Windows Setup Checklist

You will need a PC running Windows 7 or Windows 8.

1. (*Optional*) Install Visual Studio 2012 or 2013. Xamarin integrates into the Microsoft IDE tools and allows you to develop Xamarin.Android and Xamarin.iOS applications directly with VS. If you do not have access to Visual Studio, or prefer not to use it, Xamarin Studio is also available and will be installed with the Xamarin tools, however you will be limited to Xamarin.Android.
2. Install Xamarin tools using the installer. You can download the Xamarin unified installer from xamarin.com/download. The installer will automatically download any missing tools and SDKs needed for iOS/Android development.
    a. **Note:** this step will likely take a while depending on what is already installed on the computer since it has to download fairly large SDK installs to configure the machine.
3. Launch Visual Studio or Xamarin Studio - it will be in your Start Screen / Start Menu, or you can locate it with the search features of Windows.
4. Verify you are on the *stable* (this is the default). You can check this in the **Tools > Options > Xamarin** menu.
5. (*Optional*) Install GotoMeeting from http://support.citrixonline.com/en_US/gotomeeting/downloaddocument/GTMD00063, this will allow you to stream the training onto your laptop's screen during the sessions which may be easier to see than the projected image. If you would prefer to use an iOS or Android device, you can download the GotoMeeting application from the AppStore or Google Play store.

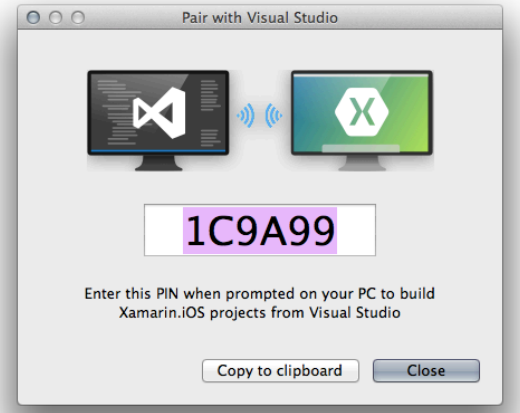
Pairing Windows with a Mac host for iOS development

If you plan to do any development for iOS, you will need Visual Studio 2012 or 2013 **and** a Mac host with the Xamarin tools installed. You will do your development on the Windows computer with Visual Studio, and the Mac is used to build and package up the iOS application using Apple's tools.

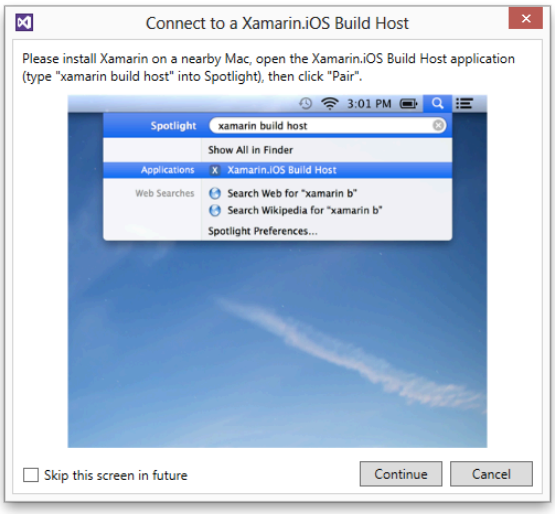
The computers will need to be on the same network and be able to ping each other. This section details how to connect the two computers together through the Xamarin Build Host and assumes you have already setup both the Mac and Windows computers properly.

You can connect your Mac to the Windows computer using the Xamarin Build Host with the following steps:

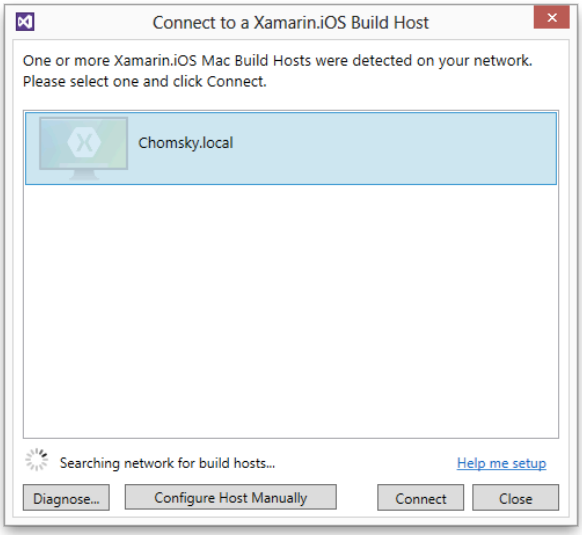
1. On the Mac, open the Xamarin Build Host (it is located in the Applications folder, or you can use Spotlight to find it).
2. If it is already paired to a different installation, then you will need to unpair it first – you can only connect to a single instance of Visual Studio (one Mac per VS installation).
3. Click the *Pair* button to initiate the pairing process, this will present a network code which you will need to enter on your Windows system:



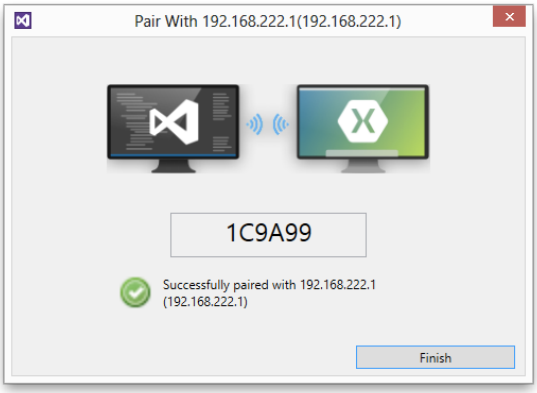
4. Switch to Windows and open Visual Studio, in either the Tasks pane, or the **Tools > Options > Xamarin** menu, you can pair the computer to the Mac host. You can also attempt to open an iOS solution – this will initiate the pairing process as well.



5. Select **Continue** and then select your Mac on the network (or type the specific IP address in if the auto-detect feature isn't working).



6. Select the Mac and click **Connect** to proceed. It will then prompt for the PIN displayed on the Mac host, enter the pin and click **Pair** to finish the connection.



Setup Android Emulators

To test Android applications, you can either deploy to a physical device (preferred), or use an Android emulator. If you plan to use an emulator, we recommend you use **Genymotion** as it delivers higher performance over the Android SDK emulator.

1. Download the Genymotion install from www.genymotion.com, for Windows you can download the standalone package, which includes Virtual Box. For OS X, you will need to install Virtual Box separately *before* you install Genymotion.
2. Once installed, run the application and download a pre-configured image, there are several to choose from.
3. Launch the image from the Genymotion app by selecting it in the list and clicking the Play button.
4. It should then show up in the emulators list in Xamarin or Visual Studio.

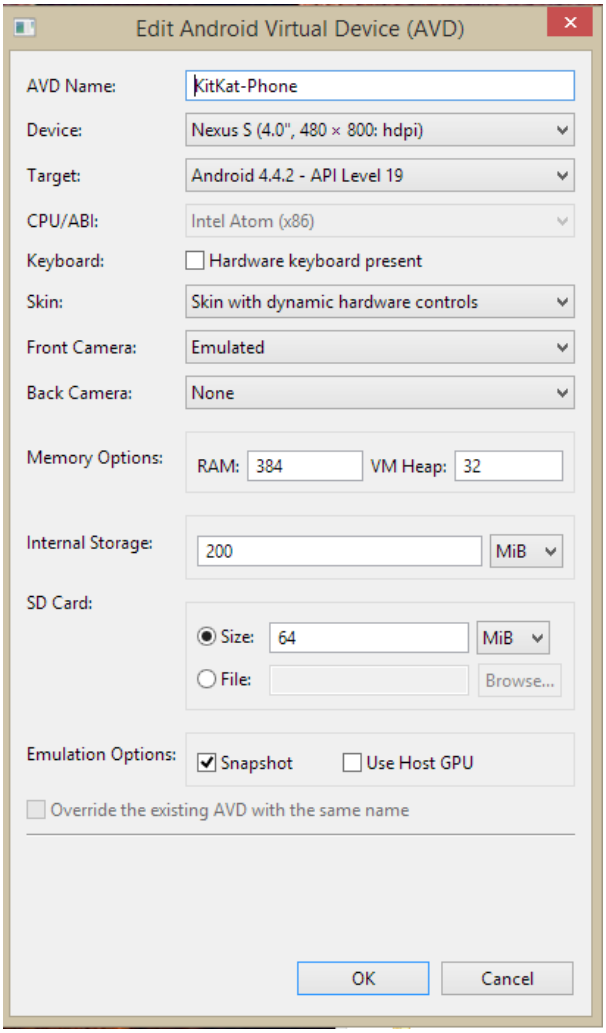
Warning!
Genymotion does not play nice with Hypervisor support on Windows – you must disable the hypervisor support using **bcdedit** before launching Genymotion. You can do this from an admin command prompt with the following command; you will need to reboot the machine for this to take effect.

```
bcdedit /set {current} hypervisorlaunchtype off
```

Android SDK Emulators

The Xamarin installer will create a few emulated devices for you, but you will want to adjust these and create one with a newer version of Android. We recommend a phone device running at least 4.4.2.

To create or adjust an Android emulator, you must run the Android SDK Emulator Manager. This can be started in Visual Studio with **Tools > Android > Open Android Emulator Manager** menu option. An example definition is shown below:



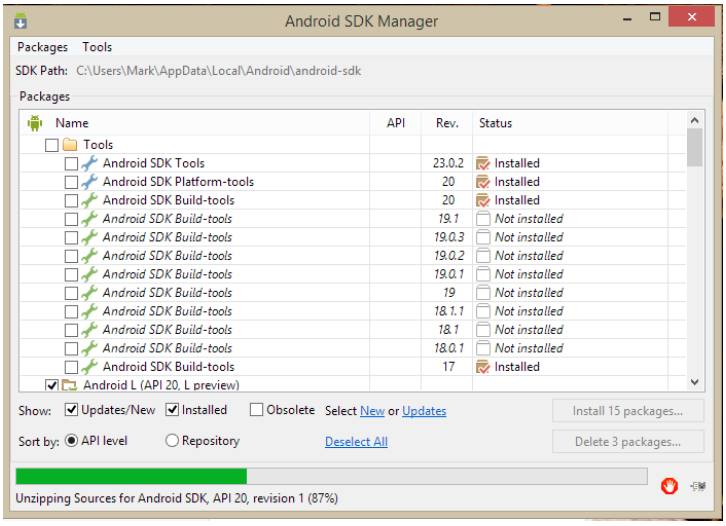
Out of the box Google Android Emulators are very slow. To improve this, you can install the **Intel HAXM Drivers.** This additional install provides hardware acceleration for x86-based emulators on Intel VT-enabled systems. The HAXM drivers are free to use and published by Intel. If you are using the Android SDK emulator, this is a must.

1. Close any running emulators
2. From the Intel website, download the latest HAXM virtualization engine from https://software.intel.com/en-us/android/articles/intel-hardware-accelerated-execution-manager.
3. Install the HAXM engine, and restart your computer if prompted.
4. Change any Android image you have created to use the Intel Atom (x86) CPU image as shown in the above screenshot – the acceleration only works for x86 images.

Warning!
HAXM run simultaneously with VirtualBox can cause stability issues. Both can exist together on the same machine, but it is best not to run both Genymotion- and HAXM-based emulators at the same time.

Verify your Setup

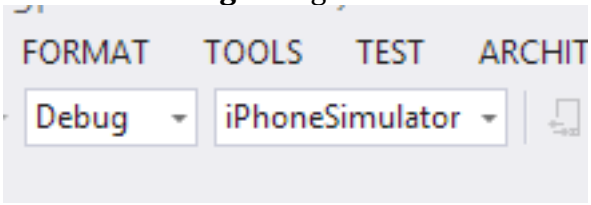
1. Launch Xamarin Studio (OS X or Windows) or Visual Studio (Windows).
2. The IDE should prompt you to either start a trial with Xamarin, or enter to your registered Xamarin account. Go ahead and do one of these two steps so you be able to build full applications, including the T-shirt test app.
3. Verify that you have all the Android SDK versions you will need. We recommend installing 4.0 through 4.4 (or even Android "L" preview if you want to try some of the new features). You can get to the Android SDK options using the **Tools > Android > Start Android SDK Manager** in Visual Studio.



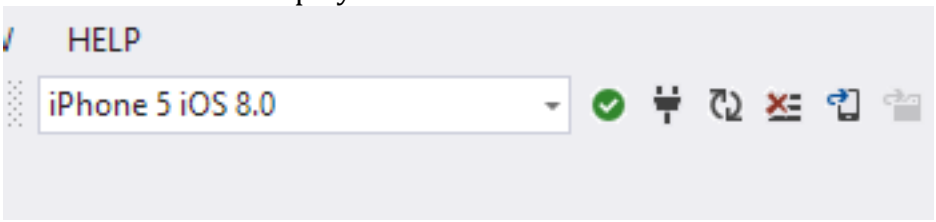
4. Download the Xamarin T-shirt application from https://xamarin.com/c-sharp-shirt, unzip it onto your desktop (or some other easily accessible location) and open the solution.
    a. **Note**: Be careful about path lengths on Windows – mobile projects tend to create deep subdirectories and you can quickly exceed the path length on Windows machines. It is best to place the solutions into shorter paths such as your desktop or right in the root of the drive.
5. Open the solution (.sln) using your IDE of choice and following the instructions below for iOS, Android or both.

Testing iOS on Windows with Visual Studio

1. Make sure your setup is connected to the Mac build host (see above instructions to verify this).
2. Make sure the **XamarinStore.iOS** project is the active project in Visual Studio (you can right-click and choose *Set As Startup Project* to change it if necessary).
3. Select the **Debug** configuration and **iOS Simulator** from the toolbar.



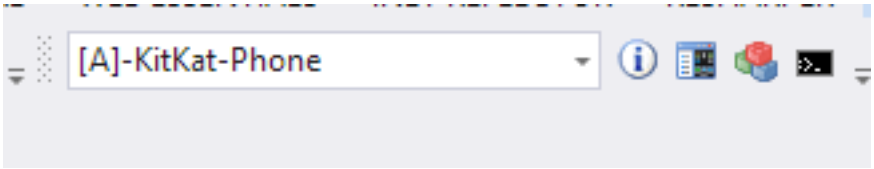
4. Select the iPhone 5 iOS 8.0 simulator from the iOS toolbar. If you don't see the toolbar, right click in an open space in the toolbar area and make sure **iOS** is checked for display.



5. Build and run the application by clicking the **Start** button in the toolbar – this should launch the iPhone simulator on the Mac.

Testing Xamarin.Android on Windows

1. Make sure the **XamarinStore.Droid** project is the active project in Xamarin or Visual Studio (you can right-click and choose *Set As Startup Project* to change it if necessary).
2. Select the **Debug** configuration and **AnyCPU** from the toolbar.
3. Select an available Andoid emulator in the Android toolbar. If you don't see the toolbar choices, right click in an open space in the toolbar area and make sure **Android** is checked for display.

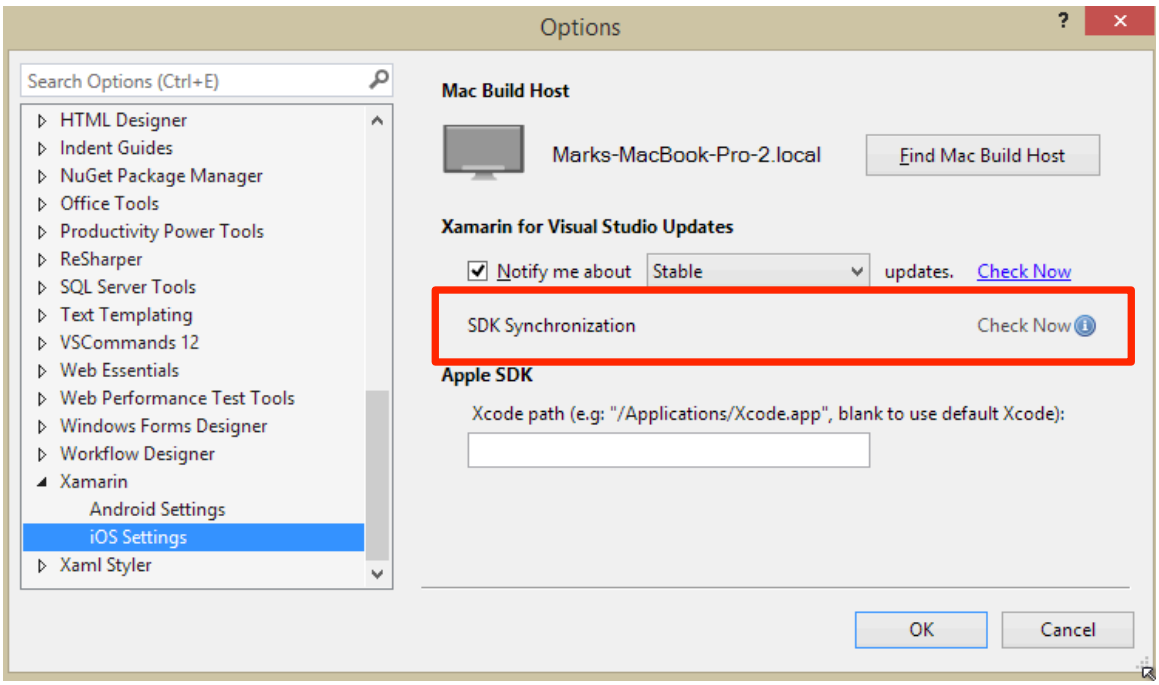


4. If you do not see any emulators, then you will need to configure at least one You can open the Android SDK tool with the **Tools > Android > Open Android Emulator Manager** menu option as described above.
5. Build and run the application by clicking the **Start** button in the toolbar – this should launch the Android emulator (if it's not running), install the application and then run it.

Troubleshooting

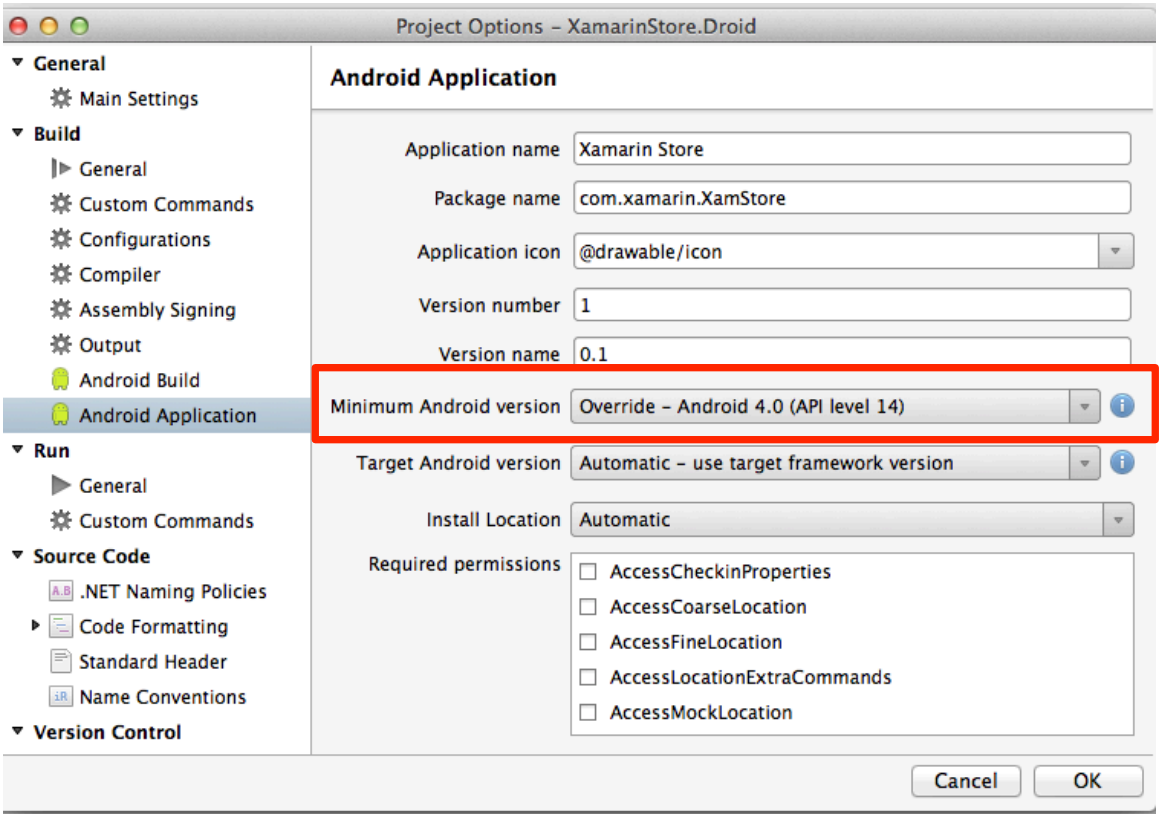
The most common issues encountered are: Missing SDK versions and Android emulator issues.

If the iOS Application fails to build on Visual Studio verify that the iOS SDK on the Mac and the iOS SDK on the Windows machine are the same – this means you are running in the same Xamarin channel (we recommend *stable*) and the two machines are synchronized. You can check this in the **Tools > Options > Xamarin > iOS** dialog – click the "Check Now" option next to SDK Synchronization.



If the Android application fails to build on either environment, then verify you have the proper Android SDK versions installed. You will often get an error that reads something like "The required Android SDK x.xx is missing…".

The T-Shirt application is setup to require Android 4.0, but you can change it to a newer version through the project options (just double-click on the XamarinStore.Droid project, or right click and select Properties). The version is in the **Android Application** section in Xamarin and Visual Studio:



Alternatively, you can use the Android SDK Manager to download the required SDK libraries for 4.0 and then you should be able to build the application.

For emulator issues, we recommend launching the Android emulator before you run your program and then *keep it running*. If Xamarin Studio or Visual Studio fails to detect the emulator, then shutdown Xamarin Studio and restart it. It should detect it on launch and then the emulator will show up in the available choices in the toolbar.

Note that the emulator *must be running on the same machine*. So, if you are running Visual Studio in a Windows VM, you will need to launch an emulator in that VM to see it. There is a technique you can use to connect to a networked instance of an emulator using the **adb** command-line tool in the SDK. Ask the lab facilitator for help if you want to try this advanced approach. We recommend you just run the emulator in the same machine for simplicity.