

IOS220

# Publishing an iOS App

Download class materials from  
[university.xamarin.com](http://university.xamarin.com)



**Xamarin** University



Information in this document is subject to change without notice. The example companies, organizations, products, people, and events depicted herein are fictitious. No association with any real company, organization, product, person or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user.

Microsoft or Xamarin may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any license agreement from Microsoft or Xamarin, the furnishing of this document does not give you any license to these patents, trademarks, or other intellectual property.

© 2014-2017 Xamarin Inc., Microsoft. All rights reserved.

Xamarin, MonoTouch, MonoDroid, Xamarin.iOS, Xamarin.Android, Xamarin Studio, and Visual Studio are either registered trademarks or trademarks of Microsoft in the U.S.A. and/or other countries.

Other product and company names herein may be the trademarks of their respective owners.



# Objectives

1. Prepare an application for publishing
2. Sign an application
3. Publish an app to the App Store





# Prepare an application for publishing

# Tasks

1. Configure the build settings
2. Fill in the application meta data
3. Add application assets



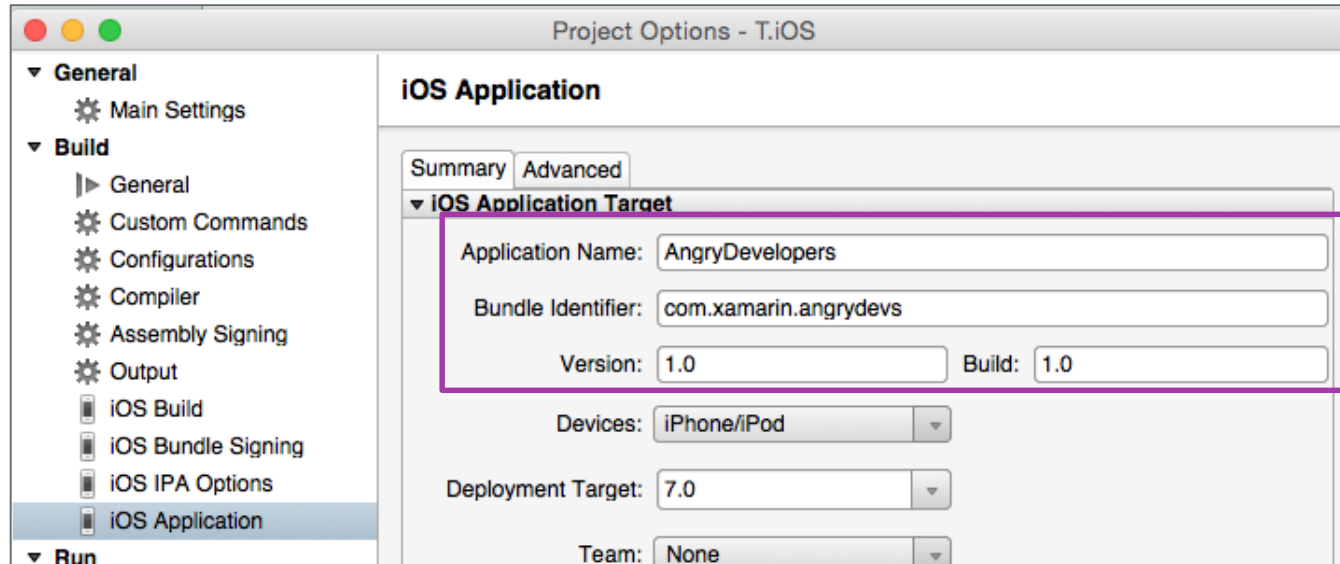
# Requirements

- ❖ Build and submission process requires a Mac w/ XCode
- ❖ Xamarin.iOS uses the same tools that Objective-C & Swift apps use to create and sign the application
- ❖ Any modern Mac will suffice – including the lower-end Mac Mini



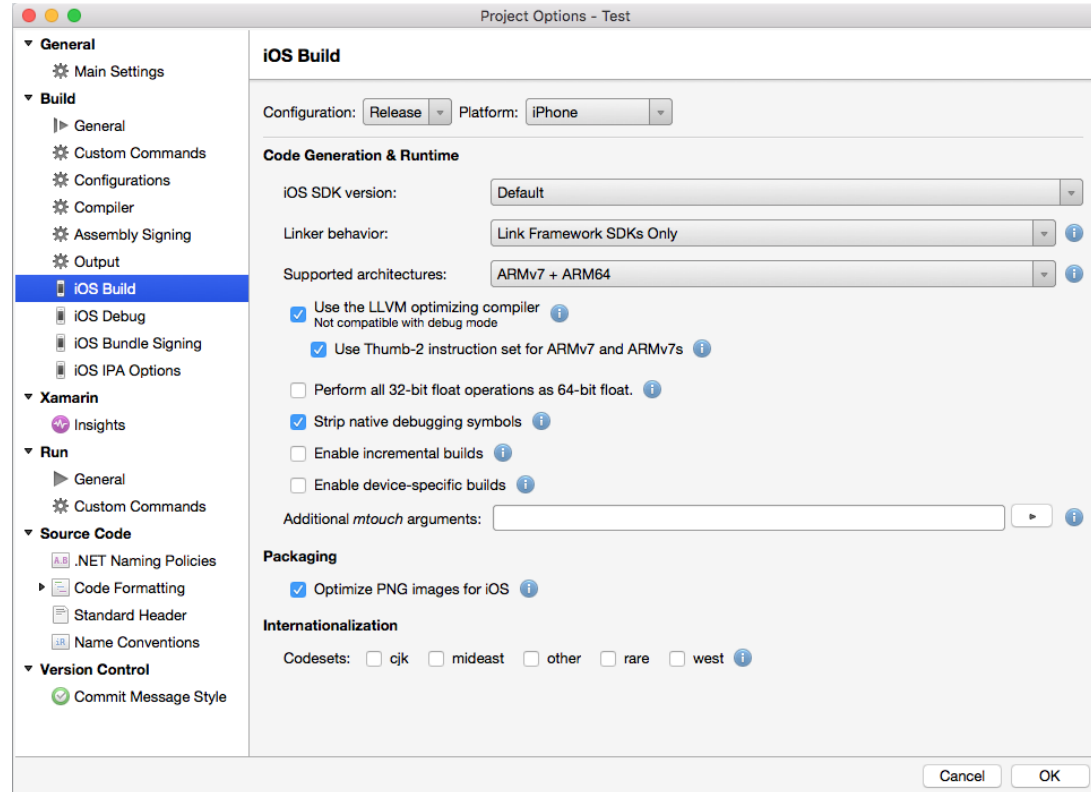
# Application Settings

- ❖ Make sure to set the application name, bundle identifier and version information – these identify your app uniquely to the user and device



# Optimize your build settings

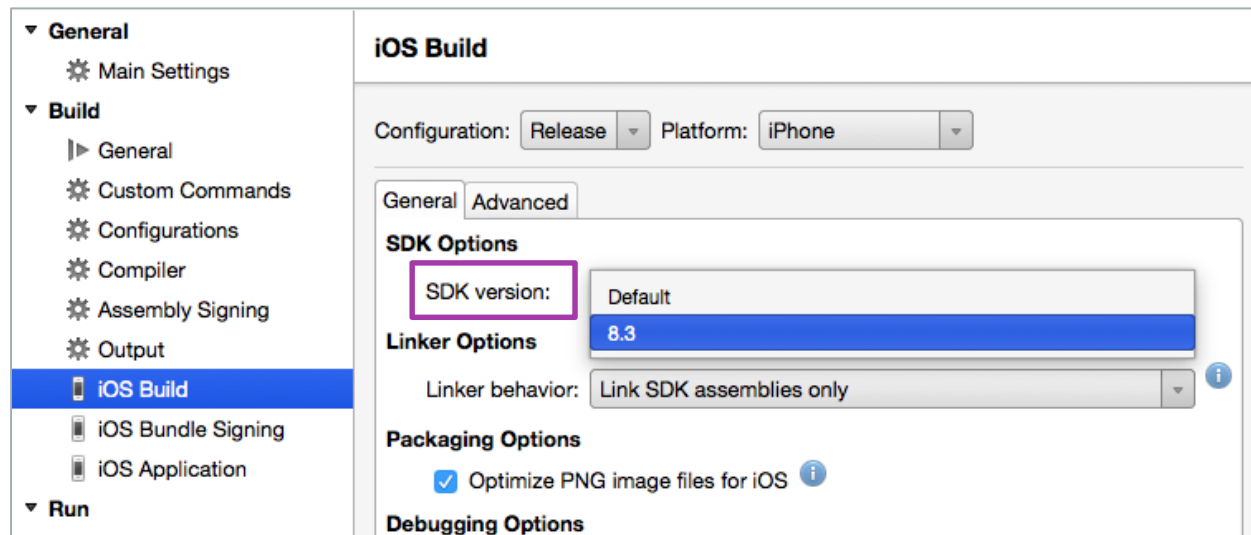
- ❖ iOS Build section has several important settings you should check
  - Architecture
  - LLVM compiler
  - Thumb-2





# Verify the SDK version

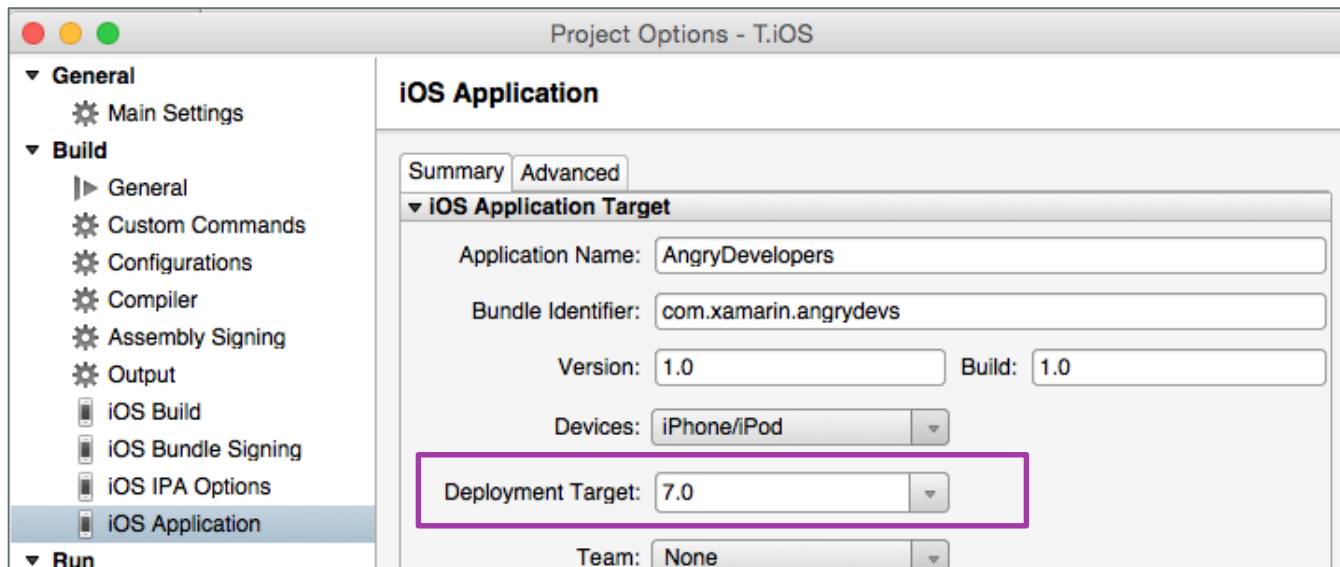
- ❖ You should always build against the latest, released version of the Apple SDK – this is a requirement from Apple



Default selects latest, but you can force a specific SDK (if you have beta versions installed for example)

# Select the proper OS target

- ❖ The deployment target should always reflect the minimum version of iOS you must have in order to run – this affects the APIs available



# Demonstration

Preparing your iOS application for distribution



# Summary

1. Configure the build settings
2. Fill in the application meta data
3. Add application assets





# Sign an application

# Tasks

1. Explore publishing types
2. Create publishing certificates
3. Sign an application



# Types of Publishing

- ❖ There are three primary ways to publish your applications

Three colored parallelograms are arranged horizontally. The first is blue and contains the text 'AdHoc (Testing)'. The second is green and contains the text 'AppStore'. The third is purple and contains the text 'Enterprise'.

AdHoc  
(Testing)

AppStore

Enterprise

# AdHoc publishing

- ❖ Apple allows for limited direct-device publishing specifically to distribute your application for **testing purposes**
- ❖ Can have up to 100 of each type of Apple device (per year)
- ❖ TestFlight automates this process for iOS8+





# App Store publishing

- ❖ App Store publishing is the most common approach, Apple distributes your app and takes a portion of each sale
- ❖ Supports a variety of other features such as in-app purchases, ads and volume purchases

- ✓ You pick the price
- ✓ You get 70% of the sales revenue
- ✓ Receive payments monthly
- ✓ No charge for free apps
- ✓ No credit card fees
- ✓ No hosting fees
- ✓ No marketing fees



# Enterprise publishing

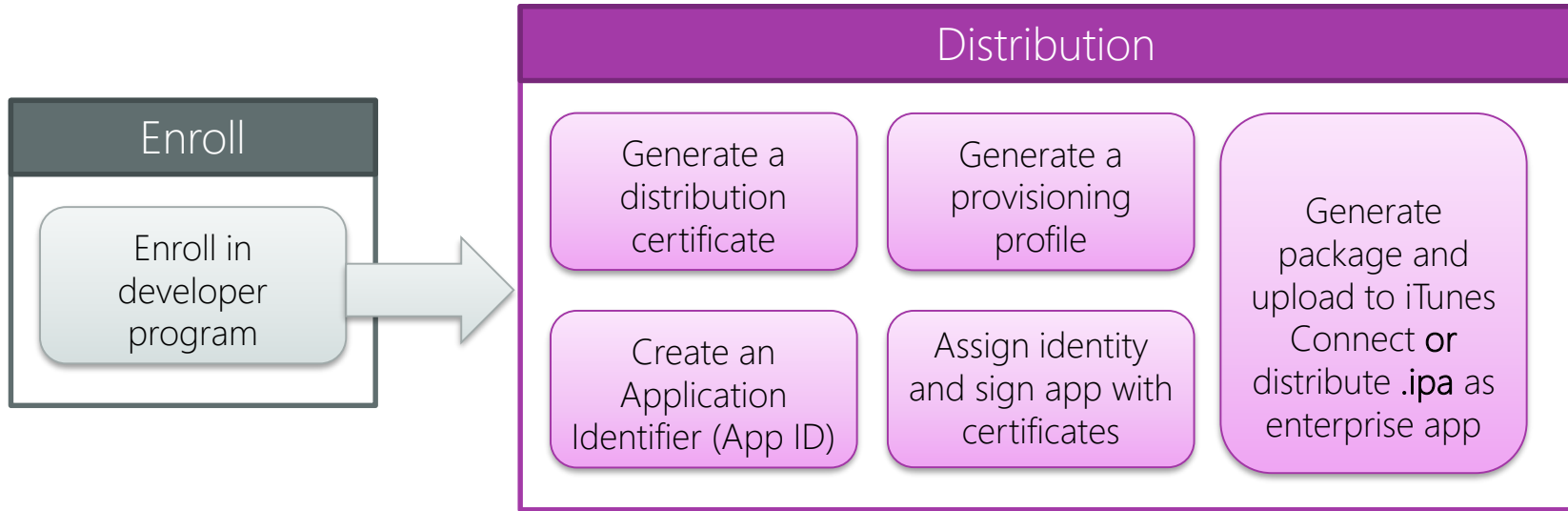
- ❖ Apple also supports an enterprise program – this is intended for larger companies that want to distribute their applications in-house, either through iTunes or over-the-air (OOA) from your own servers

- ✓ Must have **enterprise level account** and meet specific requirements from Apple
- ✓ Can distribute in-house apps to employees
- ✓ No validation process required
- ✓ App certificate expires after some period and must be re-signed to continue distribution



# Distribution process

- ❖ Specific steps are necessary to **distribute** your application



# Logging into the developer portal

- ❖ Creating a developer account (or enterprise account) gives you access to the Member Center ([developer.apple.com/](https://developer.apple.com/))



## SDKs

Download the SDKs and the latest beta software.



## Certificates, Identifiers & Profiles

Manage your certificates, identifiers, devices, and profiles for your apps.



## iTunes Connect

Manage your apps published on the App Store and Mac App Store.

# What certificates will I need

- ❖ There are several related certificates used in the iOS development and distribution process – depending on your app you will need one or more

Developer  
Certificate

The developer certificate is used to sign debug versions of your app that you want to test on real devices – you will need one of these to deploy your app locally to a device you own

# What certificates will I need

- ❖ There are several related certificates used in the iOS development and distribution process – depending on your app you will need one or more

Developer  
Certificate

Push Notifications  
Testing Certificate

If you utilize push notifications,  
you will need a testing  
certificate to communicate  
with Apple's test servers – *this  
is only used for testing*

# What certificates will I need

- ❖ There are several related certificates used in the iOS development and distribution process – depending on your app you will need one or more

Developer  
Certificate

App Store / Ad  
Hoc Certificate

Push Notifications  
Testing Certificate

This is the primary certificate you need – it is used to sign the app and identify it as trusted by any device (since it comes from Apple)

# What certificates will I need

- ❖ There are several related certificates used in the iOS development and distribution process – depending on your app you will need one or

Then you will need different production certificates for each unique Apple server-side service you take advantage of

App Store / Ad Services Certificate

Passbook Certificate

VoIP Certificate

Push Notifications Testing Certificate

Push Notifications Production Cert

Safari Push Notifications

Apple Pay Certificate



# What is the application identifier?

❖ The **Application Identifier** uniquely describes your application on Apple's developer portal and includes:

- Description (internal name)
- A unique Bundle ID
- A list of enabled services

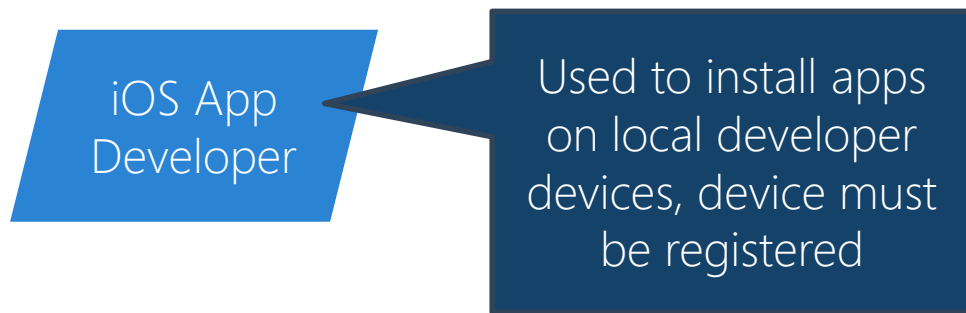
- Enable Services:
- ☐ App Groups
  - ☐ Associated Domains
  - ☐ Data Protection
    - ☐ Complete Protection
    - ☐ Protected Unless Open
    - ☐ Protected Until First User Authentication
  - ☒ Game Center
  - ☐ HealthKit
  - ☐ HomeKit
  - ☐ Wireless Accessory Configuration
  - ☐ Apple Pay
  - ☐ iCloud



Apple's testers will validate the use of the services selected as part of the certification process when submitting to the app store

# What is the provisioning profile?

- ❖ The **Provisioning Profile** is used to glue all the pieces together – it associates a distribution certificate, an App ID, and a set of device IDs and indicates where the application signed with this certificate can run – there are four types of provisioning profiles you can create:



# What is the provisioning profile?

- ❖ The Provisioning Profile is used to glue all the pieces together – it associates a distribution certificate, an App ID, and a set of device IDs and indicates where the application signed with this certificate can run – there are four types of provisioning profiles you can create:



# What is the provisioning profile?

- ❖ The Provisioning Profile is used to glue all the pieces together – it associates a distribution certificate, an App ID, and a set of device IDs and indicates where the application signed with this certificate can run – there are four types of provisioning profiles you can create:



iOS App Developer

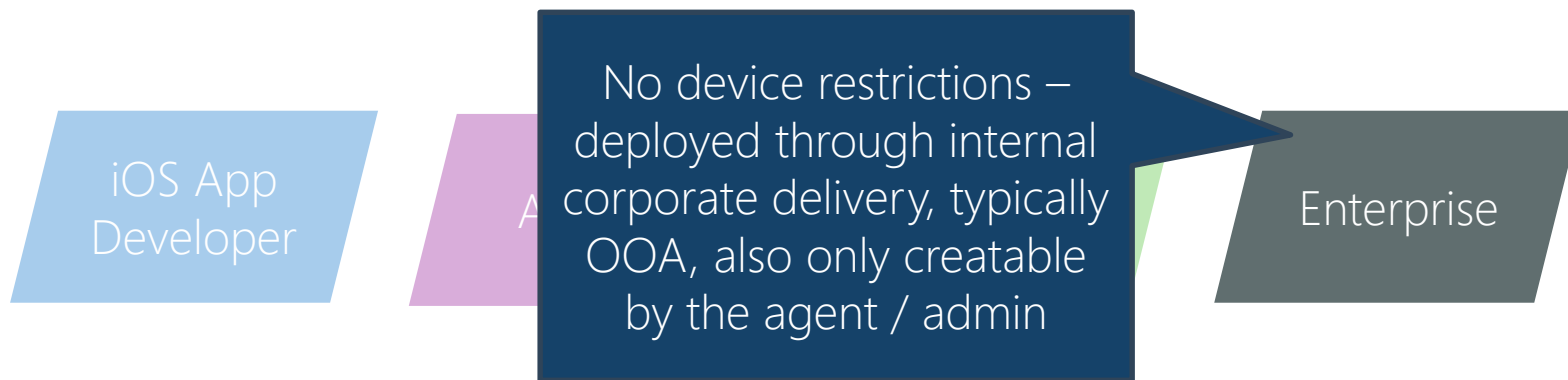
Ad Hoc

App Store

No device restrictions – deployed through App Store, can only be created by the team agent or admin

# What is the provisioning profile?

- ❖ The Provisioning Profile is used to glue all the pieces together – it associates a distribution certificate, an App ID, and a set of device IDs and indicates where the application signed with this certificate can run – there are four types of provisioning profiles you can create:



# Demonstration

Signing your app for distribution



**Xamarin**  
University

# Summary

1. Explore publishing types
2. Create publishing certificates
3. Sign an application



# Publish to the App Store



# Publish an Application

1. Provide application data for publishing
2. Submit an app to iTunes Connect



# What is iTunes Connect?

- ❖ iTunes Connect is a web portal where you can
  - supply app info
  - publish your app
  - get payment info
  - see sales trends
  - manage beta testing
  - ... and more

[itunesconnect.apple.com](https://itunesconnect.apple.com)



# Supplying app metadata

❖ When publishing an application to the App Store, you'll provide meta data describing your application including:

- name, description and version
- app video and screenshots
- SEO keywords
- categories
- ...

## General Information

Bundle ID ?

Xamarin Publish - com.xamarin.xampub

Your Bundle ID com.xamarin.xampub

SKU ?

100000

Apple ID ?

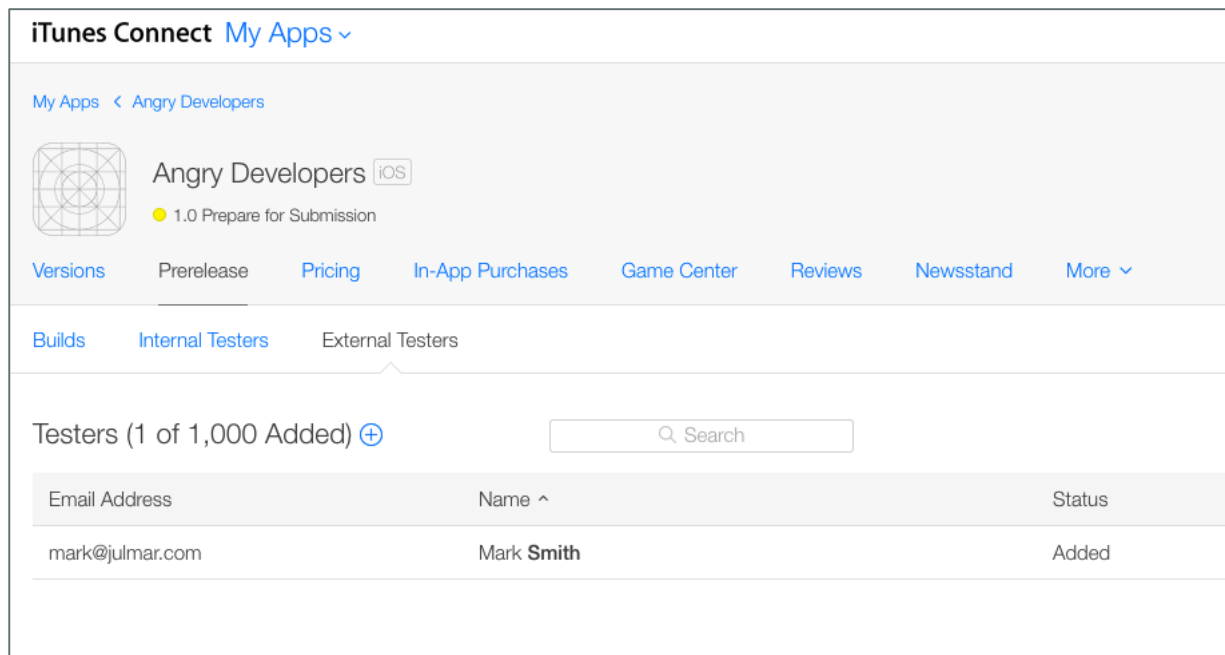
918950660



Tip: can use sites like <https://launchkit.io> to generate app screenshots which look great

# Apple TestFlight Beta Testing

- ❖ Can invite both internal and external users to test your app, system will automatically track the UUID of the device and generate the certificate



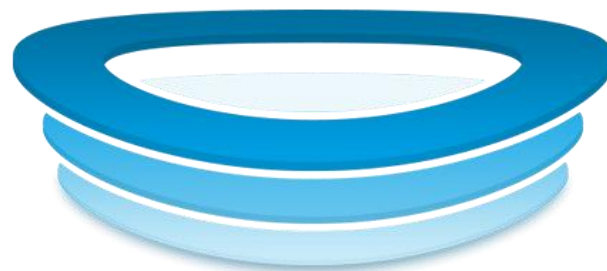
The screenshot shows the iTunes Connect interface for the app 'Angry Developers' (iOS). The 'Internal Testers' tab is selected, displaying a list of testers. The interface includes a search bar and a table with columns for Email Address, Name, and Status.

Testers (1 of 1,000 Added) [+](#)

Email Address	Name ^	Status
mark@julmar.com	Mark Smith	Added

# Alternatives to TestFlight

- ❖ HockeyApp is a Microsoft testing distribution platform which supports all major mobile platforms
  - No forced approval process
  - Can invite testers through email
  - limited to 100 devices per app
  - Supports analytics with an optional SDK

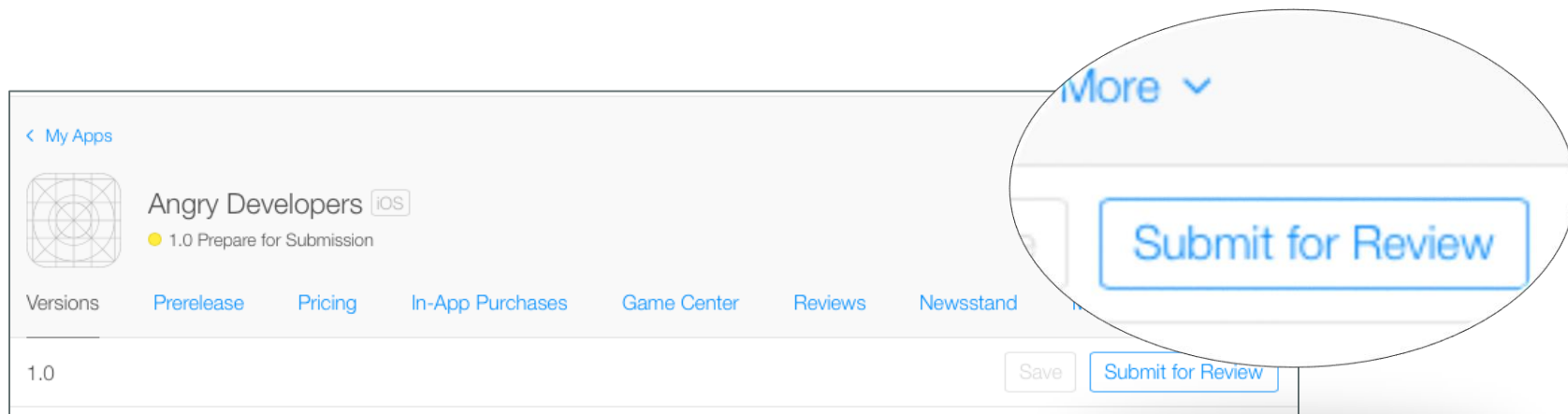


## HOCKEYAPP

[hockeyapp.net/features](https://hockeyapp.net/features)

# Submitting your app

- ❖ Once you have built the app, entered all the information and beta tested the application then you are ready to submit the app for publishing to the AppStore



# App Store size requirements

- ❖ Total uncompressed size must be  $< 4g$  with each CPU architecture packages being less than:
  - $< \text{iOS7} - 80\text{MB}$
  - $\geq \text{iOS7} - 60\text{MB}$
  - Must be  $< 100\text{MB}$  to do cell download
- ❖ Consider moving app data into files vs. embedded resources and compressing any images used



# Demonstration

Submitting your app using iTunes Connect



**Xamarin**  
University



# Common App Rejections

## Top 10 reasons for app rejections during the 7-day period ending April 20, 2015.

- 13% More information needed
- 11% Guideline 2.2: Apps that exhibit bugs will be rejected
- 6% Guideline 10.6: Apple and our customers place a high value on simple, refined, creative, well thought through interfaces. They take more work but are worth it. Apple sets a high bar. If your user interface is complex or less than very good, it may be rejected
- 3% Guideline 2.1: Apps that crash will be rejected
- 3% Guideline 3.4: App names in iTunes Connect and as displayed on a device should be similar, so as not to cause confusion
- 3% Guideline 3.1: Apps or metadata that mentions the name of any other mobile platform will be rejected
- 3% Did not comply with terms in the iOS Developer Program License Agreement
- 3% Guideline 3.8: Developers are responsible for assigning appropriate ratings to their Apps. Inappropriate ratings may be changed/deleted by Apple
- 2% Guideline 22.2: Apps that contain false, fraudulent or misleading representations or use names or icons similar to other Apps will be rejected
- 2% Guideline 3.3: Apps with names, descriptions, screenshots, or previews not relevant to the content and functionality of the App will be rejected

## Total Percent of App Rejections

49% Top 10 Reasons  
51% Other Reasons (<2% each)



[developer.apple.com/app-store/review/rejections/](https://developer.apple.com/app-store/review/rejections/)

# Common App Rejections



Apple

Nov 8, 2012  
08:25 AM

2.5

We found that your app uses one or more non-public APIs, which is not in compliance with the [App Store Review Guidelines](#). The use of non-public APIs is not permissible because it can lead to a poor user experience should these APIs change.

We found the following non-public API/s in your app:

Specifically, we found that your app uses an overlay ontop of the UIStatusBar when taking actions inside of the app. We have attached screenshots for your reference.

If you have defined methods in your source code with the same names as the above-mentioned APIs, we suggest altering your method names so that they no longer collide with Apple's private APIs to avoid your application being flagged in future submissions.

Additionally, one or more of the above-mentioned APIs may reside in a static library included with your application. If you do not have access to the library's source, you may be able to search the compiled binary using "strings" or "otool" command line tools. The "strings" tool can output a list of the methods that the library calls and "otool -ov" will output the Objective-C class structures and their defined methods. These techniques can help you narrow down where the problematic code resides.

We appreciate that you may have made the precautions in your code for using non-public APIs, however, there is no way to accurately or completely predict how an API may be modified and what effects those modifications may have. For this reason, we do not permit the use of non-public APIs in App Store apps.

[developer.apple.com/app-store/review/rejections/](https://developer.apple.com/app-store/review/rejections/)

# Where to go from here?

- ❖ The steps required to publish a Xamarin.iOS application are exactly the same as those for applications written in Swift or Objective C – Apple has excellent documentation on the signing and publishing process



Apple's documentation on publishing covers everything from advanced topics to common issues

[developer.apple.com](https://developer.apple.com)

# Summary

1. Provide application data for publishing
2. Submit an app to iTunes Connect



# Thank You!

Please complete the class survey in your profile:  
[university.xamarin.com/profile](https://university.xamarin.com/profile)