as well as on iPhone and Android devices. If you plan on supporting more than one platform, you should double check which gestures are fully supported on each before starting development.

NetConnection

Adobe has added `NetConnection` support for Android applications. The `NetConnection` class gives you the ability to connect to a Flash Media Server to create peer-to-peer applications, as well as view streaming video on your device. It will also allow you to communicate with Flash Remoting and the AMF protocol.

FLV

FLV is Adobe's Flash Video format. There are two methods of playing an FLV video on a Android device. The first is to import the video onto the Timeline in an FLA file. This will create a `MovieClip` with your video in it, and you will be able to control it just as you would a normal `MovieClip`. The second method is to bundle the file with the application and load it at runtime. Both of these methods are covered in Chapter 8, "Working with Video," later in the book.

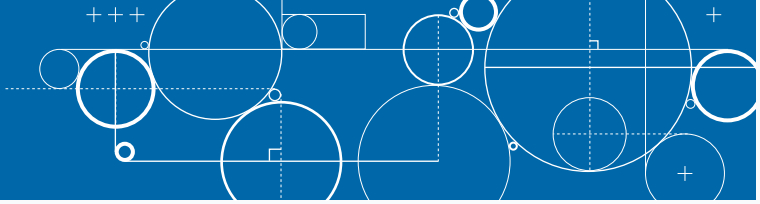
Shared Objects

`SharedObject`s are Flash's version of a browser cookie. They allow your Flash application to save user data on the device. This allows your application to load the save data when you come back to the application.

SQLite Database

The SQLite database is the most widely deployed SQL database in the world. When Adobe released AIR 1.0, it included the ability to communicate with SQLite databases from your applications. This gives you the ability to save large and complex data locally on the device. The Android platform also has SQLite libraries to develop with. A lot of the data on your device is stored in SQLite databases, such as your Contacts and Call History. Adobe has given us the ability to create, save, and load data from a SQLite database.

Check What APIs Are Not Available



Some of the AIR features that you are familiar with on the desktop are not currently available in the mobile profile of AIR. The omission of some of the features makes perfectly good sense, such as the concept of windows, as an Android application can have only one window, but some may not be so obvious. It is still the early days for the AIR mobile profile, so Adobe has decided to release the AIR Runtime without some of the features instead of having us wait longer for more features. Also, the AIR mobile profile started out with creating native iPhone applications. Apple placed certain restrictions on applications, which prevented Adobe from developing these features from the start. Adobe is committed to making the AIR Android Runtime a success and is constantly working to include new features and updates for developers to take advantage of.

HTML Loader

The `HTMLLoader` class provides AIR with the capability to render Web pages inside the application. Adobe has bundled a version of WebKit inside the AIR Runtime for the desktop, which provides a number of advanced features that are most likely not needed for mobile devices. In order to provide support for `HTMLLoader`, Adobe would have to bundle WebKit with the Android AIR Runtime, which would increase both its file size and the memory footprint. You can use the `StageWebView` class in order to load and display HTML text and Web pages within your application. However, the `StageWebView` class does not provide developers with the same level of communication between ActionScript and the JavaScript as `HTMLLoader` provides.

New AIR 2.0 Networking Classes

With the most recent release of AIR 2.0, Adobe has introduced some new networking classes. The `ServerSocket` class enables you to create your own socket server and have other AIR clients connect to it via the `Socket` class. This feature allows you to pass data to and from applications. The `ServerSocket` class is currently not available, but you can create a `Socket` in your Android application and connect to one running on your computer.

The `DatagramSocket` class enables you to send and receive UDP data. This can be used to create peer-to-peer applications or gaming. It is more unreliable than a TCP socket because you cannot guarantee the order of the data you will receive and lost packets are not retransmitted or even detected.

Adobe has also introduced a set of classes that can be used in conjunction with the new socket and server classes. For example, the `NetworkInfo` class provides you with a list of all available network interfaces available on the current machine. From those, you are able to find out what type of interface they are as well as the IP for the address. This comes in handy when you are creating peer-to-peer applications in which you may not know the IP address of the other peer.

There is technically no reason why these classes are not part of the Flash to Android offering, and they have been a highly requested feature. Adobe does plan on implementing these sometime in the future — when, however, is very hard to say.

Pixel Bender

Currently, Pixel Bender kernels and the `Shader` class are not supported in the Android AIR Runtime. These were originally not supported because the Apple App Store terms of use state that an application cannot execute any interpreted code. This may be something that Adobe visits with Android, but it is most likely lower on the priority list for Adobe and developers.

AS1 and AS2

This may come as a surprise to some developers, but there are still many developers who have not adopted AS3 and are still using AS2 or AS1. If you are one of these developers or have some older projects that you are looking to convert, you will need to start learning AS3 and start converting your projects. Currently, the AIR packager works only with AS3, and there will not be support for any earlier version of ActionScript. All the tasks and code in this book are in ActionScript 3, so if you are not familiar with it, I suggest you read up on ActionScript in order to get up to speed first; I recommend the book *ActionScript: Your visual blueprint for creating interactive projects in Flash CS4 Professional,* available from Wiley Publishing.

Alchemy

Adobe Alchemy is an Adobe Labs project, which enables developers to compile C and C++ code that would run in the ActionScript Virtual Machine (AVM). The exciting thing about this project is that it allows you to add functionality to the Flash Player that currently does not exist. Some community examples include being able to play external .wav audio files and open and preview .psd Photoshop files. Currently, this is not officially supported, but if there is some Alchemy library that you want to use, test it on your device to see if it does work.

Notifications

Android devices have a notifications area in the status bar, which can be used by applications to notify the user of an event. This is commonly used for new email notifications as well as application updates. Currently, only native Android applications can post notifications to the notification area, not AIR Android applications.

Widgets

You can develop many types of applications for the Android platform. As well as a regular application that a user would launch from the home screen or Android menu, the Android SDK allows you to develop widget applications and services. *Widget applications* are no more than a miniature application view that can be placed on the home screen. Widget applications provide users with small amounts of data or statuses, such as the weather or news. Currently, you cannot build widget applications with the Android AIR Runtime.

Become an Android Developer

If you want to submit your AIR Android applications to the Android Market, you must first create an Android developer account. The registration fee for the Android Market is $25.00 US and can be paid using Google Checkout during the registration process. This is a one-time fee and does not require a renewal every year like other mobile development platforms.

In order to sign up for an Android Market account, you must first log in to your existing Google account. If you do not have an account, you can click a link on the sign-up page to create one. After you are signed in with your Google account, you will be presented with the Getting Started page. This page has a form on it that will be used to create your Android Market account. The information will also determine how you appear to customers in the Market.

The developer name that you choose will appear underneath your application listing in the Android Market. Feel free to enter your name if you are an individual developer or your company name if you plan on selling your applications. Your developer name will also be a link that sends users to the URL provided in the Website URL field.

After you have filled out all the information correctly and submitted it, you will be brought through the payment process of your application. Google uses its Google Checkout service to process your payment. After your payment has been accepted, your application will be approved, and you will be brought to your developer home screen. From here, you can upload your applications to the Android Market. For more details on submitting your applications, see Chapter 17.

Become an Android Developer

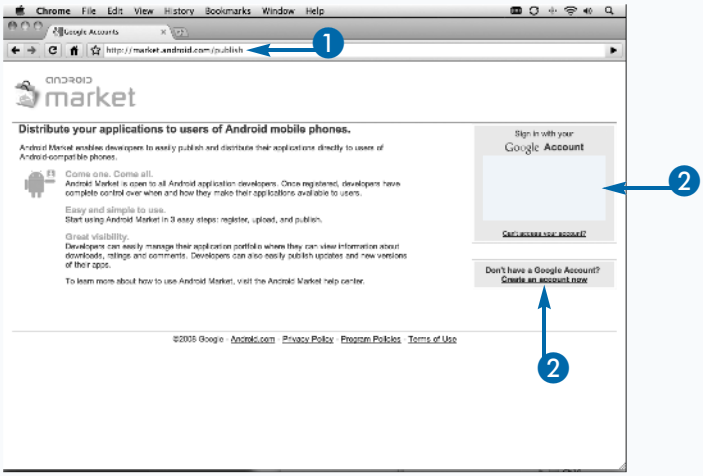
Start the Registering Process

1. In a browser, go to http://market.android.com/publish.
2. Sign in with your Google account.

   OR

2. Click Create an Account Now.

Note: *If you already have a Google account, skip to step* ***10****.*

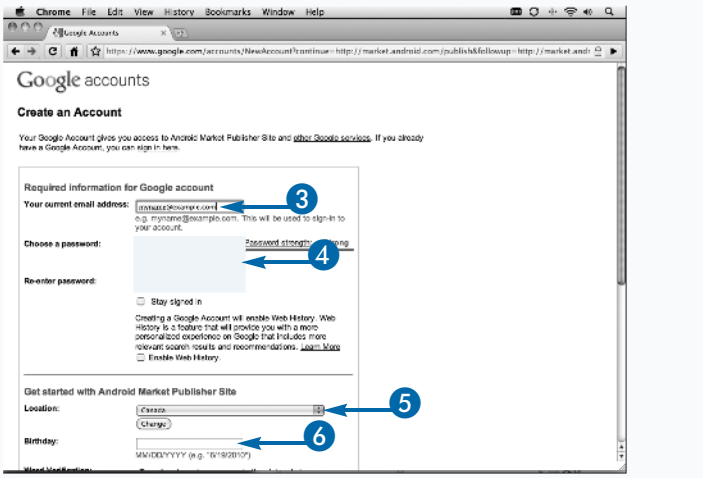


Create a Google Account

3. Type your current email address.
4. Type and confirm a password.
5. Click here and choose a location.
6. Type your birthday.
7. Scroll down and type the word shown in the image.
8. Click I Accept. Create My Account.

   The Account Create Confirmation screen appears.

9. Click Click Here to Continue.



Create an Android Market Account

The Getting Started page appears.

10. Type a developer name.
11. Double check that your prefilled-in email address is correct.
12. Type the URL of your Web site.
13. Type your phone number.
14. Click Continue.

    The Registration Fee page appears.

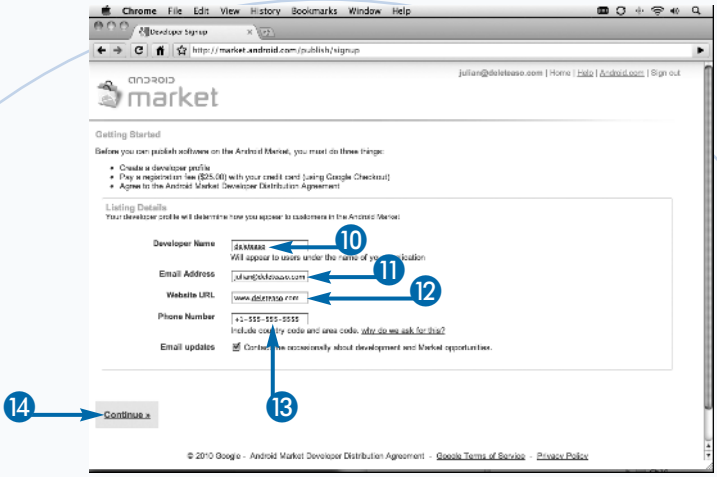
15. Click Continue.

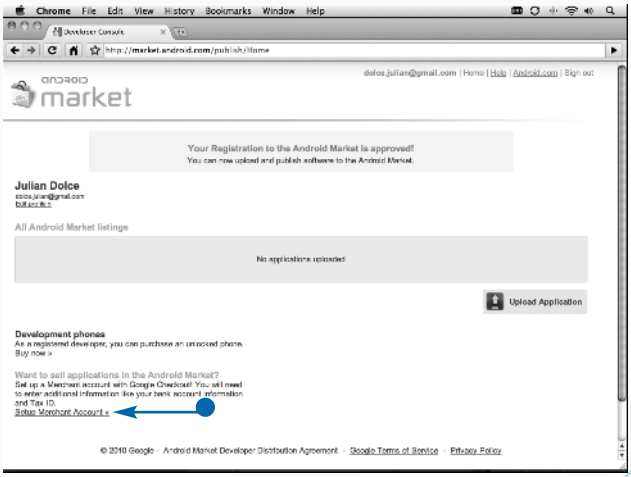
    The Google Checkout page appears.

16. Fill in all the necessary information and pay the $25.00 registration fee.

    Your Android Market developer page appears.

- You can click Setup Merchant Account and follow the site's instructions if you plan on selling your applications.





Extra

If you plan on selling your applications, you will need to set up a merchant account with Google Checkout. To set up your account, you will need to enter additional banking information as well as tax ID numbers. Make sure to have these ready before you start. If you plan on uploading only free applications, you can skip this step. To set up your merchant account, click the Setup Merchant Account link at the bottom of your developer home page. You will be presented with a form that you will need to completely fill out in order to create your account. The processing fee for the Android Market purchases is different from a regular Google Checkout purchase. When an application is sold in the Android Market, Google will keep 30% of the application price. For example, if your application is sold for $10.00, you will receive a payment of $7.00, and Google will keep $3.00 as a transaction fee. If you do not have all your business, banking, and tax information ready during the sign-up process, you can enter it any time to set up your merchant account. However, you will not be able to charge customers for your applications until you do.

Get the Android SDK

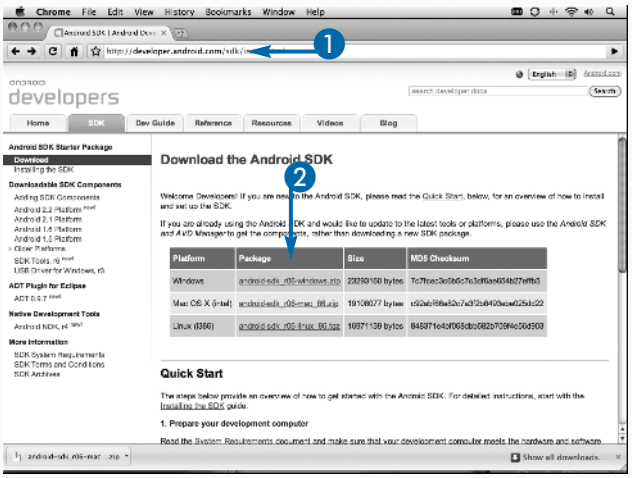
The Android SDK is not required to develop AIR Android applications; however, it is required to interact with your device in order to install your application. The Flash CS5 IDE uses the SDK to install the compiled Android application on your device when you publish. The SDK also includes an emulator that allows you to test your applications locally before testing on a device. It is highly recommend that you download the SDK before you begin developing your application.

The SDK can be downloaded from the Android Developers site. There is a separate download for Windows, Mac OS X, and Linux; simply choose the version for the operating system on which you will be developing. After the .zip file has finished downloading, extract it and place the extracted contents in a location where you can access it conveniently.

In the root of the SDK, there is a tools folder. This folder contains several of the tools or executables that you will use to interact with your device and the emulator. The android executable provides a user interface for managing the SDK. Running this application will bring up the Android SDK and AVD Manager window. From here, you can download all the up-to-date components of the SDK. Be sure to download all of the Android 2.2 API 8 packages. There is no need to download the documentation or the samples, but as long as you download the SDK platform, you should have everything you need in order to develop AIR Android applications.

Get the Android SDK

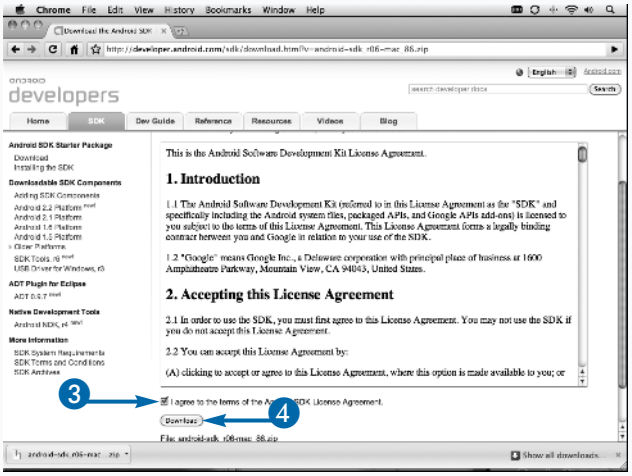
1. In a Web browser, go to http://developer.android.com/sdk/index.html.
2. Click the SDK package for your operating system.



The License Agreement page appears.

3. Click I Agree to the Terms of the Android SDK License Agreement.
4. Click Download.

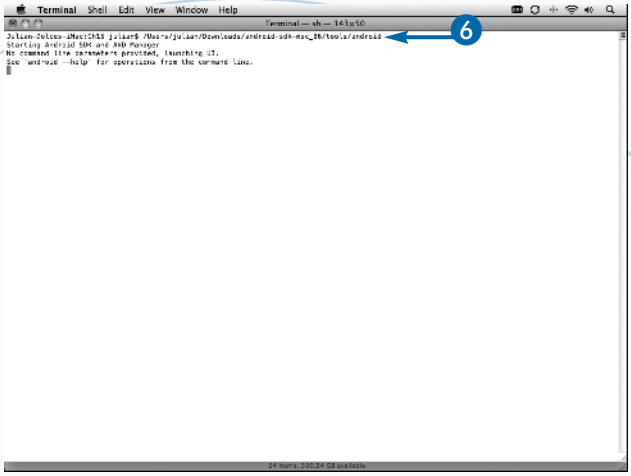
   The SDK is downloaded.
5. Extract the SDK .zip package.



6. On Mac OS X, in a Terminal window, run the android tool in the tools folder of the SDK.

   OR

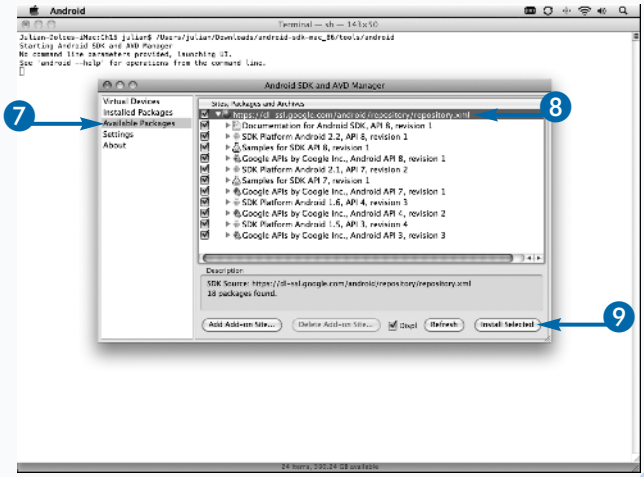
6. In Windows, execute the SDK Setup.exe at the root of the SDK folder.



   The Android SDK and AVD Manager interface appears.

7. Click Available Packages.
8. Click the packages to download and install.
9. Click Install Selected.

   The packages are downloaded and installed.



Extra

To make life easier when using the tools in the SDK through the command line, add the tools folder of the SDK to your System Path variable. This can be done on Mac OS X by creating a .bash_profile file in the root of your user directory and placing the following text in it, making sure to replace `<PATH TO SDK>` with the full path to the root folder of your SDK:

```
export PATH=${PATH}:<PATH TO SDK>/tools
```

This process for Windows is different for each version, and much more complicated. A Google search for "Adding folders to Path variable for Windows" will give you many results on how to do this.

Get the Android Eclipse Plug-in

The Android Eclipse Plug-in is primarily used for developing native Android applications with Java. However, it provides many valuable features that you can use to help debug your applications. This is especially useful if you are already using an Eclipse-based IDE, such as Flash Builder or FDT, as your primary ActionScript editor. Even if you are not currently using Eclipse, the Android plug-in can be useful if you are not comfortable with using the command line in order to interact with the SDK.

You can install the plug-in using the Install New Software option in Eclipse or your Eclipse-based IDE, such as FDT or Flash Builder. Have the URL https://dl-ssl.google.com/android/eclipse/ handy because you need to enter it in the Install dialog box. Eclipse searches this URL for any available Eclipse software and plug-ins. If the URL is entered correctly, the Development Tools item will appear in the dialog box. You can select this to begin the download and installation process.

After the plug-in is downloaded and installed, you will be asked to restart Eclipse. When you restart, you will be prompted to enter the path to your Android SDK location. If you have not downloaded the Android SDK, see the preceding section, "Get the Android SDK," for more details.

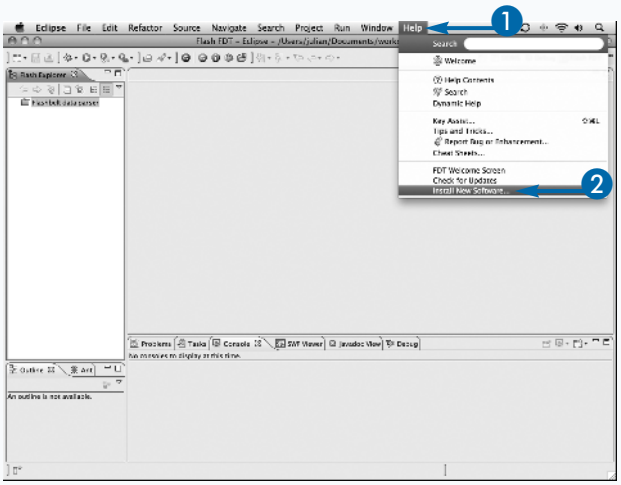
From Eclipse, you can access the Android SDK and AVD Manager window by selecting Window ➔ Android SDK and AVD Manager from the main menu. In that window, you can set up new emulator instances.

The Eclipse plug-in also installs the DDMS (Dalvik Debug Monitor Service) perspective, which provides a number of different ways to interact with your device and emulator instances.

Get the Android Eclipse Plug-in

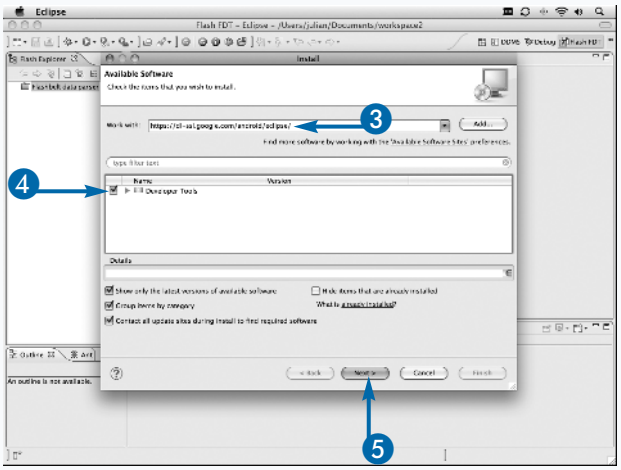
Get and Install the Plug-in

1. In Eclipse, click Help.
2. Click Install New Software.



The Install dialog box appears.

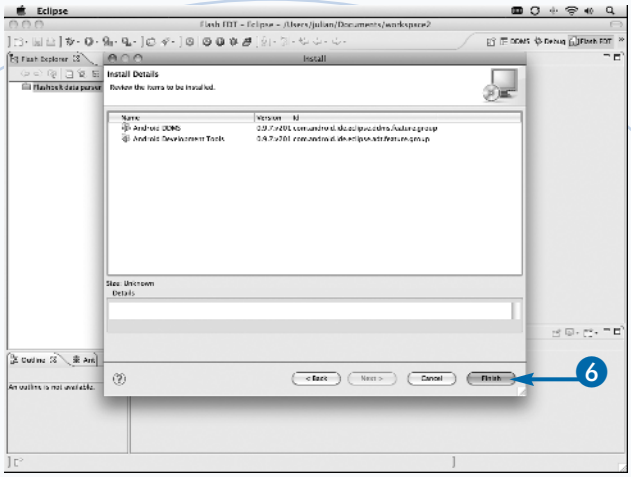
3. Type **https://dl-ssl.google.com/android/eclipse/**.
4. Click to check Developer Tools.
5. Click Next.



The Install Details page of the dialog box appears.

6. Click Finish.

   The plug-in begins to download and install.



Set the Android SDK Path

1. Click Eclipse ➔ Preferences from the main menu.

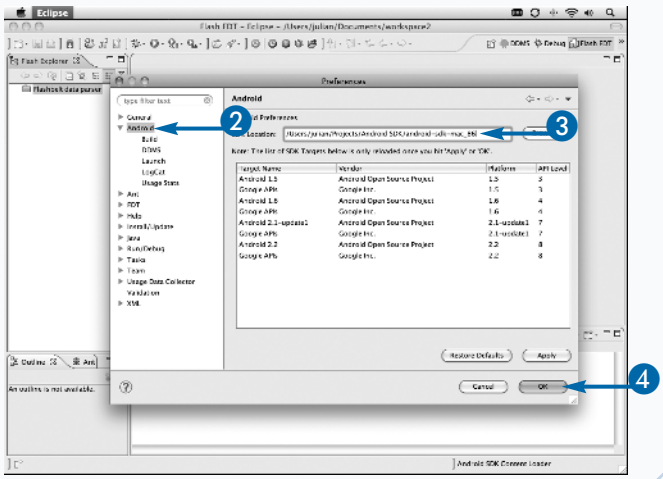
   The Preferences dialog box appears.

2. Click Android.

3. Type the location of the directory of your Android SDK.

4. Click OK.

   The path to the Android SDK is added.



Extra

The most useful feature of the DDMS perspective is the File Explorer panel. With a device selected, this panel will show the file system of that device. This can help you ensure that files are being created in the correct location. It also allows you to push and pull files to and from the device. By default, you will not have access to some of the folders on the device, such as the data directory. This is because that folder is owned by the root user of the device. In order to gain access to that folder, you must root your phone, which will void your warranty but also give you more access. Unless there is a very good reason for doing so, it is not recommended that you root your device. There are many tutorials online on how to do this if you are interested. Make sure to follow the instructions for your device, or it may result in a bricked device.

There are also many useful utilities in the DDMS perspective that can be used to debug your applications. For more details, see Chapter 16, “Debugging Your Application.”

Enable USB Debugging

In order to install applications on your device, you must first have it set up to enable debugging. Depending on which operating system you are currently developing on, the number of steps may vary.

If you are developing on a Windows computer, you will need to download and install the USB driver package using the Android SDK and AVD Manager. This process is the same as installing the additional Android SDK components. For more details, see the section "Get the Android SDK" earlier in this chapter. If you already completed the steps in that section and selected all the available packages, there is a good chance that you have already installed this. You can check to see if it has been downloaded by looking for the usb_driver directory in the root of your SDK. The process for installing the drivers is different for each version of Windows, and I suggest that you follow the instructions on the Android Developers site, http://developer.android.com/sdk/win-usb.html, in order to install them correctly.

If you are using a Mac OS X computer for development, you can skip having to install the USB drivers. The steps below should just work.

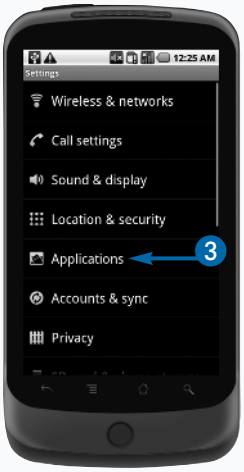
You can now connect your device to your computer with the USB cable included with your device. If it has connected correctly, you should see the USB symbol in the notifications area of the status bar. The next step is to turn on USB debugging when the device is connected. To do so, on your device, you go to the list of options that can be used during development and choose to debug applications on the device.

Enable USB Debugging

1. Connect your device with a USB cable.
2. Tap Settings.

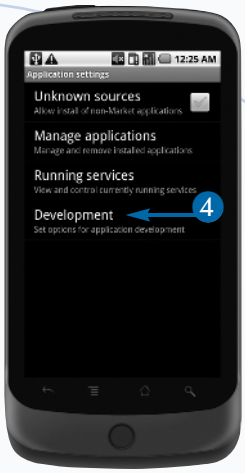
The Settings application is launched.

3. Tap Applications.



The Application Settings are displayed.

4. Tap Development.



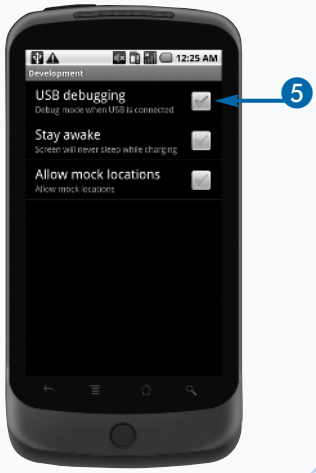
The Development settings are displayed.

5. Tap USB Debugging.

   A confirmation dialog box appears.

6. Tap OK.

   Your device is now ready for USB debugging.



Extra

There are two additional options in the Development list that are also very useful when developing and debugging applications. The Stay Awake option prevents the screen from going to sleep while it is connected to a USB cable or while charging. This can be useful when developing if you are testing and continuously reinstalling your application on the device. More times than not, the screen will go to sleep when you are compiling and installing your application. If you turn this on, make sure to turn it off when you are testing to see if your system idle code works correctly. For more details on setting the system Idle mode, see Chapter 12, "Using the Location and WiFi Features."

The third option is Allow Mock Locations. If this option is checked, you can simulate the device being at a specific geolocation coordinate. This is extremely useful when testing geolocation features of your application without physically being at a specific location. For more details on simulating geolocation coordinates on your device, see the section "Debug with the Android Eclipse Plug-in" in Chapter 16.

Create an Android Virtual Device

Android Virtual Devices (AVDs) are configured emulator instances, which enable you to set up an emulator to mimic the behavior of an actual device. Creating an Android Virtual Device gives you full control over the entire specifications of the emulator. This allows you to set up an emulator just like a device that you may want to test your application on.

Each Android Virtual Device is made of up several parts. The first is which version of the SDK you want to use on the device. In order to make sure that the device has Android 2.2 or higher installed, choose a target with an API level greater or equal to 8. Next, you can specify a specific skin and screen resolution for your device. Several of the popular device screen resolutions come preinstalled with the SDK; however, you can enter a custom resolution if you prefer. You can also specify the size of the SD card that you want your device to have, and you can select an image of an SD Card in order to have it be populated with files when created. You can also adjust the hardware profile of the device. There are a number of items that you can add to your device, such as support for a directional pad, touch-screen capability, a GPS, a camera, a trackball, and keyboard support.

Google has done a great job in allowing you to model your Android Virtual Device after an actual device as closely as possible. This keeps you from having to have a physical device for all the different Android devices in order to test your application. However, this does not replace testing on an actual device, which should always be done.

Create an Android Virtual Device

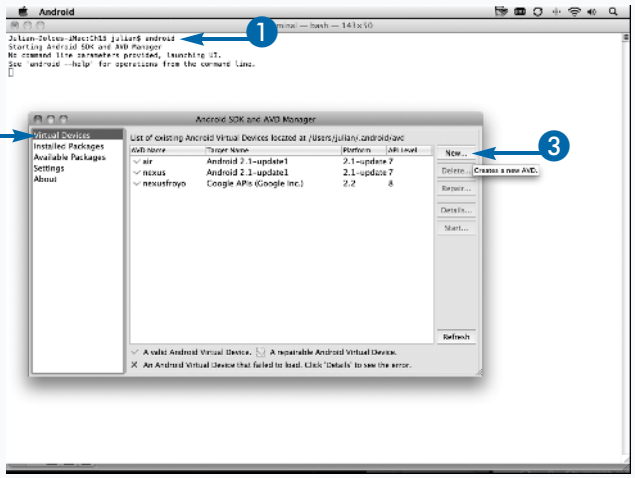
1. In a Mac OS X Terminal window, type **android** and press Enter.

   OR

1. In Windows, run the SDK Setup.exe executable in the root SDK directory.

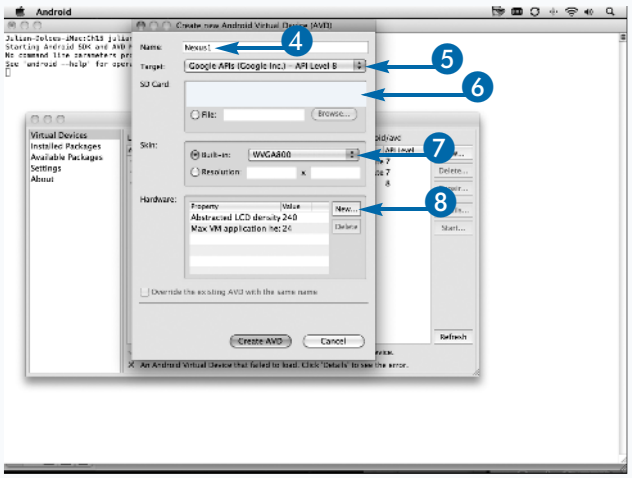
   The Android SDK and AVD Manager is launched.

2. Click Virtual Devices.
3. Click New.



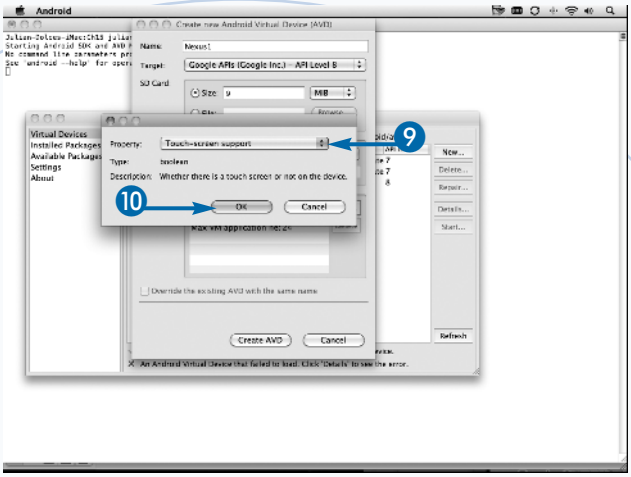
   The Create New Android Virtual Device (AVD) dialog box appears.

4. Type a name.
5. Click here and select a target, such as Google APIs (Google Inc.) – API Level 8.
6. Select a size for the SD Card, such as 9 MiB.
7. Click here and select a screen resolution, such as WVGA800.
8. Click New.



The hardware features dialog box appears.

9. Click here and select a hardware feature, such as Touch-Screen Support.
10. Click OK.
11. Repeat steps **8** to **10** for each hardware feature that you want to add.



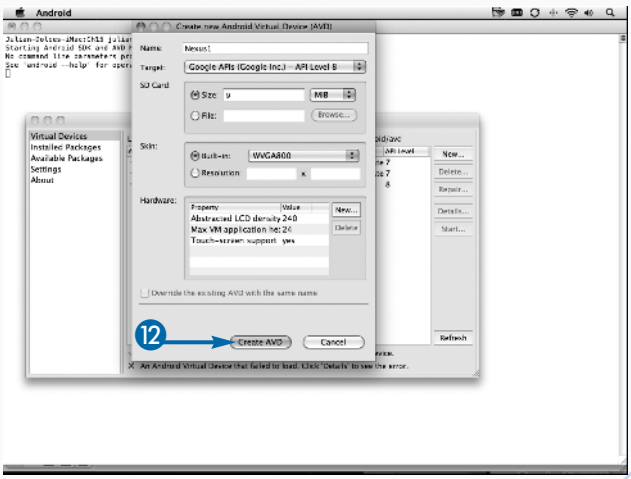
You are returned to the Create New Android Virtual Device (AVD) dialog box.

12. Click Create AVD.

A result dialog box appears.

13. Click OK.

The AVD is created.



Extra

In order to see a complete list of all the available Android Virtual Devices, you can enter the following command in a Terminal or command prompt window:

```
android list avds
```

This will print out a list of the AVDs as well as their configurations. The name, path to the .avd file, the target framework and API, and the skin are displayed for each AVD.

After some time, chances are that you will create a number of different AVDs in order to simulate the different devices on the market. As this list grows, you may want to remove any unwanted AVDs. The `android` tool has the ability to delete a specific AVD. The syntax for deleting a AVD is as follows:

```
android delete avd -n <name>
```

The `name` argument can be found when you use the `list` command. For example, if I had an AVD named `nexusone`, I would use the following command in order to delete it:

```
android delete avd -n nexusone
```

Start the Emulator

After you have created your Android Virtual Device, you can start it. Starting an Android Virtual Device causes the emulator application to be launched with the configuration of the device that you have set. If you have not created an Android Virtual Device, see the preceding section, "Create an Android Virtual Device."

There are two main ways to start an Android Virtual Device, from the command line and from the Android SDK and AVD Manager window. From the command line, you use the emulator executable in the tools folder of the Android SDK. The emulator executable has many options you can use when starting up an Android Virtual Device. The main option you will want to use is the `-avd` switch. This switch enables you to specify the name of the Android Virtual Device that you want to start. If you are unsure of what the name of the Android Virtual Device is, you can get a list of all available devices using the command mentioned earlier, `android list avds`.

If you are using the Eclipse plug-in or are not comfortable with the command line, you can use the Android SDK and AVD Manager in order to start the Android Virtual Device that you want. The Virtual Devices selection displays a list of all the available devices. This is similar to the `android list avds` command.

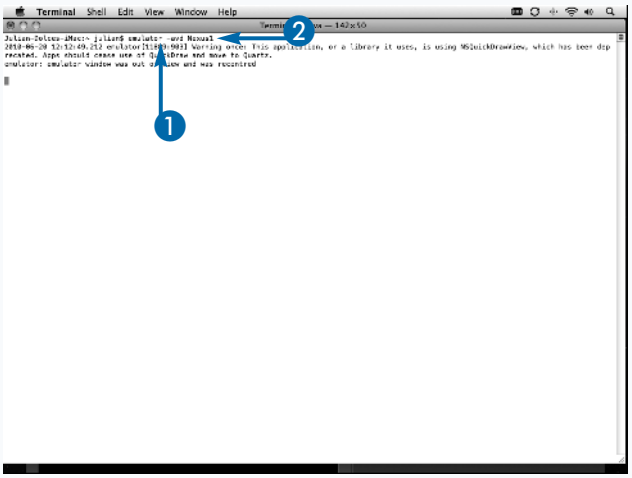
It may take a while for your device to fully boot up to the home screen of the device. Also, before you can install your applications onto the device, you will need to install the AIR Runtime for the emulator.

Start the Emulator

Start the Emulator with the Command Line

1. In a Terminal or a command prompt window, type **emulator –avd**.
2. Type the name of the AVD that you want to start, such as Nexus1.
3. Press Enter.

   The emulator launches the AVD.

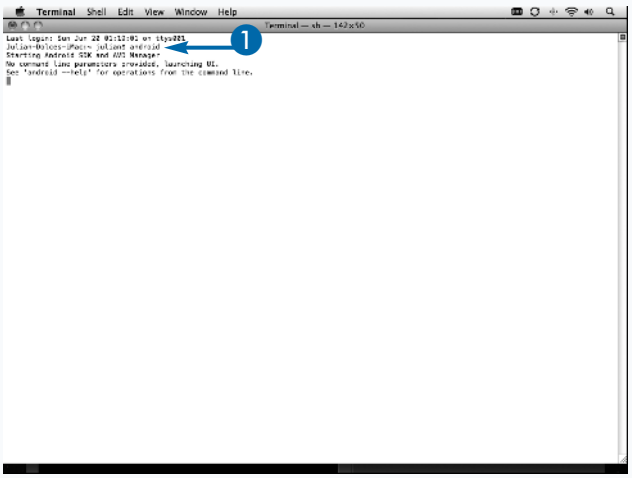


Start the Emulator with the Android SDK and AVD Manager

1. In a Mac OS X Terminal window, type **android**.

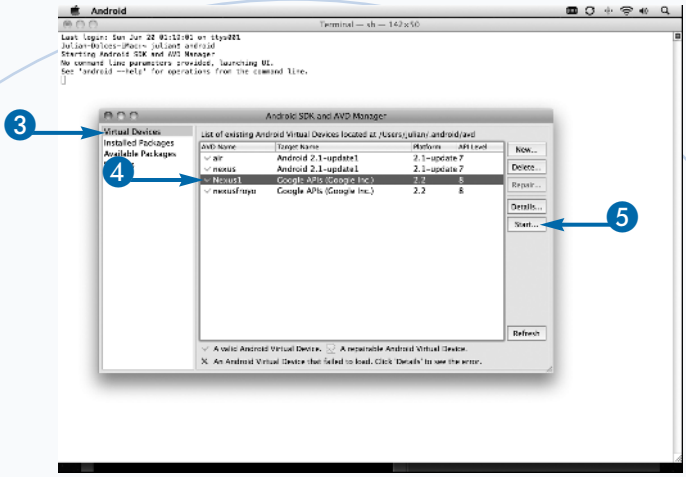
   OR

1. In Windows, run the SDK Setup.exe executable in the root SDK directory.
2. Press Enter.



The Android SDK and AVD Manager appears.

3. Click Virtual Devices.
4. Click a device.
5. Click Start.



The emulator is launched.

Extra

After your emulator is launched, you can telnet into it to perform more commands on it. When the emulator is fully booted to the home screen, you will notice that there is a number before the name in the title bar of the emulator window. In the example in this section, that number is 5554. This number is the port that the emulator is connected to. You can telnet into the emulator by typing the following command into a Terminal window:

```
telnet localhost 5554
```

This will start a telnet session with the emulator, which will allow you to interact with the emulator a number of ways. After you are connected, type **help** into the Terminal window to see a list of available commands.