# Setting up your device for AR

## General Notes

It is recommended to work with Androids for this course. iOS is a littler trickier to get started and often has unforeseen problems down the line. However, if you feel comfortable with this then follow the IOS instructions, but please note that these instructions are from 2020, and there may be some differences due to new unity engine versions.

The main difference though is going to be in the setup process – once Unity is set up to connect with your device, it should be possible to follow the course through as delivered, except you will have to tune the example programs on the canvas so they are configured for iOS rather than Android. This setup guide will show you how to do that.

If this is going to be an issue for the course, please drop me an email at [natashab@chalmers.se](mailto:natashab@chalmers.se).

Please give this a try earlier rather than later, even if it’s just opening up an empty Unity game on your device. In our first class I will consider it a win if everyone is able to get an AR application deployed to their device, but even that depends somewhat on identifying issues ahead of the lecture.

## Unity

I’m going to ask everyone to use a Unity version **equal to or greater than** **2022.1.16f1** for this part of the course. This is the version used for module 3.

you use an earlier version, it’s much more complicated to get the correct packages installed and working correctly and it varies from version to version and OS to OS so I can’t really help. If you have the Unity Hub, you can install this as an additional version of the IDE – that’s the path I’d recommend. Make sure you install support for Android / iOS as appropriate.

# Android

If you’re working on an Android device, you’ll need to set yourself up with the Android Debug Bridge (ADB) for testing purposes. Instructions for that can be found here. Depending on your version, the build number might be under **Software Information** in the **About Phone** section.

<https://www.xda-developers.com/install-adb-windows-macos-linux/>

Debugging a program working on a phone can be very tricky, but ADB allows you to reach into the phone and get a live feed of debugger information like so:

adb logcat -s Unity ActivityManager PackageManager dalvikvm DEBUG

You will also need to set up your phone so that it can be used as a developer device:

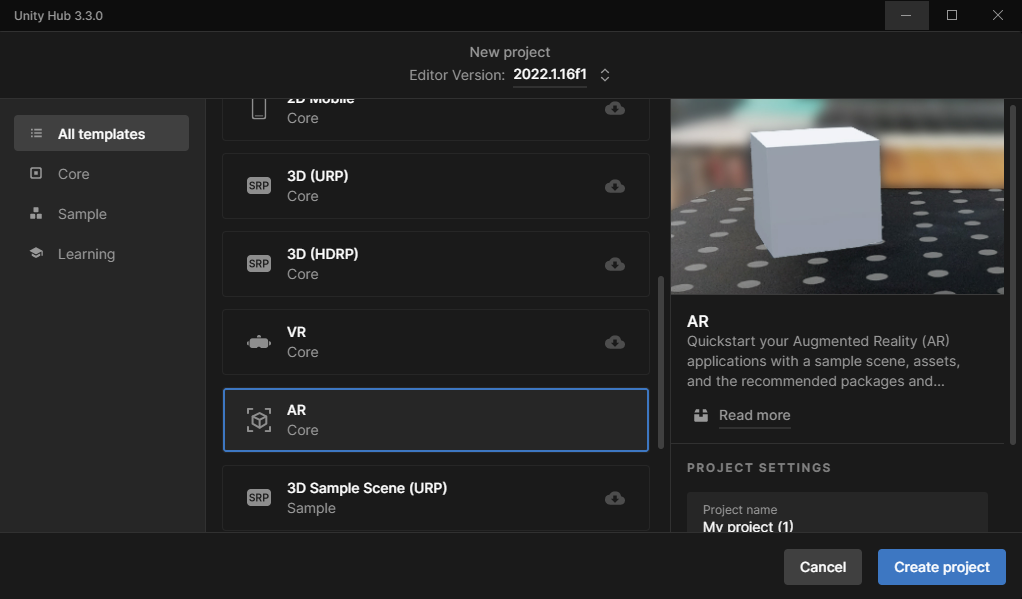
<https://developer.android.com/studio/command-line/adb>

You’ll also need the Unity AR Foundation toolkit – see below for the instructions.

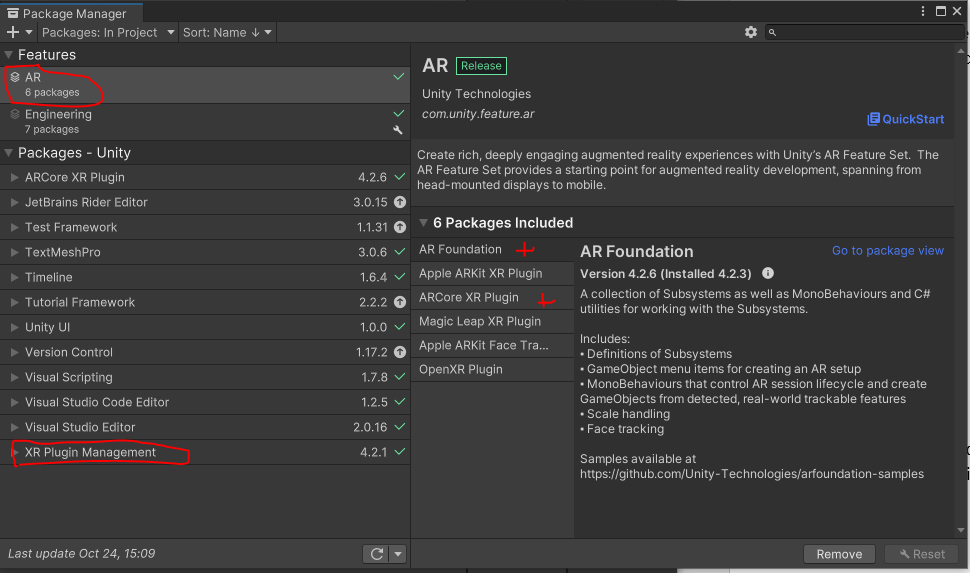
<https://developers.google.com/ar/develop/unity-arf/getting-started-ar-foundation>

Once you’ve got all that, you should be able to start your first Unity AR project. However, there are some things you’ll need to enable to get it working.

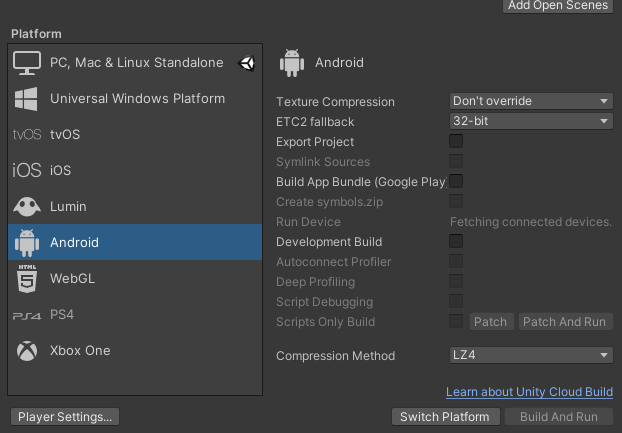
My suggestion is when creating a new project select the AR Core Template, this will include most of the settings needed to run the AR on a device.



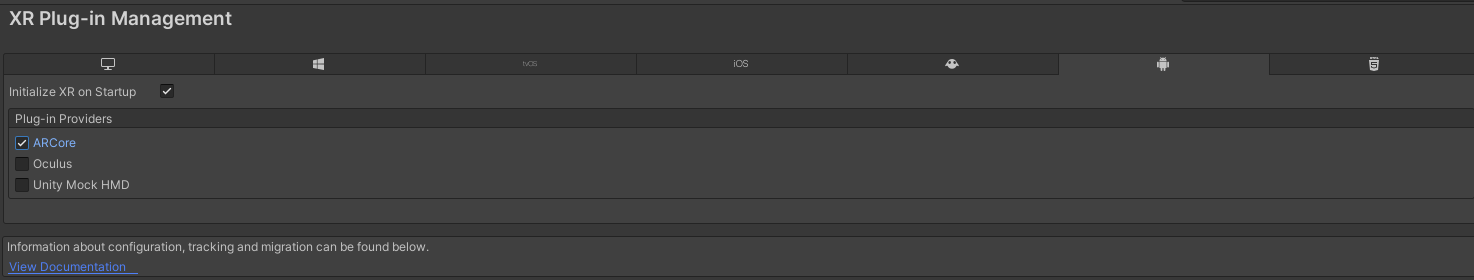
In this package there is a new layout in the Package Manager. Make sure that AR is fully included in the project and the XR Plugin Management is up-to-date, specifically make sure all red highlighted areas are installed, included and up to date.



You’ll also need to change the build settings a little to make sure it works properly when deployed to your device. File -> Build Settings from the menu and change to Android. If it is not available in the list, then add the android build settings from the unity hub, for your engine version.

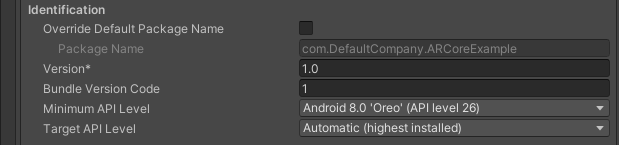


You then need to go into the player settings and change a few options. You may need to restart your project to get the options to be recognised. Go to XR Plugin Management from your project settings an enable ARCore, if you created the project from the AR Core template you should have a pop-up on opening the project which tells you to do this (ps. This is only available once android build settings have been installed onto your engine, if it was installed correctly and is not showing then close and re-open the project):

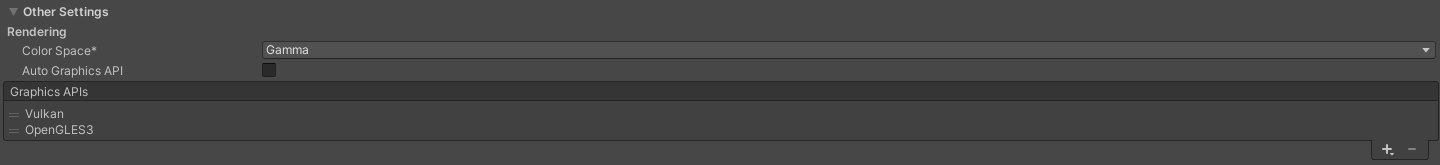


And you’ll also need to change a few other settings in Player -> Android (tab) -> Other Settings:

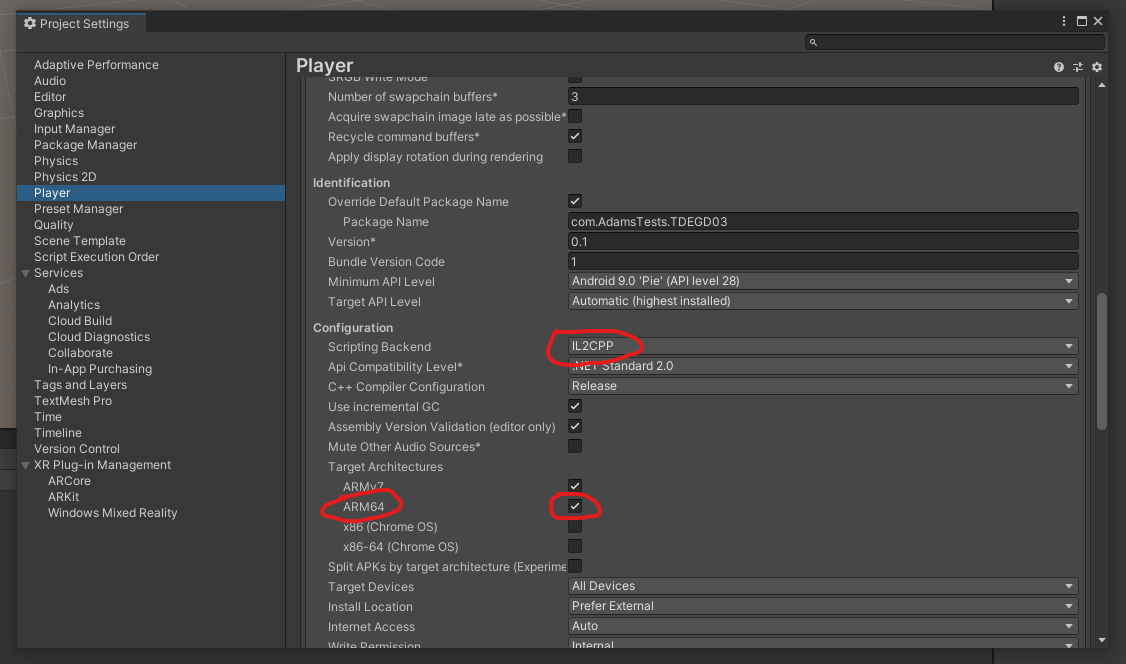
* Set minimum API level to 26



* If Vulkan is in Graphical APIs, then remove Vulkan from the list.



For those running a 64-bit ARM processor, there are a few more changes you’ll need to make sure are correctly set:



And with all of **that** you should be able to get your first Unity AR project running on your device!

However, there are some more rules due to new updates from Unity and AR. To fix any errors when building the project, follow these instructions:

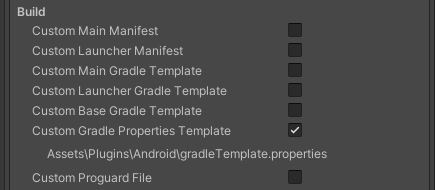
* Create a new Keystore for your project.

<https://docs.unity3d.com/Manual/android-keystore-create.html>

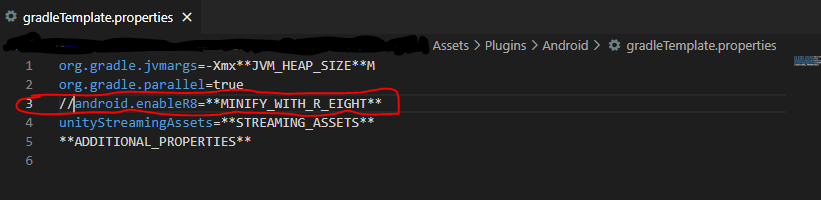
If you have trouble with the “add key” button being greyed out, this usually means that the password you entered in the beginning is not formatted correctly, try adding some numbers, capital letters etc. If it has been entered correctly then the screen should display a note to add an alias etc.

* Select a Custom Gradle Properties Template.

Navigate to Project Settings -> Player -> Android (tab) -> Publishing Settings -> Build



Navigate to the Assets\Plugins\Android\gradleTemplate.properties file, make sure to comment out or delete ‘android.enableR8’ and save the file:



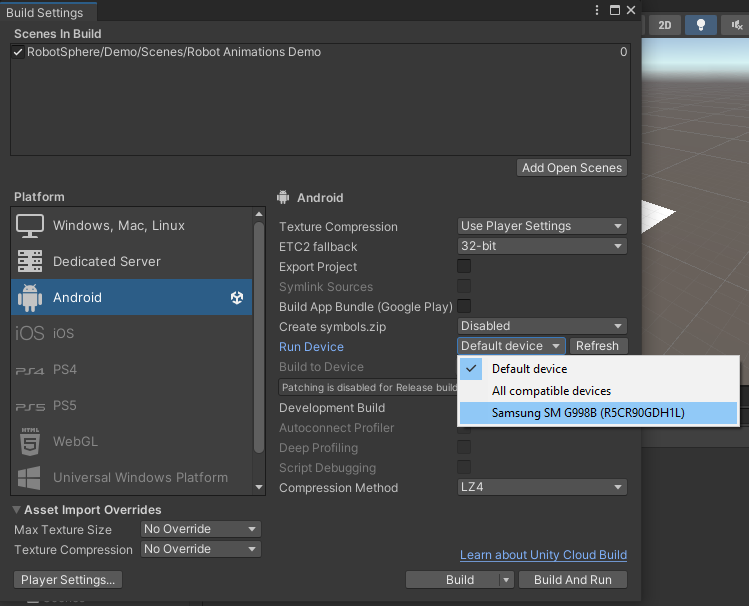
// = Commenting out this line

If ‘android.enableR8’ is not there, then you’re all good!

If you still have trouble building the game to your device, you can also export the project as android and build from Android Studio or VS Code, <https://docs.unity3d.com/Manual/android-export-process.html>

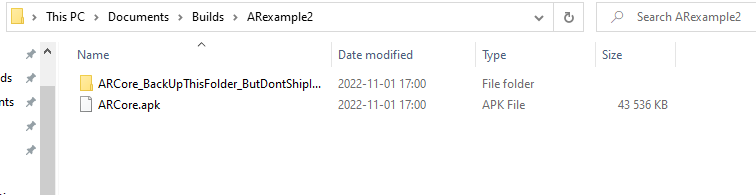
# Building to Device

Once your device is connected and you can see it in unity under build settings:



Then build using the selected device and clicking on Build and Run. You’ll need to save the APK file to a destination folder of your choice. Once built the app will automatically install and run on your device.

You can also just press Build and save the APK to a destination folder of your choosing.



Then drag and drop the ARCore.apk file (your filename.apk), onto your device’s internal storage (or external storage). On your device, find the file and click and open. Then make sure to accept “building to device from unknown sources” prompt. In android you might need to go to the main settings and find this option there and make sure it is turned on.

# iOS (untested since the 2020 course)

These instructions are from December 2020, I hope they work, otherwise follow Unity Documentation on Setup. I also recommend using the AR Core template when creating a project, this should include most of the setup needed.

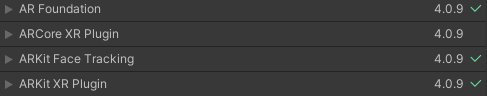
Warning: I am not a developer for iOS, and only have a small amount of experience with developing mobile apps to iOS.

First, you’re going to need ARKit on your system. This is included as part of the XCode 9 development system, so any version equal to or greater than that will be fine. You will need to set yourself up with an Apple Developer account to do this, which you can do here. There’s a bit of a delay so you need to do this sooner rather than later.

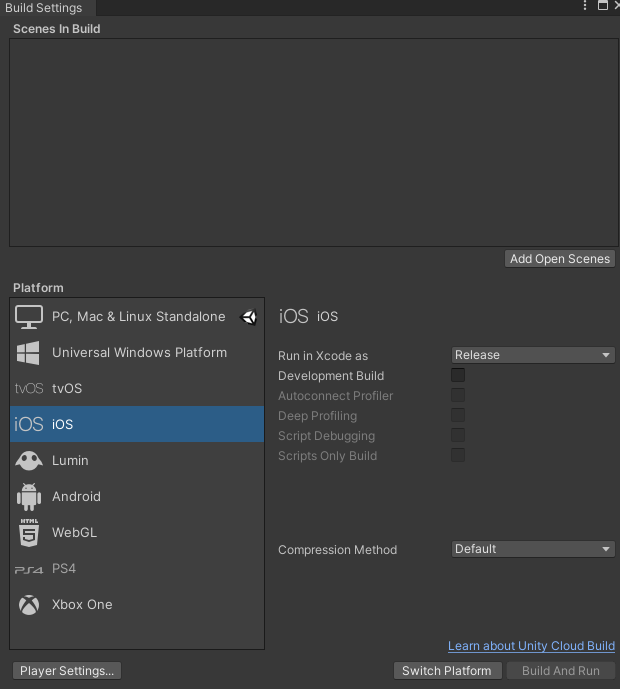
<https://developer.apple.com/programs/enroll/>

You can get XCode from the app store in the usual way.

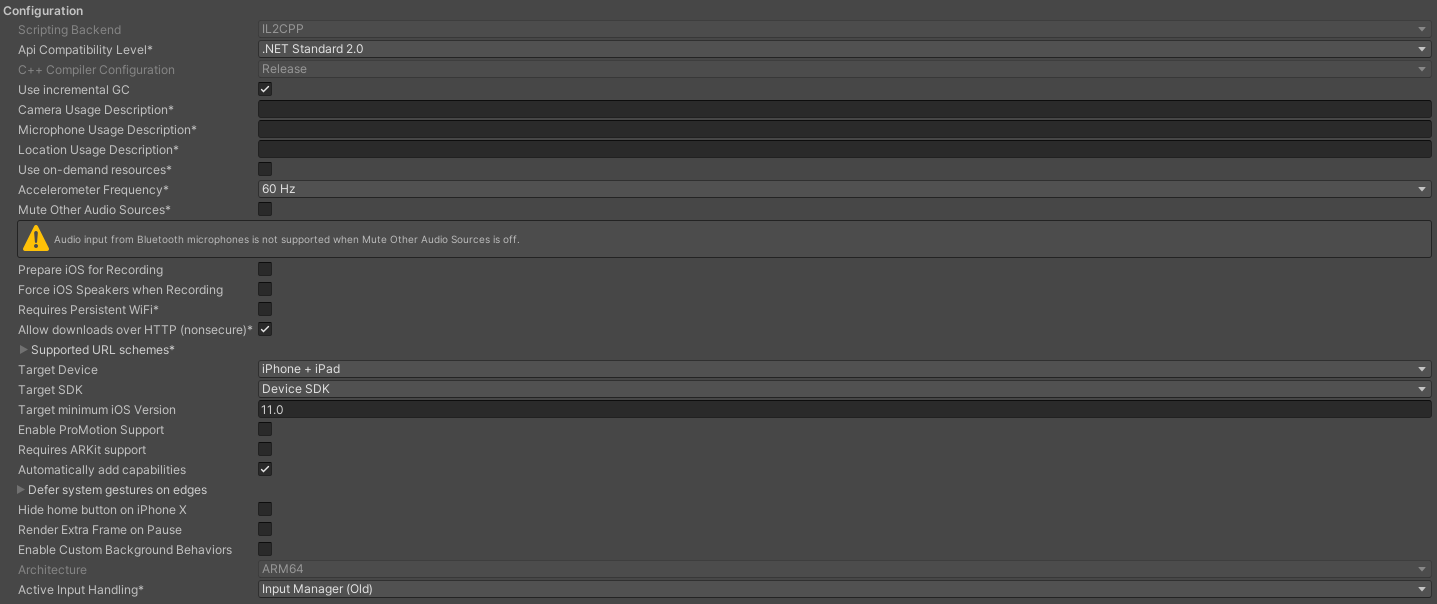
The packages you’ll need to install in Unity are AR Foundation, ARKit XR Plugin, and ARKit Face Tracking:



Having done that you’ll need to change your **build settings** so that you’re deploying to an iOS device:

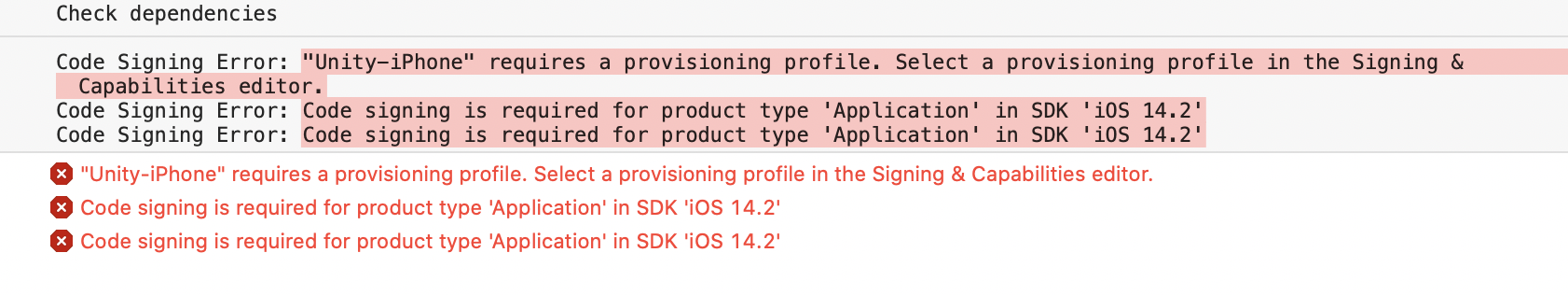


Go into your player architecture settings and change the architecture tro ARM64 if it isn’t set to that already (it probably will be):

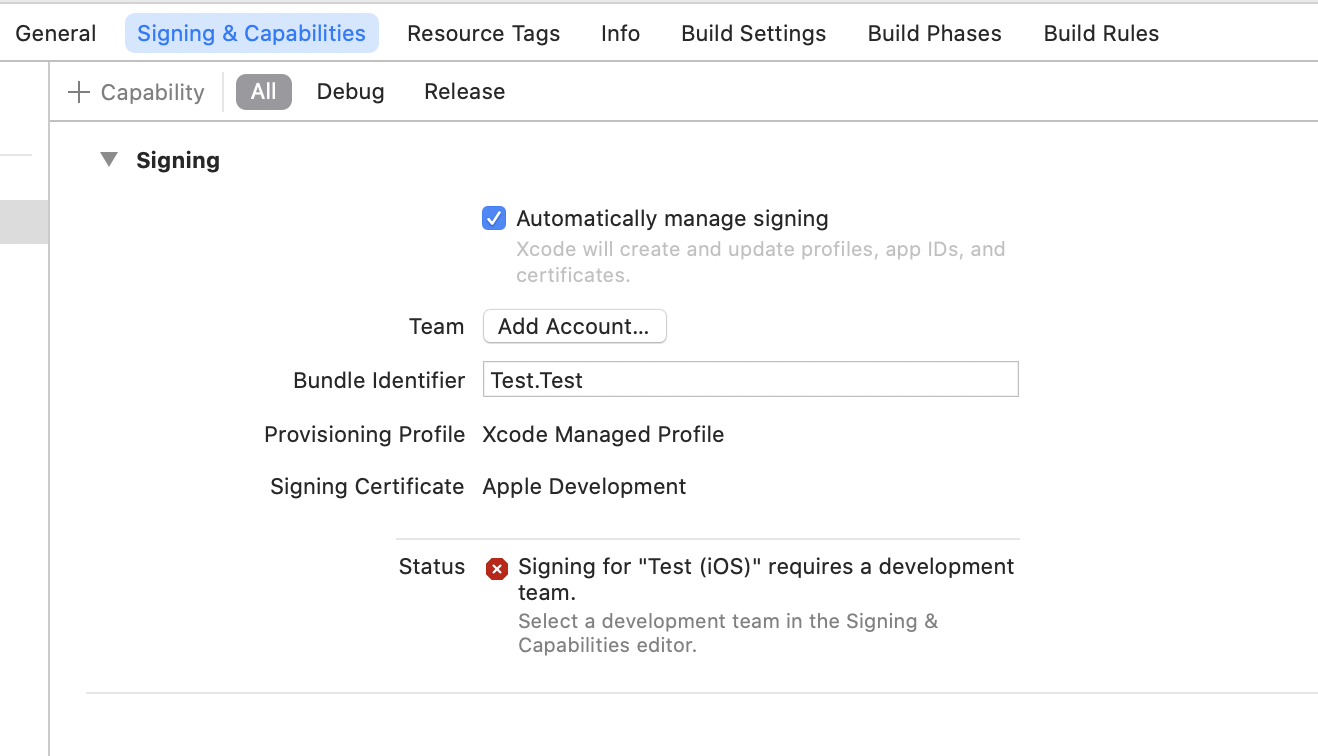


You’ll also probably want to change the product name and company name for your project because that is what automatic signing will use to create a name for the profile.

When you build and run an iOS mobile app in Unity it’ll create a new XCode project, start the building process through XCode and then copy the executable to your handheld device. For building, your phone needs to be connected.



When you do a first build and run the project will fail at the XCode end. At the top of the navigation bar for your project you’ll see a tab that says ‘Signing & Capabilities’ – this is where you can set the details for **provisioning**. This determines if this is a development only app, or something you plan for the App Store. Automatic signing should handle everything provided you are linked from XCode to your developer account. You’ll add yourself as the team, choose a unique identifier, and automatic signing should do the rest.



Finally…

You will need to tell your phone it can trust your computer, and also that your phone can run untrusted applications you deploy to it. to setting -> general -> device management and trust yourself. Assuming you do. If you don’t, that’s outside the scope of this document.

|  |  |
| --- | --- |
|  |  |

That should be it – your app should deploy on the phone, you should be able to run it, and everything **should** be fine.

# Help it doesn’t work

Oh no!  
  
My expectation of the first class of this module is that I’ll consider it a win if people can get the simplest kind of AR apps installed on their devices, so some time is set aside for troubleshooting in the classes that follow the lecture. However, please do email me if there are problems that I can address ahead of time. Let me know your computer model, handset model, where you got to in the instructions and what error you are getting.

It’s worth thinking of this guide as a living document – it will be updated in line with queries and problems. All I can really say is ‘This worked for my devices’ but you’ll have different computers, different configurations, and different handsets.

As with above, drop me a message at [natashab@chalmers.se](mailto:natashab@chalmers.se) if you can’t get this working.