The following guide has 5 steps that detail how to install all the necessary software to work with the front end. This pdf doucument has bookmarks and links which can be used to navigate the steps.

Step 1) Make sure you have an appropriate Apple device. ARKit only supports M9 processors and above. This includes the iPhone 6s, SE, iPad 2017 and more. Next, check if you are able to attain the required software below.

System Requirements

3/21/2018 • 5 min to read • Edit Online

Pre-requisites for using Xamarin

Xamarin products rely upon the platform SDKs from Apple and Google to target iOS or Android, so our system requirements match theirs. This page outlines system compatibility for the Xamarin platform and recommended development environment and SDK versions.

- Development Environments
- macOS Requirements
- Windows Requirements

Visit the installation instructions for more information on obtaining the software and required SDKs.

Development Environments

This table shows which platforms can be built with different development tool & operating system combinations:

	MACOS	WINDOWS
Development Environment	Visual Studio for Mac	Visual Studio
Xamarin.iOS	Yes	Yes (with Mac computer)
Xamarin.Forms	iOS & Android only (macOS in preview)	Android, Windows/UWP (iOS with Mac computer)
Xamarin.Mac	Yes	Open project & compile only

NOTE

To develop for iOS on Windows computers there must be a Mac computer accessible on the network, for remote compilation and debugging. This also works if you have Visual Studio running inside a Windows VM on a Mac computer.

macOS Requirements

Using a Mac computer for Xamarin development requires the following software/SDK versions. Check your operating system version and follow the instructions for the Xamarin installer.

	RECOMMENDED	NOTES
Operating System	macOS Sierra	The minimum required version is macOS Sierra (10.12)
Xamarin.iOS	iOS 11 SDK	The iOS that ships with Xcode 9

	RECOMMENDED	NOTES
Xamarin.Forms		• Xamarin.Forms apps built on macOS can include iOS, Android, and macOS projects, subject to the SDK requirements above. • Xamarin.Forms projects for Windows/UWP cannot be built on macOS.
Xamarin.Mac	OS X El Capitan (10.11) SDK	The OS X El Capitan SDK ships with Xcode 7.2; macOS SDKs ship with Xcode 8.

NOTE

Xcode can be installed (and updated) on developer.apple.com or via the Mac App Store.

Testing & Debugging on macOS

Xamarin mobile applications can be deployed to physical devices via USB for testing and debugging (Xamarin.Mac apps can be tested directly on the development computer; Apple Watch apps are deployed first to the paired iPhone).

	TESTING NOTES
Xamarin.iOS	• The easiest way to get started is using the iPhone, iPad, Apple Watch, and Apple TV simulators that are included with Xcode. • To use a device for testing, follow these instructions.
Xamarin.Forms	Xamarin.Forms apps for iOS and Android can be deployed to the relevant platforms as described above.
Xamarin.Mac	Xamarin.Mac apps can be tested directly on the development computer.

Windows Requirements

Using a Windows computer for Xamarin development requires the following software/SDK versions. Check your operating system version (and confirm that you are not using an *Express* version of Visual Studio - if so, consider updating to a *Community* edition). Visual Studio 2015 and 2017 installers include an option to install Xamarin automatically.

	RECOMMENDED	NOTES
Operating System	Windows 10	The minimum operating system version is Windows 7. Xamarin.Forms Windows support requires Windows 8.1, and Xamarin.Forms UWP support requires Windows 10.
Xamarin.iOS	iOS 10 SDK (installed on a Mac)	To build iOS projects on Windows requires: • Visual Studio 2015 or newer, and • a Mac computer, network-accessible from the Windows computer, that conforms to the minimum requirements for running Xamarin on macOS.
Xamarin.Forms		• Xamarin.Forms apps for iOS and Android can be deployed to the relevant platforms as described above. • Using Visual Studio also means you can test apps for Windows and the Universal Windows Platform (on Windows 10) using Microsoft's emulators. Windows apps can be tested directly on the development computer.
Xamarin.Mac		Xamarin.Mac projects (macOS desktop apps) can be opened in Visual Studio and compiled to check for errors, but Mac applications cannot currently be built for distribution in Visual Studio. See the release notes on Xamarin.Mac support for more information about the limitations of Mac projects in Visual Studio.

NOTE

- Xamarin for Visual Studio supports any Visual Studio 2015 or 2017 (Community, Professional, and Enterprise).
- To develop Xamarin.Forms apps for the Universal Windows Platform (UWP) requires Windows 10 with Visual Studio 2015 or 2017.

Testing & Debugging on Windows

Xamarin mobile applications can be deployed to physical devices via USB for testing and debugging (iOS devices must be connected to the Mac computer, not the computer running Visual Studio).

	TESTING NOTES
Xamarin.iOS	• The easiest way to get started is using the iPhone, iPad, Apple Watch, and Apple TV simulators that are included with Xcode. The simulators can be accessed on the connected Mac while debugging with Visual Studio. • To use a device for testing, follow these instructions (performing most steps on the connected Mac computer).
Xamarin.Forms	Xamarin.Forms apps can be deployed to the relevant devices and emulators as described above. The iOS app can only be tested via the connected Mac hardware; and the Windows tablet/desktop apps for Windows 8.1 or UWP can be tested directly on the development computer.

Step 2) If you are using Windows, install Visual Studio 2017 with Xamarin, as detailed below.

Installing Xamarin in Visual Studio on Windows

3/21/2018 • 2 min to read • Edit Online

Xamarin is free to use and included in all editions of Visual Studio.

Requirements

The following are required for installing Visual Studio tools for Xamarin:

1. Windows 7 or higher.
2. Visual Studio 2017 (Community, Professional, or Enterprise).
3. Xamarin for Visual Studio.

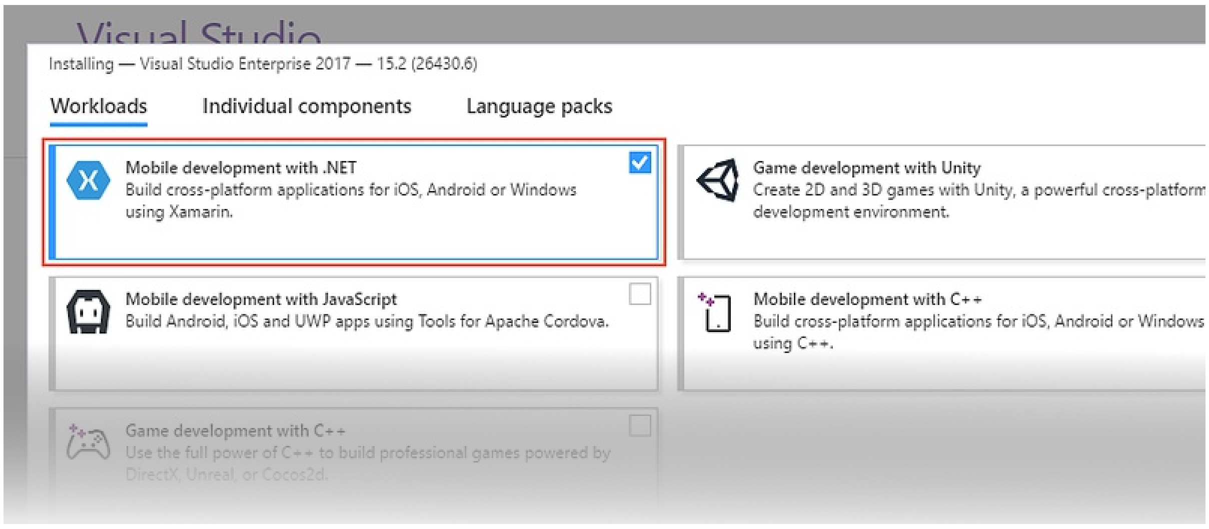
Note that Xamarin cannot be used with Express editions of Visual Studio due to lack of plug-in support.

For more information about the pre-requisites for installing and using Xamarin, see System Requirements.

Installation

Xamarin can be installed as part of a new Visual Studio installation. To achieve this, use the following steps:

1. Download Visual Studio Community, Visual Studio Professional, or Visual Studio Enterprise from the Visual Studio page (download links are provided at the bottom).
2. Double-click the downloaded package to start installation.
3. Select the **Mobile development with .NET** workload from the installation screen:



Installing — Visual Studio Enterprise 2017 — 15.2 (26430.6)

Mobile development with .NET ✓
Build cross-platform applications for iOS, Android or Windows using Xamarin.

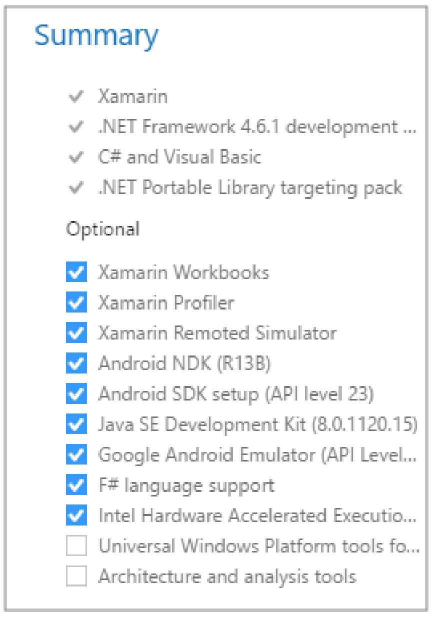
Game development with Unity
Create 2D and 3D games with Unity, a powerful cross-platform development environment.

Mobile development with JavaScript
Build Android, iOS and UWP apps using Tools for Apache Cordova.

Mobile development with C++
Build cross-platform applications for iOS, Android or Windows using C++.

Game development with C++
Use the full power of C++ to build professional games powered by DirectX, Unreal, or Cocos2d.

4. While **Mobile development with .NET** is selected, have a look at the **Summary** panel on the right. Here, you can deselect mobile development options that you do not want to install. By default, all options shown in the following screenshot are installed (**Xamarin Workbooks**, **Xamarin Profiler**, **Xamarin Remoted Simulator**, **Android NDK**, **Android SDK**, **Java SE Development Kit**, **Google Android Emulator**, **F# support**, and **Intel HAXM**):

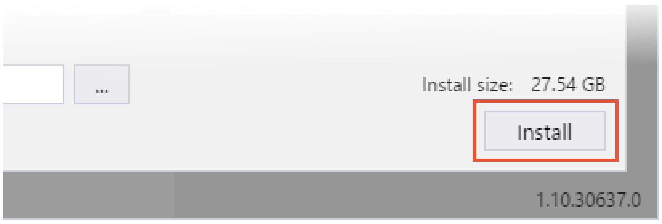

Summary

- ✓ Xamarin
- ✓ .NET Framework 4.6.1 development ...
- ✓ C# and Visual Basic
- ✓ .NET Portable Library targeting pack

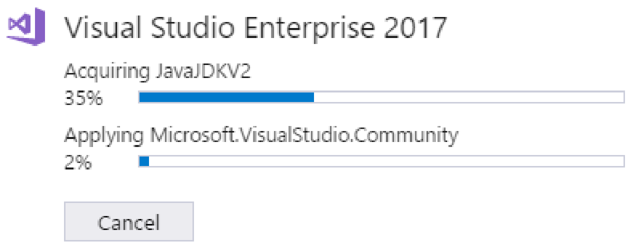
Optional

- ✓ Xamarin Workbooks
- ✓ Xamarin Profiler
- ✓ Xamarin Remoted Simulator
- ✓ Android NDK (R13B)
- ✓ Android SDK setup (API level 23)
- ✓ Java SE Development Kit (8.0.1120.15)
- ✓ Google Android Emulator (API Level...
- ✓ F# language support
- ✓ Intel Hardware Accelerated Executio...
- ☐ Universal Windows Platform tools fo...
- ☐ Architecture and analysis tools

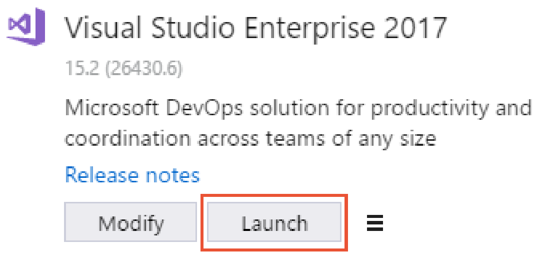
5. When you are ready to begin Visual Studio installation, click the **Install** button in the lower right-hand corner:

   
   Install size: 27.54 GB

   Depending on which edition of Visual Studio you are installing, the installation process can take a long time to complete. You can use the progress bars to monitor the installation:

   
   Visual Studio Enterprise 2017
   Acquiring JavaJDKV2
   35%
   Applying Microsoft.VisualStudio.Community
   2%

6. When Visual Studio installation has completed, click the **Launch** button to start Visual Studio:

   
   Visual Studio Enterprise 2017
   15.2 (26430.6)
   Microsoft DevOps solution for productivity and coordination across teams of any size

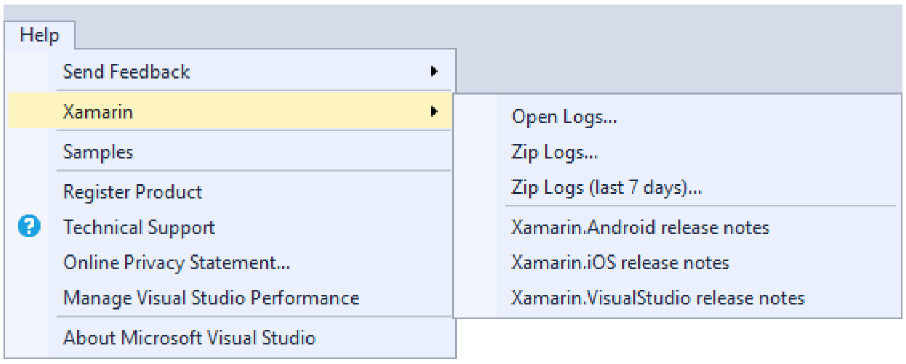
Adding Xamarin to Visual Studio 2017

If Visual Studio 2017 is already installed, you can add Xamarin by re-rerunning the Visual Studio installer to modify workloads (see Modify Visual Studio for details). Next, follow the steps listed above to install Xamarin.

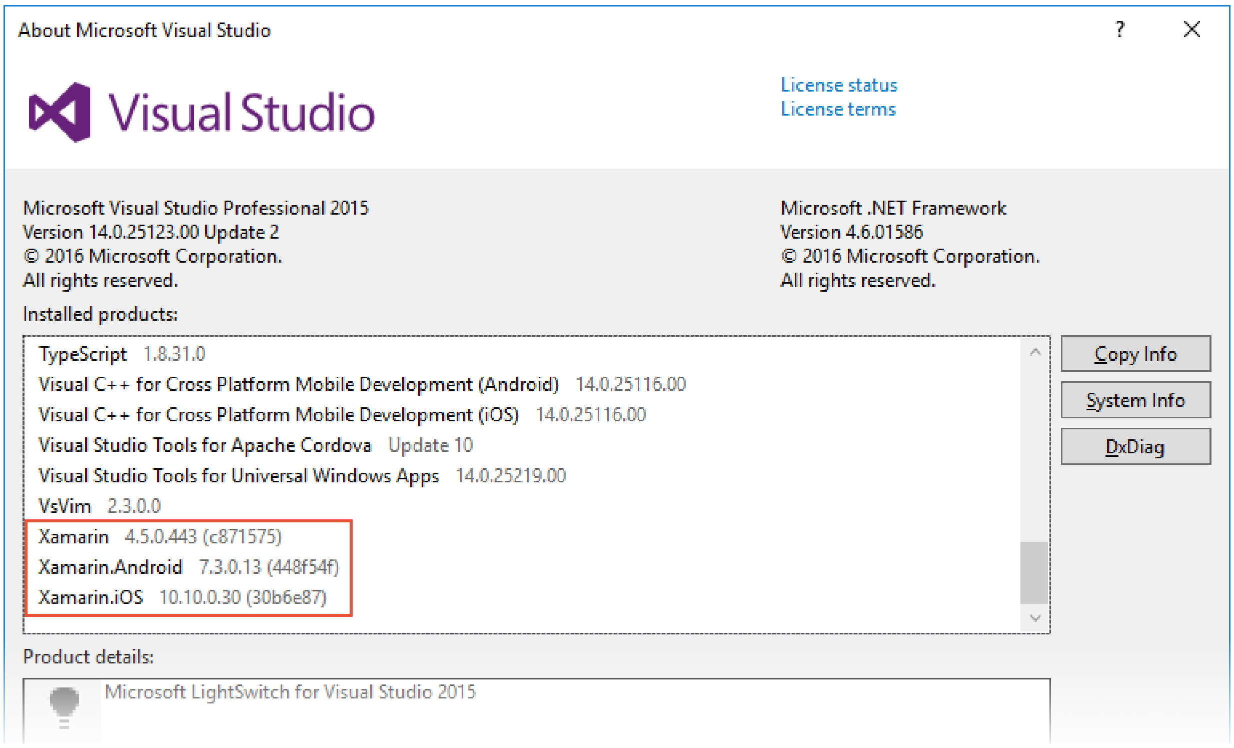
For more information about downloading and installing Visual Studio 2017, see Install Visual Studio 2017.

Verifying Installation

In Visual Studio 2017, you can verify that Xamarin is installed by clicking the **Help** menu. If Xamarin is installed, you should see a **Xamarin** menu item as shown in this screenshot:


Help
Send Feedback
Xamarin
Samples
Register Product
Technical Support
Online Privacy Statement...
Manage Visual Studio Performance
About Microsoft Visual Studio
Open Logs...
Zip Logs...
Zip Logs (last 7 days)...
Xamarin.Android release notes
Xamarin.iOS release notes
Xamarin.VisualStudio release notes

If you are using an earlier versions of Visual Studio, you can click **Help > About Microsoft Visual Studio** and scroll through the list of installed products to see if Xamarin is installed:



Microsoft Visual Studio Professional 2015
Version 14.0.25123.00 Update 2
© 2016 Microsoft Corporation.
All rights reserved.

Microsoft .NET Framework
Version 4.6.01586
© 2016 Microsoft Corporation.
All rights reserved.

Installed products:

```
TypeScript   1.8.31.0
Visual C++ for Cross Platform Mobile Development (Android)   14.0.25116.00
Visual C++ for Cross Platform Mobile Development (iOS)   14.0.25116.00
Visual Studio Tools for Apache Cordova   Update 10
Visual Studio Tools for Universal Windows Apps   14.0.25219.00
VsVim   2.3.0.0
Xamarin   4.5.0.443 (c871575)
Xamarin.Android   7.3.0.13 (448f54f)
Xamarin.iOS   10.10.0.30 (30b6e87)
```

Product details:

Microsoft LightSwitch for Visual Studio 2015

For more information about locating version information, see Where can I find my version information and logs?

Next Steps

Installing Visual Studio Tools for Xamarin allows you to start writing code for your apps, but does require additional setup for building and deploying your apps to simulator, emulator, and device. Visit the following guides to complete your installation and start building cross platform apps.

iOS

For more detailed information, see the Installing Xamarin.iOS on Windows guide.

1. Install Xamarin.iOS tools on your Mac
2. Configuring your Mac
3. iOS Developer Setup (To run your application on device).
4. Connecting Visual Studio to your Mac build host
5. Remoted iOS Simulator
6. Introduction to Xamarin.iOS for Visual Studio

Android

Xamarin Firewall Configuration Instructions

3/21/2018 • 1 min to read • Edit Online

A list of hosts that you need to whitelist in your firewall to allow Xamarin's platform to work for your company.

In order for Xamarin products to install and work properly, certain endpoints must be accessible to download the required tools and updates for your software. If you or your company have strict firewall settings, you may experience issues with installation, licensing, components, and more. This document outlines some of the known endpoints that need to be whitelisted in your firewall in order for Xamarin to work. This list does not include the endpoints required for any third-party tools included in the download. If you are still experiencing trouble after going through this list, refer to the Apple or Android installation troubleshooting guides.

Endpoints to Whitelist

Xamarin Installer

The following known addresses will need to be added in order for the software to install properly when using the latest release of the Xamarin installer:

- xamarin.com (installer manifests)
- dl.xamarin.com (Package download location)
- dl.google.com (to download the Android SDK)
- download.oracle.com (JDK)
- visualstudio.com (Setup packages download location)
- go.microsoft.com (Setup URL resolution)
- aka.ms (Setup URL resolution)

If you are using a Mac and are encountering Xamarin.Android install issues, please ensure that macOS is able to download Java.

Components Store and NuGet (including Xamarin.Forms)

The following addresses will need to be added to access the Xamarin Component Store or NuGet (Xamarin.Forms is packaged as a NuGet):

- components.xamarin.com (to use Xamarin Components Store)
- xampubdl.blob.core.windows.net (hosts Components Store downloads)
- www.nuget.org (to access NuGet)
- az320820.vo.msecnd.net (NuGet downloads)
- dl-ssl.google.com (Google components)

Software Updates

The following addresses will need to be added to ensure that software updates can download properly:

- software.xamarin.com (updater service)
- download.visualstudio.microsoft.com
- dl.xamarin.com

Xamarin Insights

The following addresses will need to be added to ensure that activity reaches the Xamarin Insights server:

- https://xaapi.xamarin.com

Xamarin Mac Agent

To connect Visual Studio to a Mac build host using the Xamarin Mac Agent requires the SSH port to be open. By default this is **Port 22**.

Summary

This guide covered the endpoints to whitelist to allow Xamarin products to install and update properly on your machine.

Step 3) Install Visual Studio 2017 with Xamarin for Mac, and Xcode as detailed below

Installing Xamarin.iOS on Windows - Xamarin

bradumbaugh

- Docs
- Xamarin
- Xamarin.iOS
- Getting Started
- Setup and Installation
- Windows

- 09/29/2017
- 7 minutes to read
- Contributors

In this article

This article shows how to set up Xamarin.iOS for Visual Studio. It covers the installation process for the Xamarin extension for Visual Studio, and discusses connecting to the Apple SDK installed on the Mac.

Overview

Xamarin.iOS for Visual Studio allows iOS applications to be written and tested on Windows computers, with a networked Mac providing the build and deployment service.

Developing for iOS inside Visual Studio provides the ability to:

- Create cross-platform solutions for iOS, Android, and Windows applications.
- Use Visual Studio tools (such as *Resharper* and *Team Foundation Server*) for all your cross-platform projects, including iOS source code.
- Work with a familiar IDE, while taking advantage of Xamarin.iOS bindings of all Apple's APIs

Xamarin.iOS for Visual Studio supports configurations where Visual Studio is running inside a Windows virtual machine on a Mac (using Parallels or VMWare), or when it is on a separate machine that is visible on the same network as a Mac. Regardless of which configuration works best for you, Visual Studio will connect to the Mac promptly and securely using SSH.

This article covers the steps to install and configure the Xamarin.iOS tools on both the Mac and Windows machine, as well as the steps to connect to the Mac host so that developers can build, debug, and deploy Xamarin.iOS applications using Visual Studio.

The diagram below shows a simple overview of the Xamarin.iOS development workflow:

The Xamarin.iOS development workflow

Important

Visual Studio actually launches a separate MSBuild process to build the projects. This process creates a new connection to the Mac, meaning there are actually two SSH connections from Windows to Mac when Visual Studio builds. Building from the command-line only creates the one MSBuild process. For the simplicity of this diagram, all the connections are simply represented by one arrow.

Requirements

Xamarin.iOS for Visual Studio accomplishes an amazing feat: it lets developers create, build, and debug iOS applications on a Windows computer using the Visual Studio IDE. It cannot do this alone – iOS applications cannot be created without Apple's compiler, and they cannot be deployed without Apple's certificates and code-signing tools. This means that a Xamarin.iOS for Visual Studio installation requires a connection to a networked Mac OS X computer to perform these tasks. Once configured, Xamarin's tools will make the process as seamless as possible.

System Requirements

The system requirements are:

Windows

1. Windows 7 or higher.
2. Visual Studio 2015 Professional or higher

   a. If you have an Enterprise license, you will need to install Visual Studio Enterprise.

3. Xamarin for Visual Studio.

The Xamarin tools cannot be used with Express editions of Visual Studio due to lack of plug-in support. Xamarin is supported in Visual Studio Community.

Mac

1. A Mac running macOS Sierra (10.12) or higher (although the latest stable version is recommended).
2. Xamarin iOS SDK. This is installed when downloading Visual Studio for Mac
3. Apple's Xcode IDE and iOS SDK (The latest stable version from the Mac App Store is recommended).

The Windows computer must be able to reach the Mac via the network.

Apple Developer Account

To deploy applications to a device or to submit them to the App Store, an Apple Developer account is required. The relevant developer certificates and provisioning profiles must be created and installed on the networked Mac before Xamarin.iOS for Visual Studio can work. See the Device Provisioning article for steps to obtain a development certificate and to provision a device.

Features

Xamarin.iOS for Visual Studio enables the creation, editing, building, and deployment of Xamarin.iOS projects from Windows. This includes the following features:

- Create new iOS projects.
- Edit iOS projects and cross-platform solutions that also include Xamarin.Android and UWP projects.
- Compile iOS projects and cross-platform solutions that also include Xamarin.Android and UWP projects.
- Storyboard and .xib support using the iOS Designer.
- Deploy and debug iOS applications, where the app itself runs in a simulator, or on a device connected to the Mac.
- iOS simulator on Windows – For more information on using the iOS simulator on Windows, refer to this guide.

Configuring your Mac

Installation

To install Xamarin.iOS tools on your mac host you must install Visual Studio for Mac.

Once the software is installed, follow the steps in the next sections to configure Xamarin.iOS on macOS to allow Xamarin for Visual Studio to connect to it.

Important

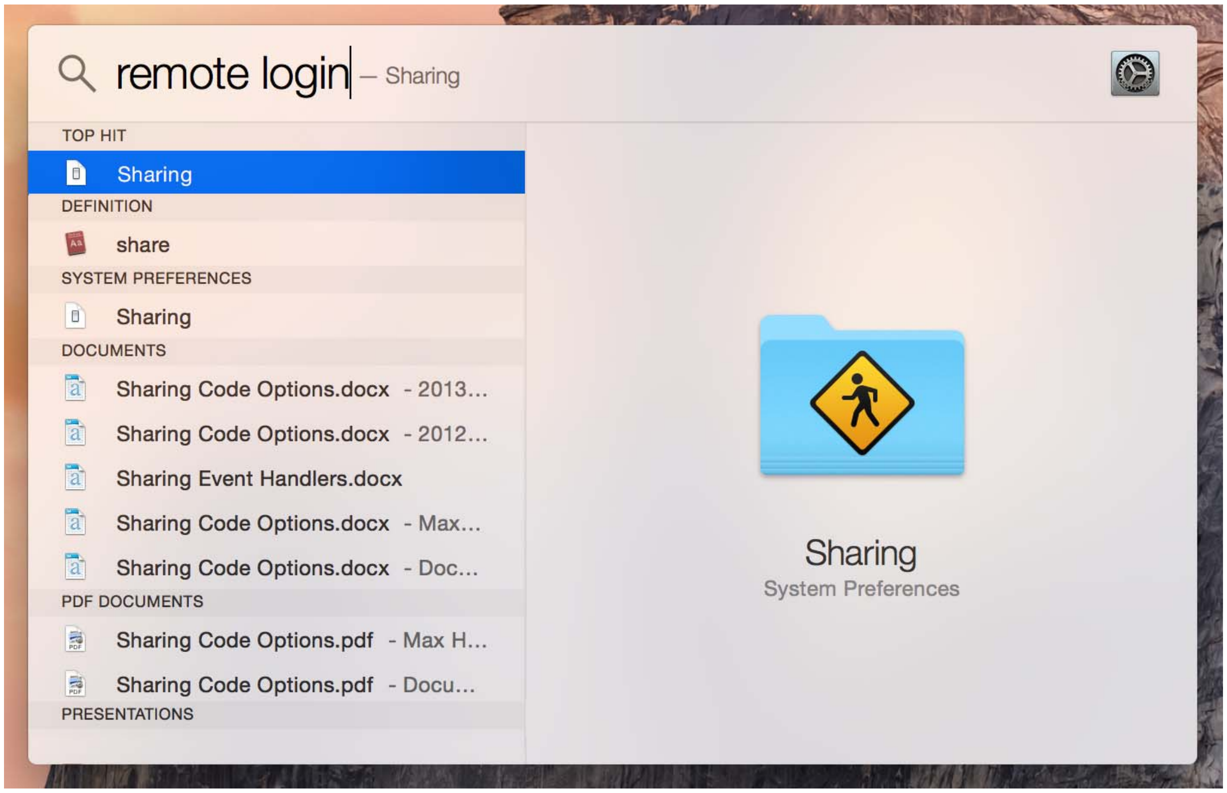
The Windows machine must be using the same version of Xamarin.iOS as the Mac to which it is connected. To ensure this is true:

- **Visual Studio 2015 and earlier**: Ensure that you are on the same updates channel as Visual Studio for Mac.
- **Visual Studio 2017, Release Version**: Ensure that you are on the **Stable** channel of Visual Studio for Mac.
- **Visual Studio 2017, Preview Version**: Ensure that you are on the **Alpha** channel of Visual Studio for Mac.

Configuration

To access communication between the Xamarin extension for Visual Studio and your Mac, you will need to allow **Remote Login** on your Mac. Follow the steps below to set this up:

1. Open *Spotlight* (**Cmd-Space**) and search for **Remote Login** and then select the **Sharing** result. This will open the **System Preferences** at the **Sharing** panel.


remote login – Sharing
TOP HIT
Sharing
DEFINITION
share
SYSTEM PREFERENCES
Sharing
DOCUMENTS
Sharing Code Options.docx - 2013...
Sharing Code Options.docx - 2012...
Sharing Event Handlers.docx
Sharing Code Options.docx - Max...
Sharing Code Options.docx - Doc...
PDF DOCUMENTS
Sharing Code Options.pdf - Max H...
Sharing Code Options.pdf - Docu...
PRESENTATIONS
Sharing
System Preferences

2. Tick the **Remote Login** option in the **Service** list on the left in order to allow Xamarin for Visual Studio to connect to the Mac.

Tick the Remote Login option in the Service list

3. Make sure that **Remote Login** is set to allow access for **All users**, or that your Mac username or group is included in the list of allowed users in the list on the right.

The Mac should now be discoverable by Visual Studio if it's on the same network.

Note

If you have the macOS firewall set to block signed applications by default, you may need to allow `mono-sgen` to receive incoming connections. An alert dialog will appear to prompt you if this is the case.

iOS Developer Setup

For iOS development, it is important that the Mac machine is configured with the relevant signing identities. This allows you to correctly sign your apps so that they can be distributed either via the App Store or Ad Hoc. Follow the link below for instructions on setting up a Mac for iOS development with Xamarin:

- Device Provisioning

Once your Mac is configured, it's time to set up your Windows computer.

Windows Installation

Xamarin can be installed as part of your Visual Studio 2017 or 2015 installation. To install Visual Studio tools for Xamarin, see the Windows Installation guide.

Installation Complete

After the installation process is complete, there are still a few more steps required to get everything working:

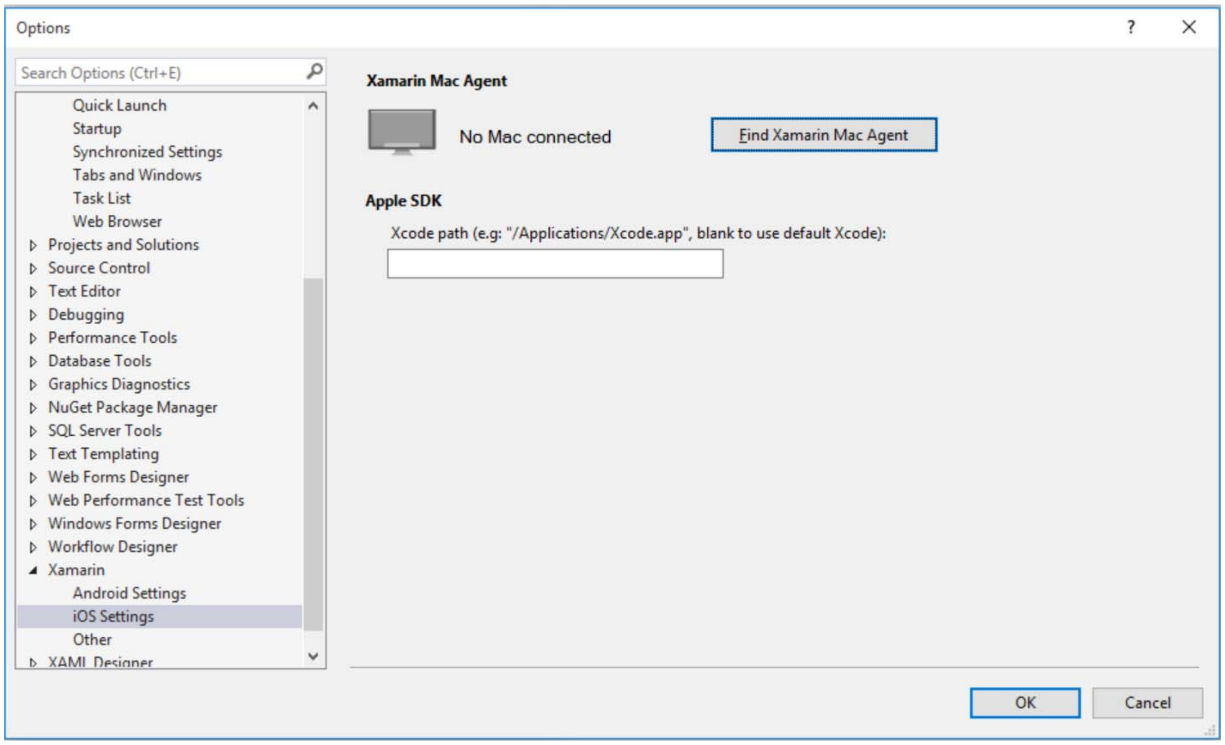
- Connect Visual Studio to the Mac – Visual Studio must be connected to the Mac build host before it can build Xamarin.iOS projects.
- Configure the Visual Studio Toolbar – This will let you easily access Xamarin.iOS features in Visual Studio.

Connecting to the Mac

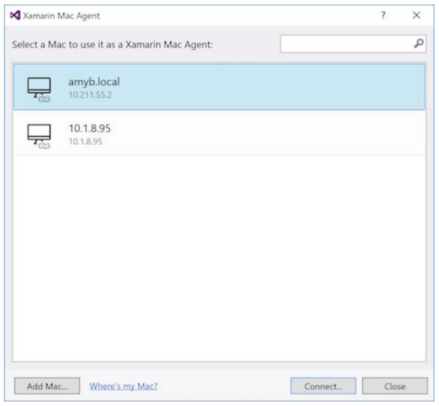
A connection is made from Xamarin.iOS for Visual Studio to your Mac build host via an SSH connection between machines. For more information on the connection, refer to the Connecting to Mac guide.

To connect your Mac, follow the steps below:

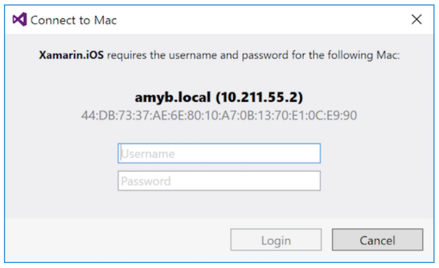
- Browse to **Tools > Options** and under **Xamarin** select **iOS Settings**:



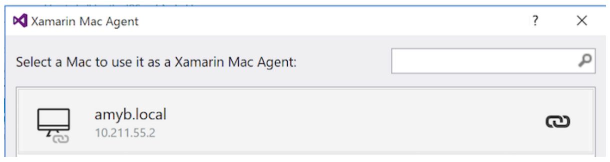
- Provided the Mac has been correctly configured to allow **Remote Login**, you should be able to select your Mac in the list:



- This will prompt for the administrative credentials of your Mac host:


Connect to Mac
Xamarin.iOS requires the username and password for the following Mac:
amyb.local (10.211.55.2)
44:DB:73:37:AE:6E:80:10:A7:0B:13:70:E1:0C:E9:90
Username
Password
Login
Cancel

- When you have connected, it will display the 'Connection Successful' icon next to the machine name:

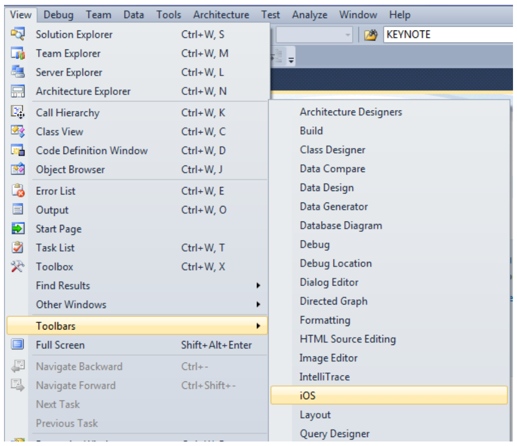


You will be reconnected each time you start Visual Studio.

When an iOS project is open the iOS Toolbar will be visible by default, and does not need to be configured.

The steps below can be used if the iOS toolbar does not appear.

To configure the toolbar first open the **View > Toolbars** menu and make sure the **iOS** entry is selected. Choose the menu item as shown in this screenshot—it should be ticked to indicate that the toolbar is visible:



Visual Studio 2015

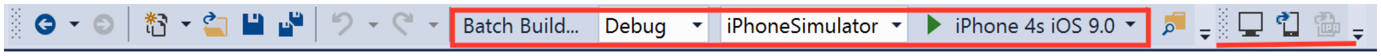
In versions earlier than Visual Studio 2017, the **Solution Platforms** button may need to be added to the Standard toolbar. This allows an iOS Device or the iOS Simulator to be selected when debugging. Follow the instructions below to set this up

Click the menu button at the right side of the Standard bar:

- Choose **Add or Remove Buttons**
- Select **Solution Platforms**

Select Solution Platform

The **Standard** and **iOS** toolbars should now resemble this screenshot:



Once the toolbar configuration is complete, you are ready to begin using Xamarin iOS for Visual Studio.

Summary

This article presented a step-by-step guide to installing, configuring, and using Xamarin iOS for Visual Studio.

It covered installing and configuring the prerequisite tools on Windows and Mac OS X.

- Installation
- Device Provisioning
- Introduction to Xamarin.iOS for Visual Studio
- Connecting a Mac to your Visual Studio environment with XMA (video)

Feedback

There is currently no feedback for this document.

Step 4) Read about provisioning below, and choose a way to provision your Apple device. I used free provisioning during development, but an Apple developer license would be more beneficial, if more people were developing and testing the application. The paid provisioning guides are directly below, and the free provisioning guide is the shown after those.

Device Provisioning - Xamarin | Microsoft Docs

asb3993

- Docs
- Xamarin
- Xamarin.iOS
- Getting Started
- Setup and Installation
- Set Up Device for Development

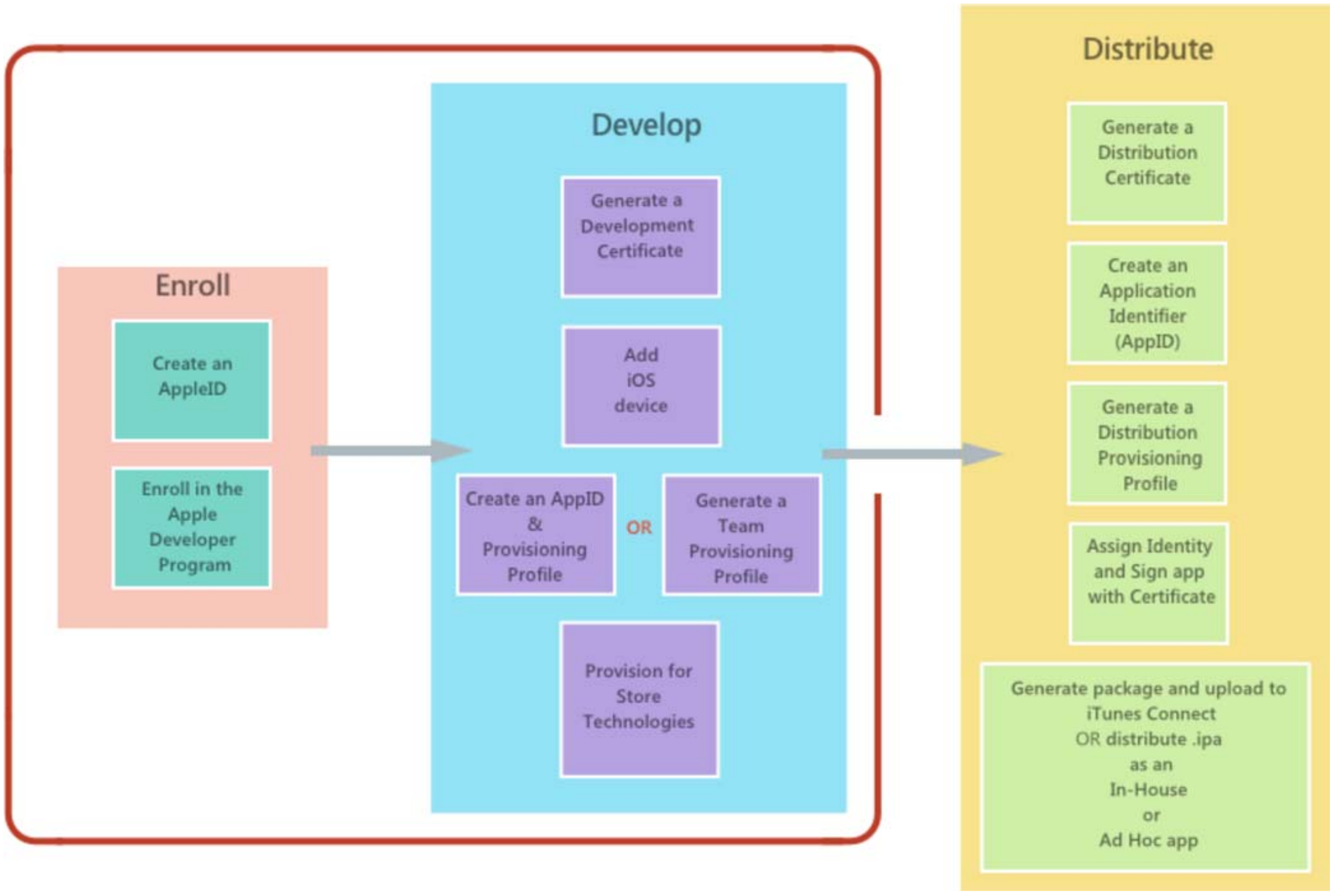
- 07/15/2017
- 4 minutes to read
- Contributors

In this article

Once Xamarin.iOS has been successfully installed, the next step in iOS development is to provision your iOS device. This guide will explore requesting development certificates and profiles, working with app services, and deploying an app to device.

While developing a Xamarin.iOS application it is essential to test it by deploying the app to a physical device, in addition to the simulator. Device-only bugs and performance issues can transpire when running on a device, due to hardware limits such as memory or network connectivity. To test on a physical device, the device must be *provisioned*, and Apple must be informed that the device will be used for testing.

The highlighted sections in the image below show the steps required to get set up for iOS provisioning:


Enroll
Create an AppleID
Enroll in the Apple Developer Program
Develop
Generate a Development Certificate
Add iOS device
Create an AppID & Provisioning Profile
OR
Generate a Team Provisioning Profile
Provision for Store Technologies
Distribute
Generate a Distribution Certificate
Create an Application Identifier (AppID)
Generate a Distribution Provisioning Profile
Assign Identity and Sign app with Certificate
Generate package and upload to iTunes Connect OR distribute .ipa as an In-House or Ad Hoc app

After this, the next step is to distribute the application. For more information on deployment, visit the App Distribution guides.

Before deploying the application to a device, you need to have an active subscription to Apple's Developer Program, *or* use Free Provisioning. Apple offers two program options:

- **Apple Developer Program** – Regardless of whether you are an individual or represent an organization, the Apple Developer Program allows you to develop, test, and distribute apps.
- **Apple Developer Enterprise Program** – The Enterprise program is most suited to organizations that want to develop and distribute apps in-house only. Members of the Enterprise program do not have access to iTunes Connect, and apps created cannot be published to the App Store.

To register for either of these programs, visit the Apple Developer Portal to register. Note that to register as an Apple developer, it is necessary to have an Apple ID. This guide has been created with the assumption that you **are** a member of an Apple Developer Program.

Alternatively, Apple introduced Free Provisioning in Xcode 7 which allows a single application to run on a single device *without* being a member of Apple's Developer Program. There are a number of limitations when provisioning in this way, as detailed here.

Any application that runs on a device needs to include a set of metadata (or *thumbprint*), which contains information about the application and the developer. Apple uses this thumbprint to make sure that the application is not tampered with when deploying to, or running on, a user's device. This is achieved by requiring app developers to register their Apple ID as a developer, and to setup an App ID, request a Certificate, and register the device on which the application will be deployed.

When deploying an application to a device, a Provisioning Profile is also installed on the iOS device. The Provisioning Profile exists to verify the information that the app was signed with at build time and is cryptographically signed by Apple. Together, the Provisioning Profile and 'thumbprint' checks determine if an application can be deployed to a device by checking:

- **Who** (Certificates – has the app been signed with a private key, which has a corresponding public key in the provisioning profile? The certificate also associates the developer with a development team)
- **What** (Individual App ID – Does the Bundle Identifier set in the Info.plist match the App ID in the provisioning profile?)
- **Where** (Devices – Is the device contained in the provisioning profile?)

These steps ensure that everything that is created or used during the development process, including the applications and devices, can be traced back to an Apple Developer account.

- Visual Studio for Mac
- Visual Studio

Provisioning your Device

There are two ways to provision your iOS device with Visual Studio for Mac:

- **Automatically (Recommended)** – Select the **Automatically manage signing** option in the Info.plist file to have Visual Studio for Mac automatically create and manage your Signing Identities, App IDs, and Provisioning Profiles. For information on how to automatically manage provisioning, see the Automatic Provisioning guide. This is the recommended way of provisioning an iOS device.

- **Manually** – Signing Identities, App IDs, and Provisioning Profiles can be created and managed via the Apple Developer Portal, as described in manual provisioning guide. These artifacts can then be managed as described in the Apple Account Management guide.

Provisioning for Application Services

Apple provides a selection of special Application Services, also called capabilities, that can be activated for a Xamarin.iOS application. These Application Services must be configured on both the iOS Provisioning Portal when the **App ID** is created and in the **Entitlements.plist** file that is part of the Xamarin.iOS application's project. For information on adding Application Services to your app, refer to the Introduction to Capabilities guide and the Working with Entitlements guide.

- Create an App ID with the required app services.
- Create a new provisioning profile that contains this App ID.
- Set Entitlements in the Xamarin.iOS Project

Note

Currently, provisioning profiles created in Visual Studio for Mac will not take into account entitlements selected in your projects (Entitlements.plist). This functionality will be added in future versions of the IDE. If you need to use App Services, it is recommended that you follow the instructions in the Manual Provisioning guide.

- Free Provisioning
- App Distribution
- Troubleshooting
- Apple - App Distribution Guide

Feedback

There is currently no feedback for this document.

The following is the free provisioning option. Pay close attention to how you have to match the bundle identities in both XCode and Visual Studio below. These license files expire once per week. To refresh, you should just launch your blank XCode project, to get a new license.

Free Provisioning - Xamarin | Microsoft Docs

asb3993

- Docs
- Xamarin
- Xamarin.iOS
- Getting Started
- Setup and Installation
- Set Up Device for Development
- Free Provisioning

- 03/19/2017
- 3 minutes to read
- Contributors

In this article

With Apple's release of Xcode 7 came an important change for all iOS and Mac developers–free provisioning.

Free provisioning allows developers to deploy their Xamarin.iOS application to their iOS device **without** being part of any **Apple Developer Program**. This is extremely advantageous to developers, as testing on a device allows many benefits over testing on the simulator, including, but not limited to memory, storage, network connectivity among others.

Provisioning without an Apple Developer account must be performed through Xcode, which creates a *Signing Identity* (containing a developer certificate and private key), and a *Provisioning Profile* (containing an explicit App ID and the UDID of your connected iOS device).

Requirements

To take advantage of deploying your Xamarin.iOS applications to a device with free provisioning you must be using Xcode 7 or above.

The Apple ID being used must not be connected to any Apple Developer Program.

The Bundle ID used in your app must be unique and cannot have been used in another app previously. Any Bundle ID used with free provisioning CAN NOT be re-used again. If you have already distributed an app, you cannot provision that app with free provisioning.

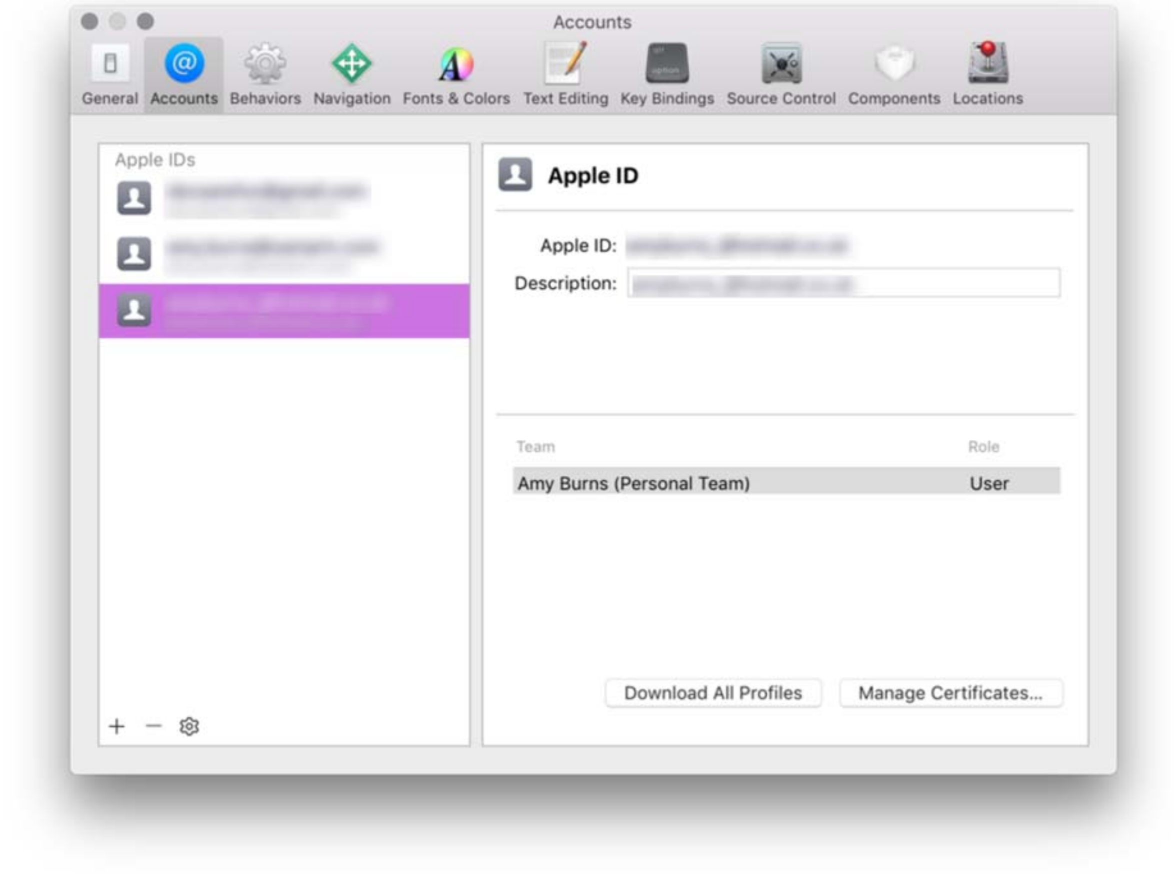
Refer to the App Distribution guides for more information.

If your app uses App Services, then you will need to create a provisioning profile as detailed in the device provisioning guide. You can see further limitations in the relevant section below.

Launching your App

To use free provisioning for deploying an application to a device you will use Xcode to create the signing identity and provisioning profiles, and will then use Visual Studio for Mac or Visual Studio choose the correct profile to sign our app with. Follow the step-by-step walkthrough below to do this:

1. If you do not have an Apple ID, create one at appleid.apple.com.
2. Open Xcode and browse to **Xcode > Preferences**.
3. Under **Accounts**, use the **+** button to add your existing Apple ID. It should look similar to the screenshot below:



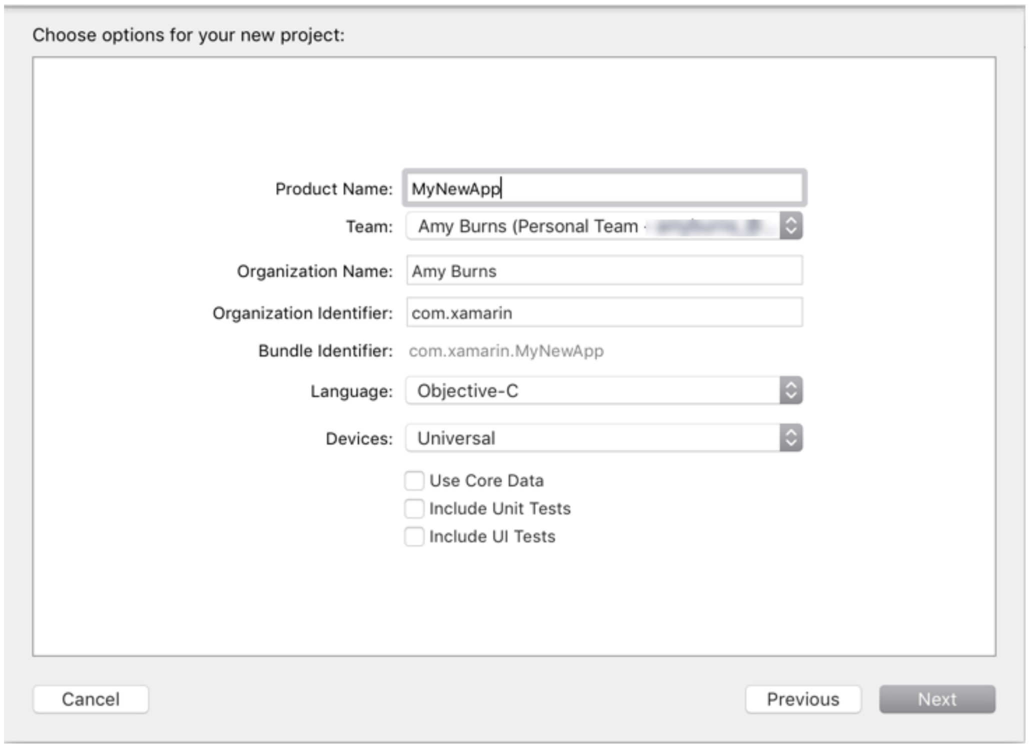
Apple ID

Apple ID: [redacted]

Description: [redacted]

Team	Role
Amy Burns (Personal Team)	User

4. Plug in the iOS device you wish to deploy to and create a new blank single-view iOS project in Xcode. Set the **Team** drop-down to the Apple ID that you have just added. It should be in a format similar to `your name (Personal Team - your Apple ID)`:

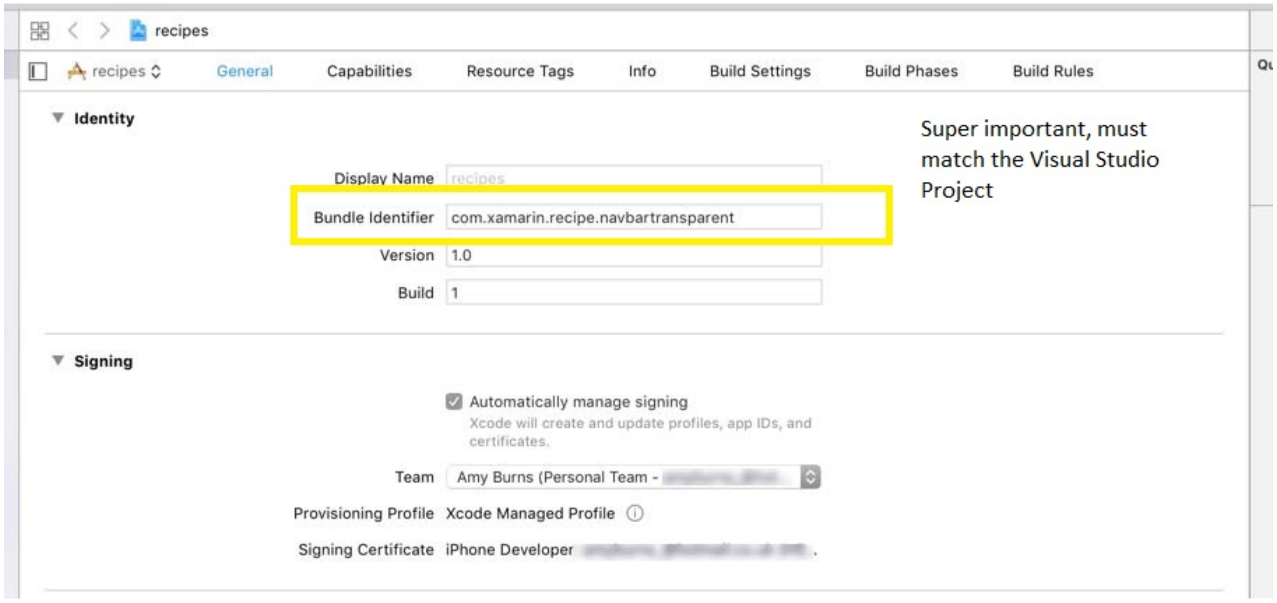


Choose options for your new project:

Product Name:	MyNewApp
Team:	Amy Burns (Personal Team - [redacted]
Organization Name:	Amy Burns
Organization Identifier:	com.xamarin
Bundle Identifier:	com.xamarin.MyNewApp
Language:	Objective-C
Devices:	Universal

- [ ] Use Core Data
- [ ] Include Unit Tests
- [ ] Include UI Tests

5. Under the **General > Identity** section, make sure that the Bundle Identifier matches *exactly* the Bundle Identifier of your Xamarin.iOS app and ensure the deployment target matches or is lower than your connected iOS device. This step is extremely important, as Xcode will only create a provisioning profile with an explicit App ID:



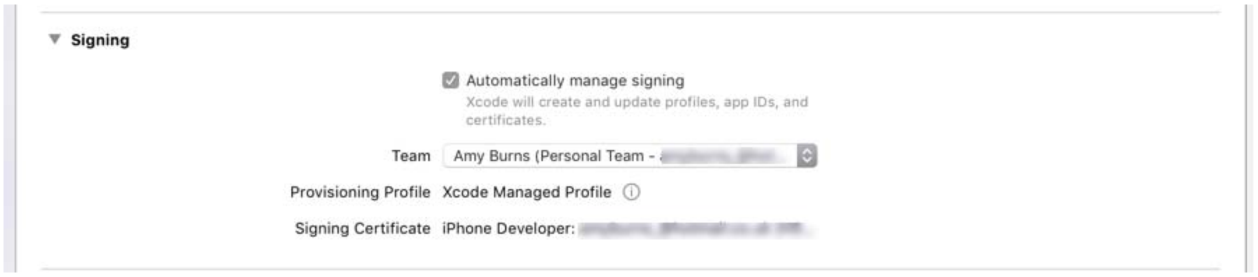
Identity

Display Name: recipes
Bundle Identifier: com.xamarin.recipe.navbartransparent
Version: 1.0
Build: 1

Signing

Automatically manage signing
Xcode will create and update profiles, app IDs, and certificates.
Team: Amy Burns (Personal Team -
Provisioning Profile: Xcode Managed Profile
Signing Certificate: iPhone Developer

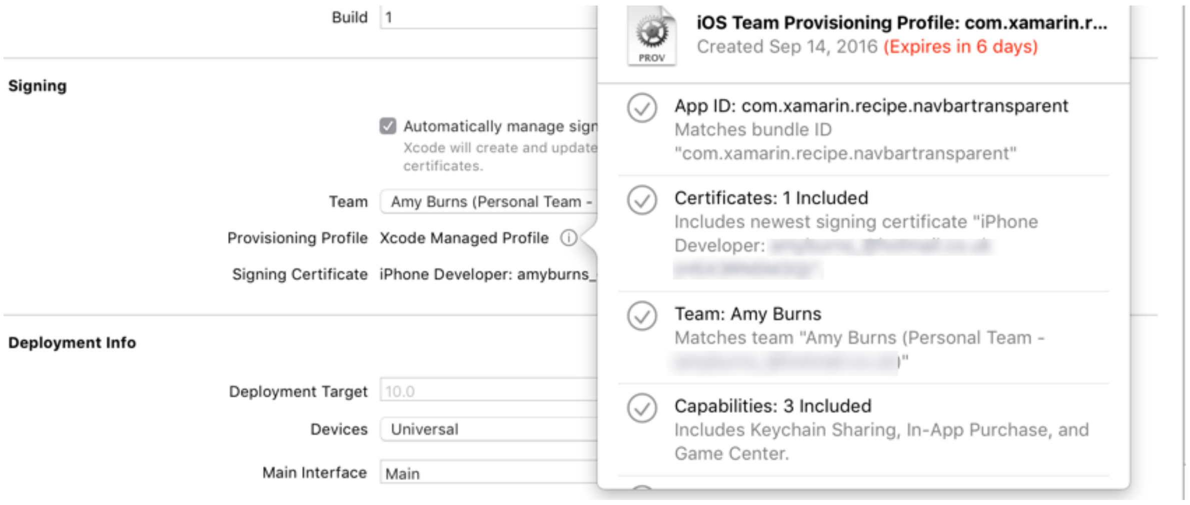
6. In the Signing section, select **Automatically Manage Signing** and select your team from the drop down list:



Signing

Automatically manage signing
Xcode will create and update profiles, app IDs, and certificates.
Team: Amy Burns (Personal Team -
Provisioning Profile: Xcode Managed Profile
Signing Certificate: iPhone Developer:

7. The previous step will automatically generate a provisioning profile and signing identity for you. You can view this by clicking on the information icon next to provisioning profile:



Build: 1

Signing

Automatically manage sign
Xcode will create and update certificates.
Team: Amy Burns (Personal Team -
Provisioning Profile: Xcode Managed Profile
Signing Certificate: iPhone Developer: amyburns_

Deployment Info

Deployment Target: 10.0
Devices: Universal
Main Interface: Main

iOS Team Provisioning Profile: com.xamarin.r...
Created Sep 14, 2016 (Expires in 6 days)

App ID: com.xamarin.recipe.navbartransparent
Matches bundle ID "com.xamarin.recipe.navbartransparent"

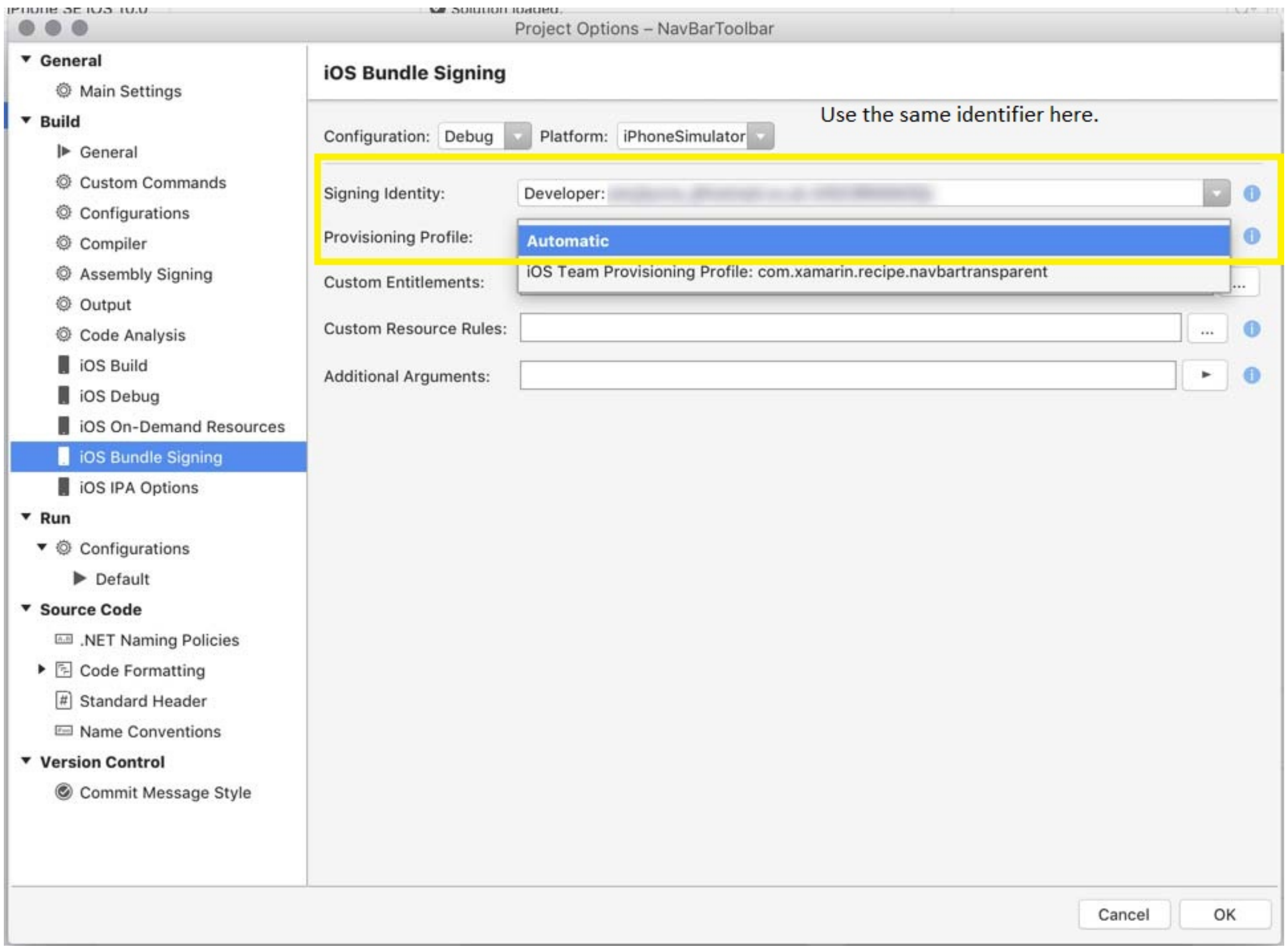
Certificates: 1 Included
Includes newest signing certificate "iPhone Developer:

Team: Amy Burns
Matches team "Amy Burns (Personal Team -)"

Capabilities: 3 Included
Includes Keychain Sharing, In-App Purchase, and Game Center.

8. To test in Xcode, deploy the blank application to your device by clicking the run button.

9. Return to your IDE, with the same device plugged in, and right-click on your Xamarin.iOS project name to open the **Project Options** dialog. Browse to the iOS Bundle Signing section and explicitly set your signing identity and provisioning profile:


Project Options – NavBarToolbar

iOS Bundle Signing

Use the same identifier here.

Configuration: Debug Platform: iPhoneSimulator

Signing Identity: Developer:

Provisioning Profile: Automatic

iOS Team Provisioning Profile: com.xamarin.recipe.navbartransparent

If you cannot see your signing identity or the correct provisioning profile in your IDE, you may need to restart it.

Limitations

Apple has imposed a number of limitations on when and how you can use free provisioning to run your application on an iOS device, ensuring that you can only deploy to *your* device. These are listed in this section.

Access to iTunes Connect is also limited and therefore services such as publishing to the App Store and TestFlight are unavailable to developers provisioning their applications freely. An Apple Developer Account (Enterprise or Personal) is required to distribute via Ad Hoc and In-House means.

Provisioning Profiles created in this way will expire after one week, Signing Identities after one year. Furthermore, provisioning profiles will only be created with explicit App IDs and so you will need to follow the instructions above for every app that you wish to install.

Provisioning for most application services is also not possible with free provisioning. This includes:

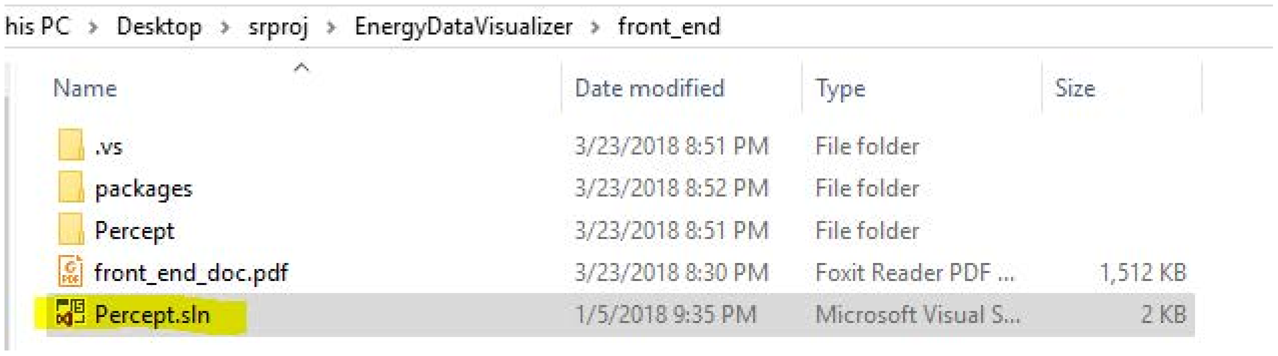
- Apple Pay
- Game Center
- iCloud
- In-App Purchasing
- Push Notifications
- Wallet (Was Passbook)

A full list is provided by Apple in their Supported Capabilities guide. To Provision your app for use with application services, visit the Working with Capabilities guides.

Summary

This guide has explored the advantages and limitations of using free provisioning to install applications on an iOS device. It also went through, step-by-step, using free provisioning to install a Xamarin.iOS app.

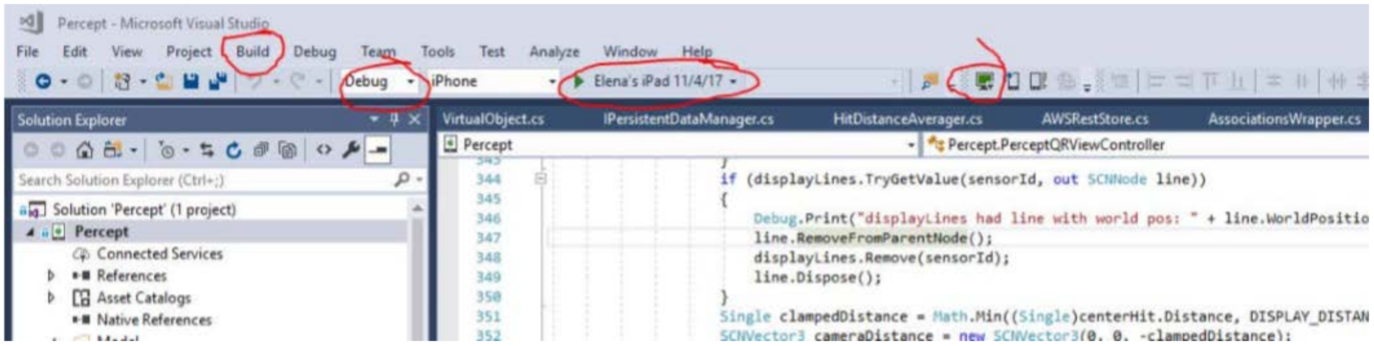
Step 5) Get git, if you don't have it installed. Then git clone the project. Open the solution file.



his PC > Desktop > srproj > EnergyDataVisualizer > front_end

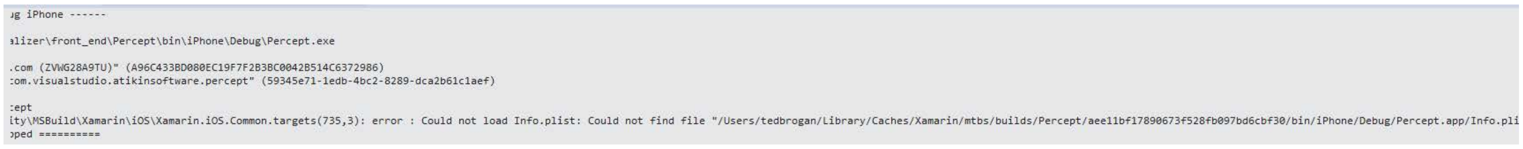
Name	Date modified	Type	Size
.vs	3/23/2018 8:51 PM	File folder	
packages	3/23/2018 8:52 PM	File folder	
Percept	3/23/2018 8:51 PM	File folder	
front_end_doc.pdf	3/23/2018 8:30 PM	Foxit Reader PDF ...	1,512 KB
Percept.sln	1/5/2018 9:35 PM	Microsoft Visual S...	2 KB

Make sure you see your iOS device as a runnable option next to the green play button, and try to build the debug release of the solution.



```
if (displayLines.TryGetValue(sensorId, out SCNNode line))
{
    Debug.Print("displayLines had line with world pos: " + line.WorldPositio
    line.RemoveFromParentNode();
    displayLines.Remove(sensorId);
    line.Dispose();
}
Single clampedDistance = Math.Min((Single)centerHit.Distance, DISPLAY_DISTAN
SCNVector3 cameraDistance = new SCNVector3(0, 0, -clampedDistance);
```

You may get errors referring to old file locations that do not exist, or old provisioning profiles. Go to build > Clean Solution and try to build again. Make sure to reslect a new provisioning profile in the project settings.

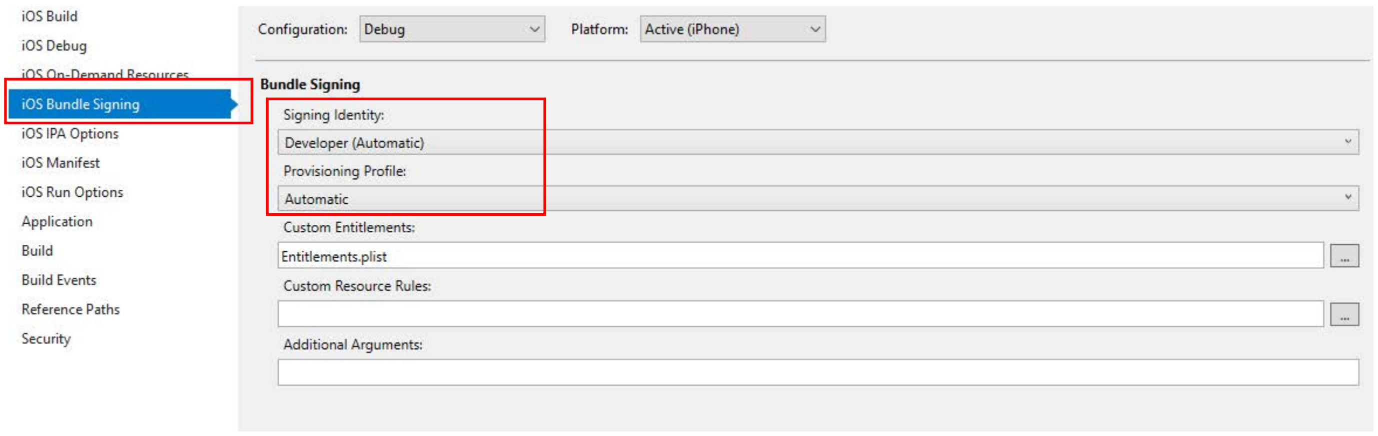


```
ug iPhone ------

alizer\front_end\Percept\bin\iPhone\Debug\Percept.exe

.com (ZVWG28A9TU)" (A96C433BD080EC19F7F2B3BC0042B514C6372986)
com.visualstudio.atikinsoftware.percept" (59345e71-1edb-4bc2-8289-dca2b61c1aef)

:ept
ity\MSBuild\Xamarin\iOS\Xamarin.iOS.Common.targets(735,3): error : Could not load Info.plist: Could not find file "/Users/tedbrogan/Library/Caches/Xamarin/mtbs/builds/Percept/aee11bf17890673f528fb097bd6cbf30/bin/iPhone/Debug/Percept.app/Info.pli
oped ==========
```



Configuration: Debug
Platform: Active (iPhone)

Bundle Signing

Signing Identity: Developer (Automatic)

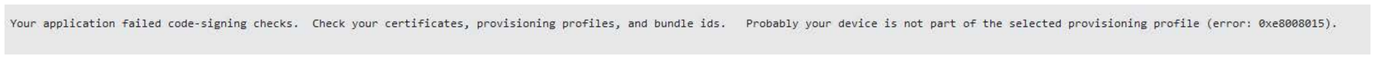
Provisioning Profile: Automatic

Custom Entitlements: Entitlements.plist

Custom Resource Rules:

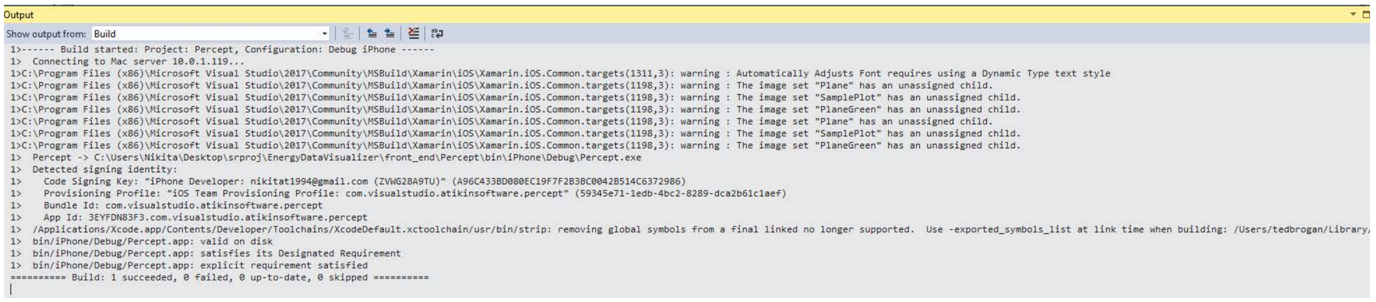
Additional Arguments:

You will get this message if you did not perform Step 4 - Provisioning Profile correctly.



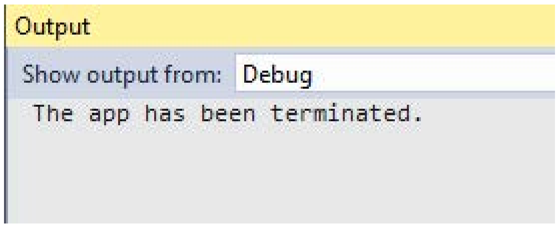
```
Your application failed code-signing checks.  Check your certificates, provisioning profiles, and bundle ids.   Probably your device is not part of the selected provisioning profile (error: 0xe8008015).
```

This is what a sucessful build looks like.



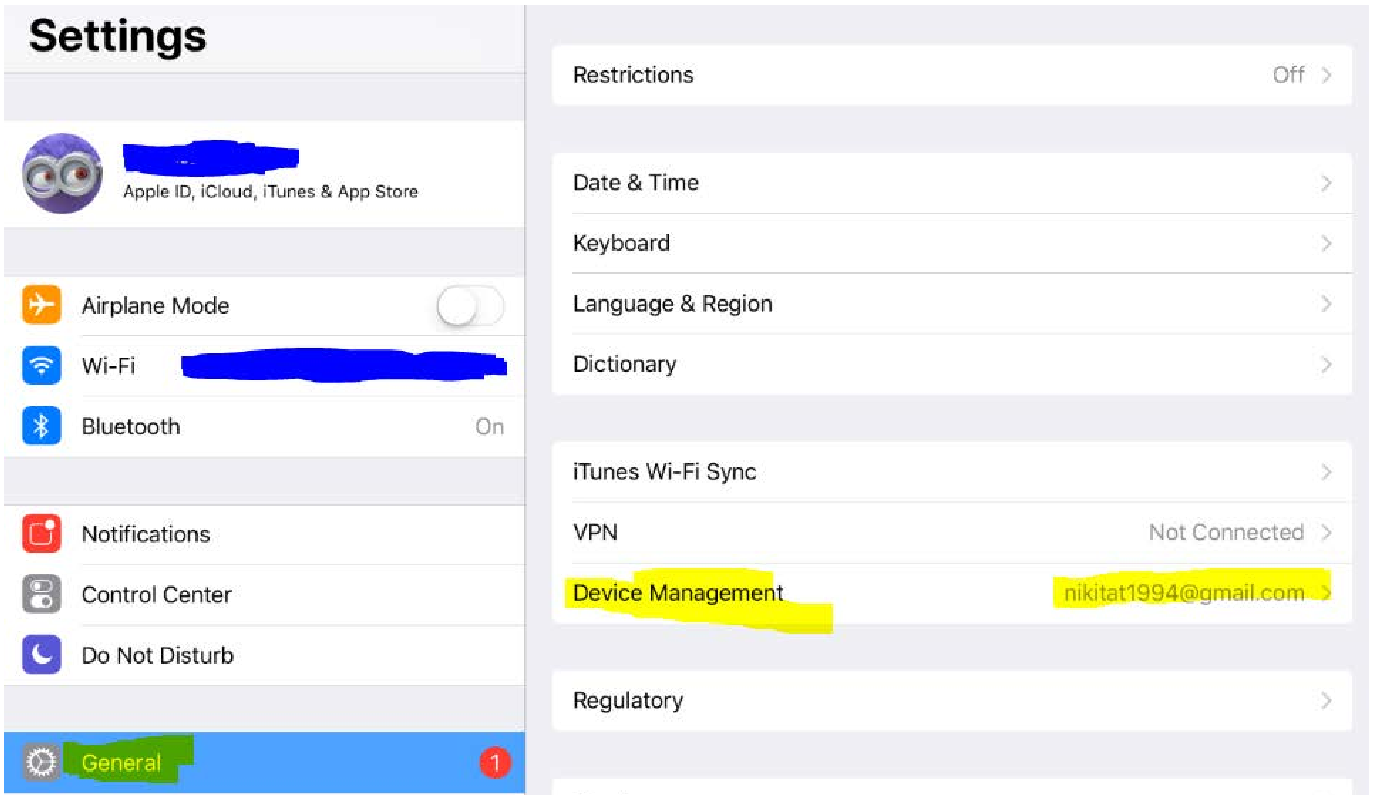
```
1>------ Build started: Project: Percept, Configuration: Debug iPhone ------
1>  Connecting to Mac server 10.0.1.119...
1>C:\Program Files (x86)\Microsoft Visual Studio\2017\Community\MSBuild\Xamarin\iOS\Xamarin.iOS.Common.targets(1311,3): warning : Automatically Adjusts Font requires using a Dynamic Type text style
1>C:\Program Files (x86)\Microsoft Visual Studio\2017\Community\MSBuild\Xamarin\iOS\Xamarin.iOS.Common.targets(1198,3): warning : The image set "Plane" has an unassigned child.
1>C:\Program Files (x86)\Microsoft Visual Studio\2017\Community\MSBuild\Xamarin\iOS\Xamarin.iOS.Common.targets(1198,3): warning : The image set "SamplePlot" has an unassigned child.
1>C:\Program Files (x86)\Microsoft Visual Studio\2017\Community\MSBuild\Xamarin\iOS\Xamarin.iOS.Common.targets(1198,3): warning : The image set "PlaneGreen" has an unassigned child.
1>C:\Program Files (x86)\Microsoft Visual Studio\2017\Community\MSBuild\Xamarin\iOS\Xamarin.iOS.Common.targets(1198,3): warning : The image set "Plane" has an unassigned child.
1>C:\Program Files (x86)\Microsoft Visual Studio\2017\Community\MSBuild\Xamarin\iOS\Xamarin.iOS.Common.targets(1198,3): warning : The image set "SamplePlot" has an unassigned child.
1>C:\Program Files (x86)\Microsoft Visual Studio\2017\Community\MSBuild\Xamarin\iOS\Xamarin.iOS.Common.targets(1198,3): warning : The image set "PlaneGreen" has an unassigned child.
1>  Percept -> C:\Users\Nikita\Desktop\srproj\EnergyDataVisualizer\front_end\Percept\bin\iPhone\Debug\Percept.exe
1>  Detected signing identity:
1>    Code Signing Key: "iPhone Developer: nikitat1994@gmail.com (ZVWG28A9TU)" (A96C433BD080EC19F7F2B3BC0042B514C6372986)
1>    Provisioning Profile: "iOS Team Provisioning Profile: com.visualstudio.atikinsoftware.percept" (59345e71-1edb-4bc2-8289-dca2b61c1aef)
1>    Bundle Id: com.visualstudio.atikinsoftware.percept
1>    App Id: 3EYFDN83F3.com.visualstudio.atikinsoftware.percept
1>  /Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/bin/strip: removing global symbols from a final linked no longer supported.  Use -exported_symbols_list at link time when building: /Users/tedbrogan/Library/
1>  bin/iPhone/Debug/Percept.app: valid on disk
1>  bin/iPhone/Debug/Percept.app: satisfies its Designated Requirement
1>  bin/iPhone/Debug/Percept.app: explicit requirement satisfied
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
```

Run the application with the green play button. If it looks like it did things, but then you see an immediate termination, you need to give permission on your iOS device.



```
The app has been terminated.
```

To give permission, go to the settings application on your iOS device. Go to the general settings, and you should see your developer email under Device Management.



Settings

[redacted]
Apple ID, iCloud, iTunes & App Store

Airplane Mode
Wi-Fi [redacted]
Bluetooth On

Notifications
Control Center
Do Not Disturb

General 1

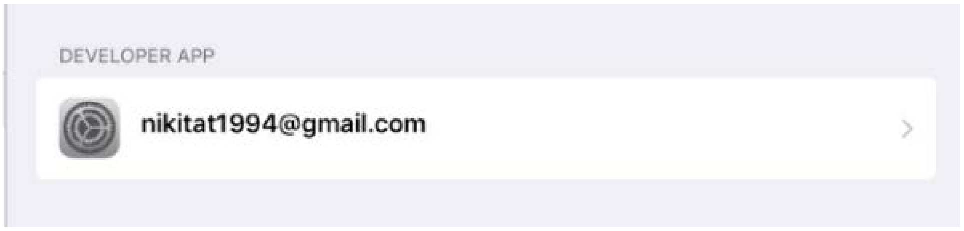
Restrictions Off

Date & Time
Keyboard
Language & Region
Dictionary

iTunes Wi-Fi Sync
VPN Not Connected
Device Management nikitat1994@gmail.com

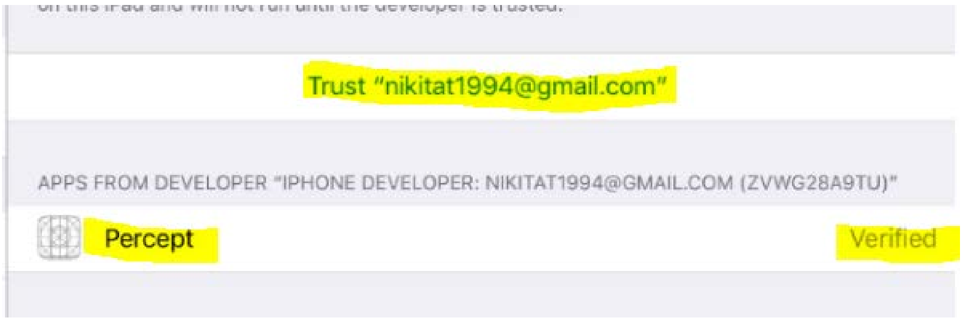
Regulatory

Click on your developer email.


DEVELOPER APP

nikitat1994@gmail.com

You should see your application listed as Verified. Now, you just need to click on Trust for your developer email.

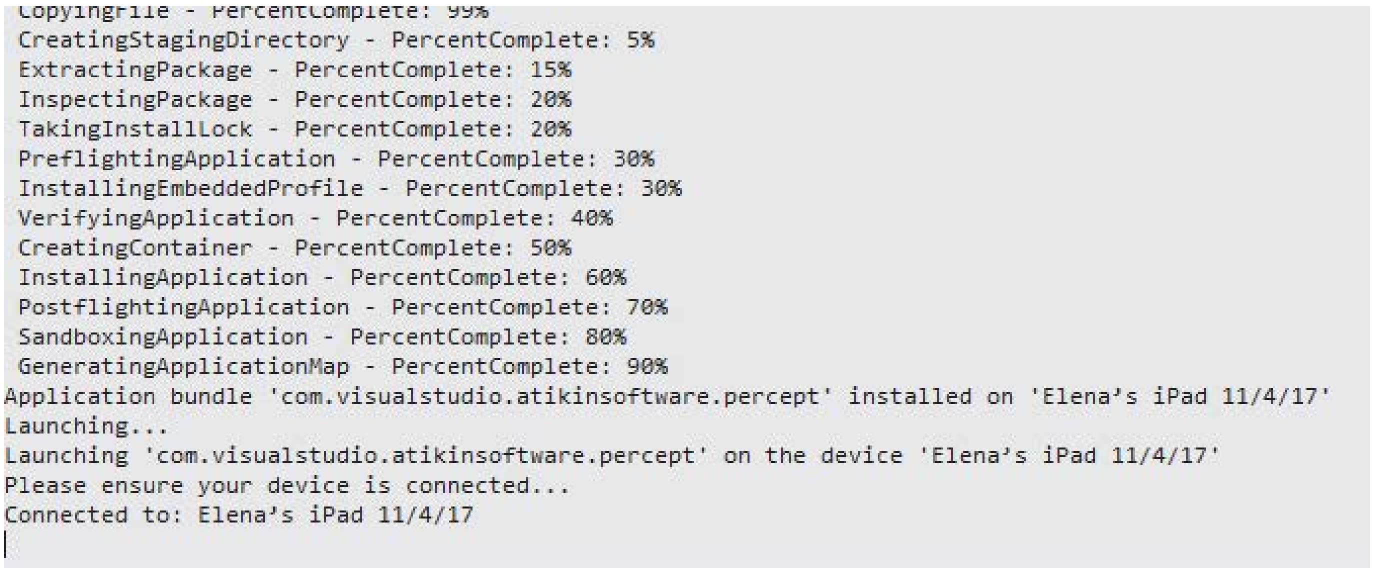

on this iPad and will not run until the developer is trusted.

Trust "nikitat1994@gmail.com"

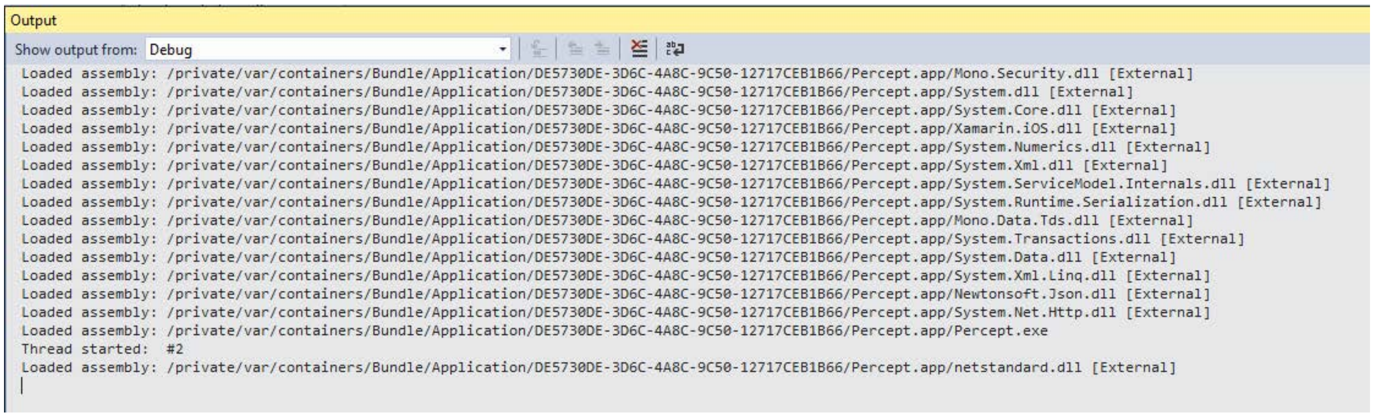
APPS FROM DEVELOPER "IPHONE DEVELOPER: NIKITAT1994@GMAIL.COM (ZVWG28A9TU)"

Percept Verified

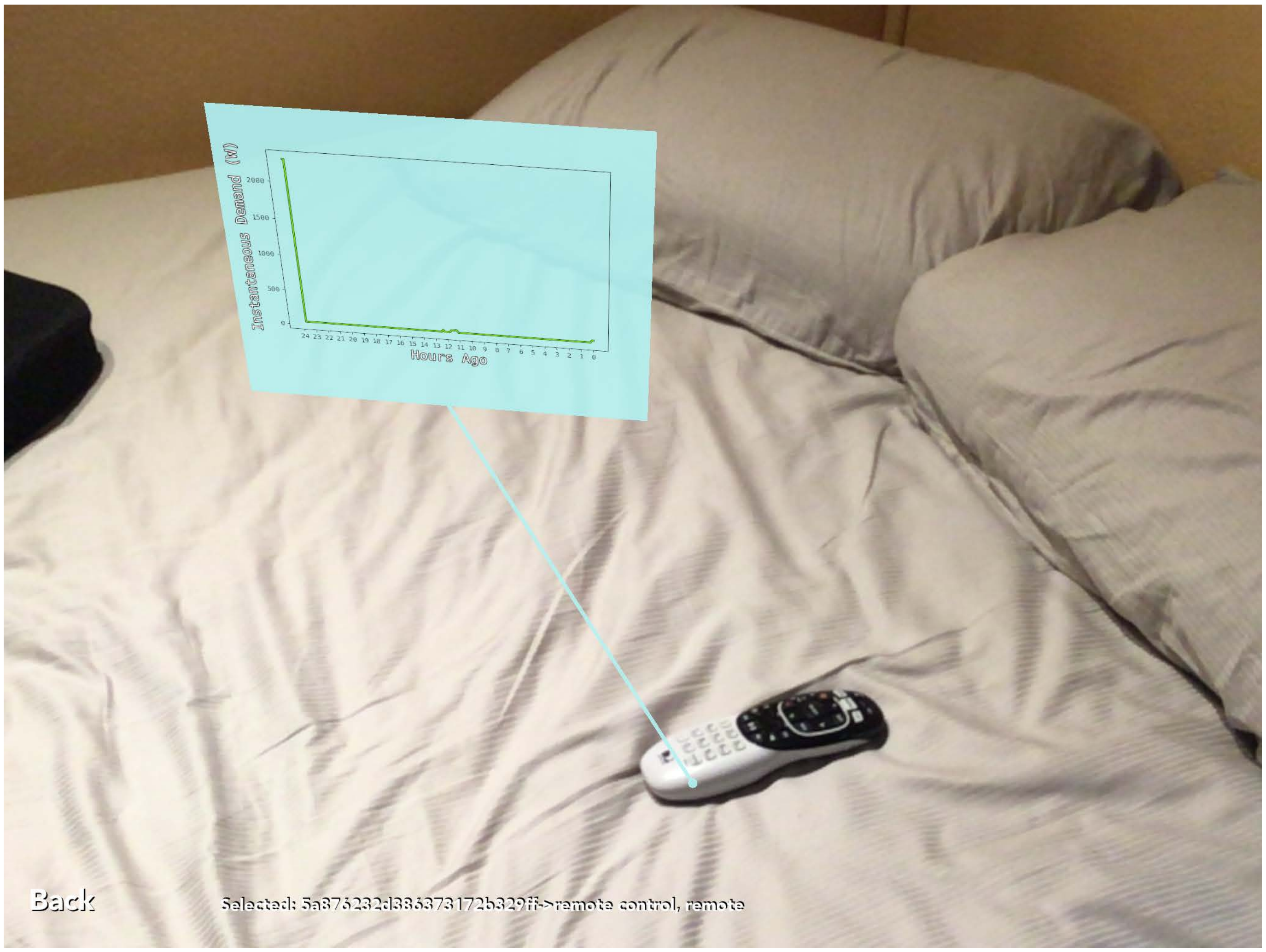
You should now be able to succesfully deploy and run the app.


```
 CopyingFile - PercentComplete: 99%
 CreatingStagingDirectory - PercentComplete: 5%
 ExtractingPackage - PercentComplete: 15%
 InspectingPackage - PercentComplete: 20%
 TakingInstallLock - PercentComplete: 20%
 PreflightingApplication - PercentComplete: 30%
 InstallingEmbeddedProfile - PercentComplete: 30%
 VerifyingApplication - PercentComplete: 40%
 CreatingContainer - PercentComplete: 50%
 InstallingApplication - PercentComplete: 60%
 PostflightingApplication - PercentComplete: 70%
 SandboxingApplication - PercentComplete: 80%
 GeneratingApplicationMap - PercentComplete: 90%
Application bundle 'com.visualstudio.atikinsoftware.percept' installed on 'Elena’s iPad 11/4/17'
Launching...
Launching 'com.visualstudio.atikinsoftware.percept' on the device 'Elena’s iPad 11/4/17'
Please ensure your device is connected...
Connected to: Elena’s iPad 11/4/17
```

This is what sucessful output looks like when you are running the app. It is the application "printing" debug information as it is running, to Visual Studio.



```
Loaded assembly: /private/var/containers/Bundle/Application/DE5730DE-3D6C-4A8C-9C50-12717CEB1B66/Percept.app/Mono.Security.dll [External]
Loaded assembly: /private/var/containers/Bundle/Application/DE5730DE-3D6C-4A8C-9C50-12717CEB1B66/Percept.app/System.dll [External]
Loaded assembly: /private/var/containers/Bundle/Application/DE5730DE-3D6C-4A8C-9C50-12717CEB1B66/Percept.app/System.Core.dll [External]
Loaded assembly: /private/var/containers/Bundle/Application/DE5730DE-3D6C-4A8C-9C50-12717CEB1B66/Percept.app/Xamarin.iOS.dll [External]
Loaded assembly: /private/var/containers/Bundle/Application/DE5730DE-3D6C-4A8C-9C50-12717CEB1B66/Percept.app/System.Numerics.dll [External]
Loaded assembly: /private/var/containers/Bundle/Application/DE5730DE-3D6C-4A8C-9C50-12717CEB1B66/Percept.app/System.Xml.dll [External]
Loaded assembly: /private/var/containers/Bundle/Application/DE5730DE-3D6C-4A8C-9C50-12717CEB1B66/Percept.app/System.ServiceModel.Internals.dll [External]
Loaded assembly: /private/var/containers/Bundle/Application/DE5730DE-3D6C-4A8C-9C50-12717CEB1B66/Percept.app/System.Runtime.Serialization.dll [External]
Loaded assembly: /private/var/containers/Bundle/Application/DE5730DE-3D6C-4A8C-9C50-12717CEB1B66/Percept.app/Mono.Data.Tds.dll [External]
Loaded assembly: /private/var/containers/Bundle/Application/DE5730DE-3D6C-4A8C-9C50-12717CEB1B66/Percept.app/System.Transactions.dll [External]
Loaded assembly: /private/var/containers/Bundle/Application/DE5730DE-3D6C-4A8C-9C50-12717CEB1B66/Percept.app/System.Data.dll [External]
Loaded assembly: /private/var/containers/Bundle/Application/DE5730DE-3D6C-4A8C-9C50-12717CEB1B66/Percept.app/System.Xml.Linq.dll [External]
Loaded assembly: /private/var/containers/Bundle/Application/DE5730DE-3D6C-4A8C-9C50-12717CEB1B66/Percept.app/Newtonsoft.Json.dll [External]
Loaded assembly: /private/var/containers/Bundle/Application/DE5730DE-3D6C-4A8C-9C50-12717CEB1B66/Percept.app/System.Net.Http.dll [External]
Loaded assembly: /private/var/containers/Bundle/Application/DE5730DE-3D6C-4A8C-9C50-12717CEB1B66/Percept.app/Percept.exe
Thread started:  #2
Loaded assembly: /private/var/containers/Bundle/Application/DE5730DE-3D6C-4A8C-9C50-12717CEB1B66/Percept.app/netstandard.dll [External]
```



Selected: 5a876232d386373172b329ff->remote control, remote