

Mobile Development

using Xamarin Forms

Ahmed El-Araby | Feb-2019

**Introduction of mobile development**

Mobile development has witnessed incredible development in the decade, from introducing new frameworks to enhance an existing one. Thus, it is very important to choose the right technology/framework when starting a new project. in my opinion, all these methods and frameworks can be categorized into four categories, Native development, hybrid, flutter, and cross-platform.

Mobile Development Options

**Hybrid**

(JavaScript and HTML5)

**Flutter**

(Dart)

**Native**

(Swift & Java)

**Cross-Platform**

(Xamarin, React Native, Titanium)

**Native:**

Developing Mobile application using the platform native library, programing language, and compilers. For example, to develop iOS application you’ll need a MacOS with Xcode installed and writing Swift/Objective-C code. However, for Android you need to use Android Studio with Java programming language.

**Hybrid:**

Developing a hybrid application that can be executed on iOS, Android, Windows desktop, Linux … etc. or any other operating system with web viewer.

There are many ways and tools to develop hybrid application but all of them will be converted to JavaScript and HTML at the end. Moreover, most of the tools are using Crodova to convert the compiled code to mobile application, which can be uploaded to App Store or Google Play.

Hybrid apps load in a browser-like component called webview, they are only as good as the webview. Webview is responsible for displaying the UI and for running Javascript code.

Some of this tools and frameworks are Ionic, Angular, Vue and React.

**Flutter:**

It is a framework from Google help you to develop mobile application for iOS and Android in Dart programming language. Which can give the developer the ability to control each and every pixel on the screen.

The compiler will compile your project to machine-code directly, which provide high performance as well.

**Cross-Platform:**

This method help developer to write an application using their preferred programming language and then the compiler will convert it to native library, which can be compiled via native compiler to machine code.

This method will make the application looks and feels much like native applications. And also will have the native limitations.

**How Does Xamarin Work?**

Xamarin offers two commercial products: Xamarin.iOS and Xamarin.Android. They are both built on top of Mono, an open-source version of the .NET Framework based on the published .NET ECMA standards. Mono has been around almost as long as the .NET framework itself, and runs on nearly every imaginable platform including Linux, Unix, FreeBSD, and Mac OS X.

On iOS, Xamarin’s Ahead-of-Time (AOT) Compiler compiles Xamarin.iOS applications directly to native ARM assembly code. On Android, Xamarin’s compiler compiles down to Intermediate Language (IL), which is then Just-in-Time (JIT) compiled to native assembly when the application launches.

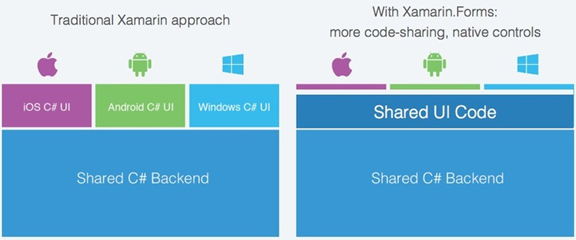
In both cases, Xamarin applications utilize a runtime that automatically handles things such as memory allocation, garbage collection, underlying platform interop, etc.

**Why using Xamarin?**

1. Using platform-specific compilers.
2. Direct access to native APIs. These are the C# object libraries that give developers access to iOS SDK and the Android SDK, respectively.
3. Native performance.
4. Develop using the same programming language you have experience with.
5. Developing using preferred IDE.
6. No need to learn iOS or Android platforms and SDK to develop native application for both operating systems. For example, you can use Xamarin Agent on Mac to deploy and test.

**Xamarin Forms:**

Xamarin.Forms is a complete array of instruments for the creation of cross-platform software. It works on top of the Xamarin.iOS and Xamarin.Android libraries and allows writing front ends in C# and XAML and may even share this code between implementations. Applications created with Xamarin.Forms are fully endowed with all the characteristics inherent to the best cross-platform solutions. In particular, they allow developers to save time and money (about 80-95% of the code is cross-platform, the remaining 5-20% contain commands to establish connections with native APIs).

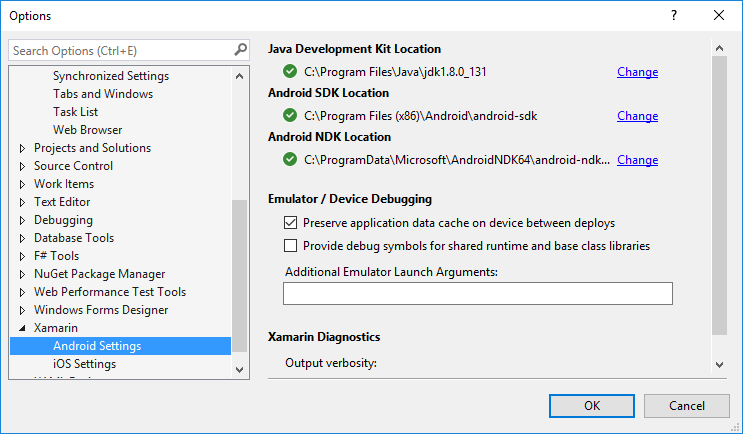


**Installation and Configuration:**

Windows Installation

Install Visual Studio 2017 with Xamarin components.

Xamarin.Android uses the Java Development Kit (JDK) and the Android SDK to build apps. During installation, the Visual Studio installer places these tools in their default locations and configures the development environment with the appropriate path configuration. You can view and change these locations by clicking **Tools > Options > Xamarin > Android Settings**:



Mac Installation

Installing The following is required to develop iPhone application:

* the latest iOS SDK
* the latest version of Xcode
* macOS High Sierra (10.13) and above

Android SDK Setup

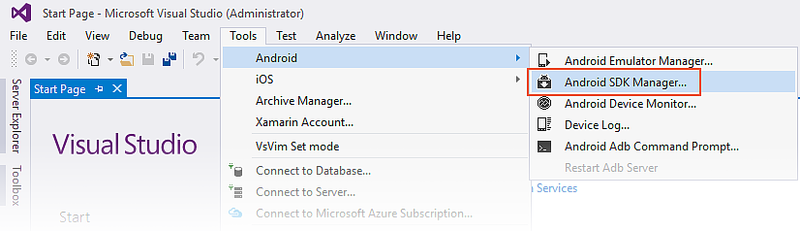
Visual Studio includes an Android SDK Manager that replaces Google's standalone Android SDK Manager.

SDK Manager

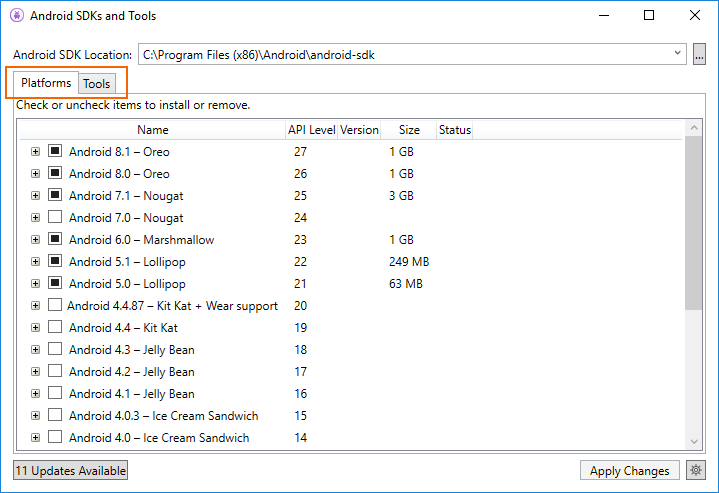
To start the SDK Manager in Visual Studio, click Tools > Android > Android SDK Manager:

**SDK Manager**

To start the SDK Manager in Visual Studio, click **Tools > Android > Android SDK Manager**:

[](https://docs.microsoft.com/en-us/xamarin/android/get-started/installation/android-sdk-images/win/02-sdk-manager-menu-item.png#lightbox)

The Android SDK Manager opens in the **Android SDKs and Tools** screen. This screen has two tabs – **Platforms** and **Tools**:

[](https://docs.microsoft.com/en-us/xamarin/android/get-started/installation/android-sdk-images/win/03-sdk-manager-platforms.png#lightbox)

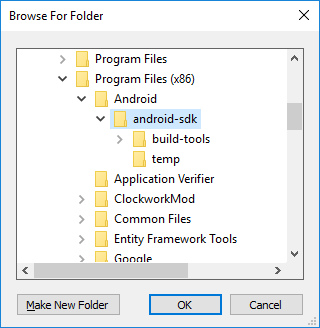
The **Android SDKs and Tools** screen is described in more detail in the following sections.

**Android SDK location**

The Android SDK location is configured at the top of the **Android SDKs and Tools** screen, as seen in the previous screenshot. This location must be configured correctly before the **Platforms** and **Tools** tabs will function properly. You may need to set the location of the Android SDK for one or more of the following reasons:

1. The Android SDK Manager was unable to locate the Android SDK.
2. You have installed the Android SDK in a alternate (non-default) location.

To set the location of the Android SDK, click the ellipsis (…) button to the far right of **Android SDK Location**. This opens the **Browse For Folder** dialog to use for navigating to the location of the Android SDK. In the following screenshot, the Android SDK under **Program Files (x86)\Android** is being selected:

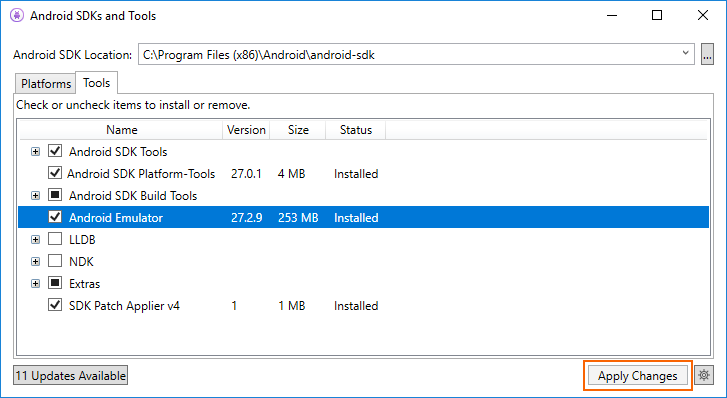


When you click **OK**, the SDK Manager will manage the Android SDK that is installed at the selected location.

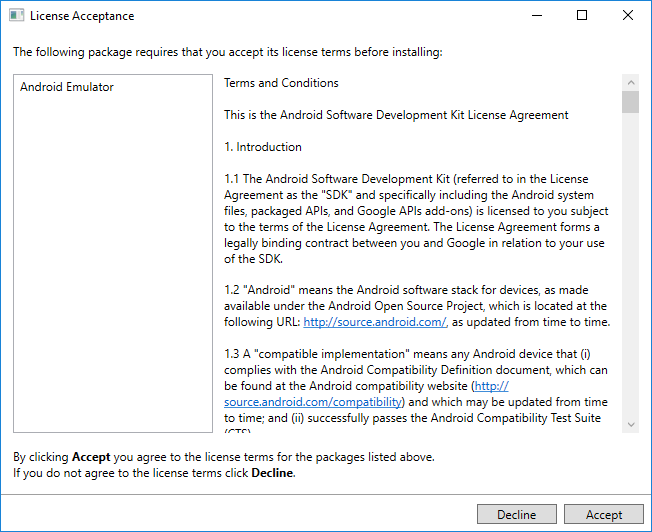
**Tools tab**

The **Tools** tab displays a list of *tools* and *extras*. Use this tab to install the Android SDK tools, platform tools, and build tools. Also, you can install the Android Emulator, the low-level debugger (LLDB), the NDK, HAXM acceleration, and Google Play libraries.

For example, to download the Google Android Emulator package, click the check mark next to **Android Emulator** and click the **Apply Changes** button:

[](https://docs.microsoft.com/en-us/xamarin/android/get-started/installation/android-sdk-images/win/06-install-emulator.png#lightbox)

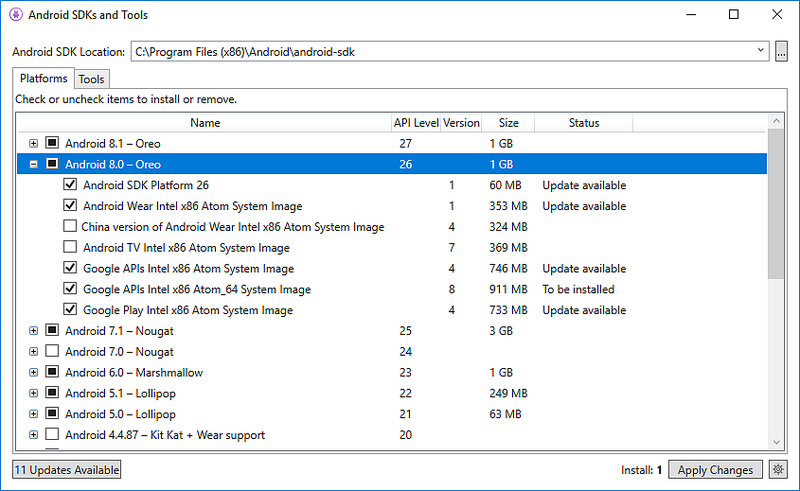
A dialog may be shown with the message, *The following package requires that you accept its license terms before installing*:



Click **Accept** if you accept the Terms and Conditions. At the bottom of the window, a progress bar indicates download and installation progress. After the installation completes, the **Tools** tab will show that the selected tools and extras were installed.

**Platforms tab**

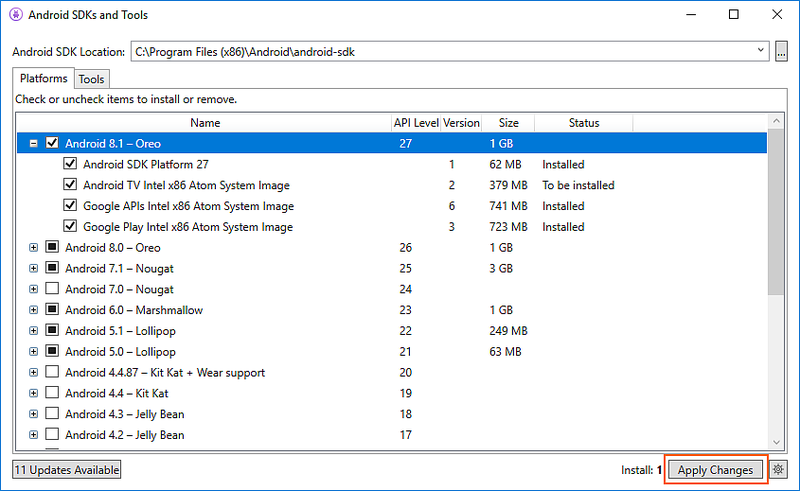
The **Platforms** tab displays a list of platform SDK versions along with other resources (like system images) for each platform:

[](https://docs.microsoft.com/en-us/xamarin/android/get-started/installation/android-sdk-images/win/08-platforms-pane.png#lightbox)

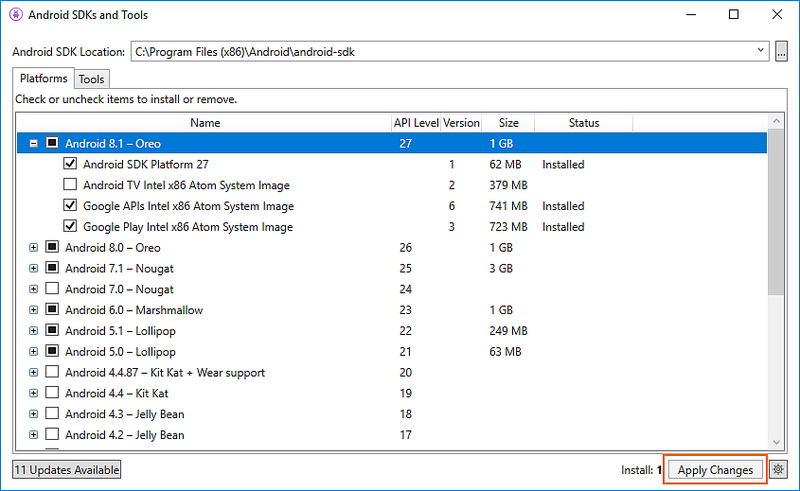
This screen lists the Android version (such as **Android 8.0**), the code name (**Oreo**), the API level (such as **26**), and the sizes of the components for that platform (such as **1 GB**). You use the **Platforms** tab to install components for the Android API level that you want to target. For more information about Android versions and API levels, see[Understanding Android API Levels](https://docs.microsoft.com/en-us/xamarin/android/app-fundamentals/android-api-levels).

When all components of a platform are installed, a checkmark appears next to the platform name. If not all components of a platform are installed, the box for that platform is filled. You can expand a platform to see its components (and which components are installed) by clicking the **+** box to the left of the platform. Click **-** to unexpand the component listing for a platform.

To add another platform to the SDK, click the box next to the platform until the checkmark appears to install all of its components, then click **Apply Changes**:

[](https://docs.microsoft.com/en-us/xamarin/android/get-started/installation/android-sdk-images/win/09-adding-a-platform.png#lightbox)

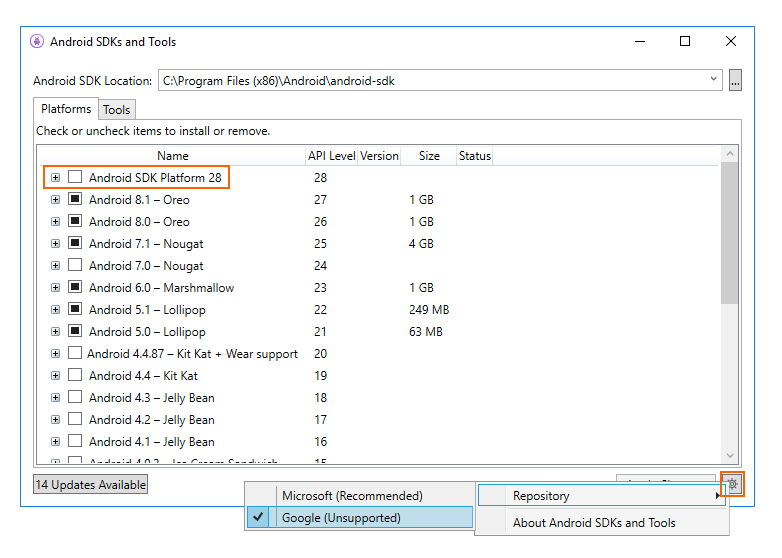
To install only specfic components, click the box next to the platform once. You can then select any individual components that you need:

[](https://docs.microsoft.com/en-us/xamarin/android/get-started/installation/android-sdk-images/win/10-adding-some-components.png#lightbox)

Notice that the number of components to install appears next to the **Apply Changes**button. After you click the **Apply Changes** button, you will see the **License Acceptance** screen as shown earlier. Click **Accept** if you accept the Terms and Conditions. You may see this dialog more than one time when there are multiple components to install. At the bottom of the window, a progress bar will indicate download and installation progress. When the download and installation process completes (this can take many minutes, depending on how many components need to be downloaded), the added components are marked with a checkmark and listed as **Installed**.

**Respository selection**

By default, the Android SDK Manager downloads platform components and tools from a Microsoft-managed repository. If you need access to experimental alpha/beta platforms and tools that are not yet available in the Microsoft repository, you can switch the SDK Manager to use Google's repository. To make this switch, click the gear icon in the lower right-hand corner and select **Repository > Google (Unsupported)**:

[](https://docs.microsoft.com/en-us/xamarin/android/get-started/installation/android-sdk-images/win/11-google-repo-w157.png#lightbox)

When the Google repository is selected, additional packages may appear in the **Platforms** tab that were not available previously. (In the above screenshot, **Android SDK Platform 28** was added by switching to the Google repository.) Keep in mind that use of the Google repository is unsupported and is therefore not recommended for everyday development.

To switch back to the supported repository of platforms and tools, click **Microsoft (Recommended)**. This restores the list of packages and tools to the default selection.

**References:**

<https://applikeysolutions.com/blog/xamarin-forms-vs-xamarin-native-what-fits-you-best>

<https://www.mobiloud.com/blog/native-web-or-hybrid-apps/>

<https://flutter.io/>

<https://docs.microsoft.com/en-us/xamarin/cross-platform/get-started/introduction-to-mobile-development>

<https://www.appcelerator.com/titanium/titanium-sdk/4/>

“Creating Mobile Apps with Xamarin.Forms”, Xamarin 2016

“Xamarin.Forms Notes for Professionals”, GoalKicker 2018