

# App Distribution Guide

# Contents

## **About App Distribution** 10

At a Glance 10

Enroll in an Apple Developer Program to Distribute Your App 11

Generate Certificates and Register Your Devices 11

Add Store Capabilities to Your App 11

Prepare Your App for Distribution 12

Test iOS Apps Across Numerous Devices 12

Submit and Release Your App 12

How to Use This Document 13

See Also 13

## **Enrolling in an Apple Developer Program and Accessing Its Tools** 15

Enrolling in an Apple Developer Program 15

You Enroll as an Individual or a Company 15

You Can Join Multiple Teams 15

Emails from Apple Contain Further Instructions and Welcome You 16

Accessing Member Center and iTunes Connect 16

Accessing Member Center 16

Managing Your Certificates, Identifiers, and Profiles 17

Accessing iTunes Connect 22

Bookmarking the Web Tools 23

Recap 23

## **Creating Your Signing Certificates** 24

About Code Signing 24

Requesting Signing Certificates 26

Verify Your Steps 30

Troubleshooting 35

Your Signing Certificates in Depth 35

Recap 37

## **Developing Apps Using the Team Provisioning Profile** 38

About the Team Provisioning Profile 38

Adding Devices to Your Team Provisioning Profile 40

- Registering and Provisioning an iOS Device Using Xcode 40
- Registering and Provisioning a Mac Using Xcode 41
- Verify Your Steps 43
- Troubleshooting 48
- Code Signing Your App Using the Team Provisioning Profile 49
  - Troubleshooting 51
- Launching Your iOS App on the Device 52
  - Troubleshooting 53
- Recap 53
  
- Provisioning Your App for Store Technologies 54**
  - About Development Provisioning Profiles 55
  - Before You Begin 56
  - Creating App IDs 57
    - Registering an App ID 57
  - Enabling Store Technologies 62
  - Creating Development Provisioning Profiles 63
  - Regenerating the Provisioning Profile 65
    - Regenerating the Team Provisioning Profile 66
    - Regenerating Provisioning Profiles Managed By You 67
  - Provisioning Your Development Devices 69
    - Refreshing Your Provisioning Profiles Using Xcode 70
    - Updating Provisioning Profiles on Your Device 70
  - Setting the Bundle ID to Match Your App ID 70
  - Signing Your App Using Your Development Provisioning Profile 71
  - Verify Your Steps 73
    - Verify Code Signing 73
    - Verify the App ID Settings in Member Center 73
  - Troubleshooting 76
    - Troubleshooting Code Signing Errors 76
    - Troubleshooting Failure to Launch 76
  - Development Provisioning Profiles in Depth 77
  - Recap 78
  
- Configuring Store Technologies in Xcode and iTunes Connect 79**
  - About Entitlements 79
  - Configuring iCloud 79
    - Enabling iCloud Entitlements 80
    - Configuring iCloud Key-Value Storage 81
    - Configuring iCloud Document Storage 81

- Configuring Push Notifications 82
  - Creating Push Notification Client SSL Certificates 83
  - Installing Client SSL Certificates 86
- Configuring Game Center 86
- Configuring In-App Purchase 87
- Configuring Passbook for iOS Apps 88
- Configuring Data Protection for iOS Apps 89
- Configuring Routing Apps for iOS Apps 90
  - Providing Routing Directions 91
  - Enabling Routing Apps in Xcode 91
  - Creating an App Record in iTunes Connect 91
  - Submitting a Binary to the Store 92
  - Uploading the Geographic Coverage File to iTunes Connect 92
- Configuring Newsstand for iOS Apps 92
- Verify Your Steps 92
- Recap 97

## **Configuring Your Xcode Project for Distribution** 98

- About Bundle IDs 98
- Before You Begin 100
- Setting Properties When Creating Your Xcode Project 100
- Configuring Application Target Settings 102
  - Setting the Mac Application Category 103
  - Setting the Bundle ID 103
  - Setting the Version Number and Build String 104
  - Setting the Target iOS Devices 105
  - Setting the Deployment Target 105
- Adding App Icons and Launch Images 106
  - Setting App Icons 107
  - Creating and Setting iOS Launch Images 107
- Configuring Entitlements 110
- Configuring App Sandbox for Mac Apps 112
- Editing the Information Property List 113
  - Setting the Copyright Key for Mac Apps 114
- Specifying Build Settings 114
  - Setting Architectures for iOS Apps 115
  - Setting the Base SDK 116
  - Setting the Debug Information Format for Mac Apps 116
- Recap 116

- Beta Testing Your iOS App** 117
  - About Ad Hoc Provisioning Profiles 117
  - Creating Your App Record in iTunes Connect 118
  - Registering Test Devices 118
  - Creating Distribution Certificates 119
    - Verify Your Steps 119
  - Creating Ad Hoc Provisioning Profiles 120
  - Archive and Validate Your App 121
    - Code Signing Your App 122
    - Review the Archive Scheme Settings 123
    - Creating and Validating an Archive 124
  - Creating an iOS App Store Package 127
    - Troubleshooting 128
  - Installing Your App on Test Devices 129
  - Soliciting Crash Reports from Testers 130
  - Ad Hoc Provisioning Profiles in Depth 131
  - Recap 131

## **Analyzing Crash Reports** 132

- Submitting Your App** 133
  - About Store Provisioning Profiles 133
  - Before You Begin 134
  - Creating Distribution Certificates 135
    - Verify Your Steps 135
  - Creating Store Provisioning Profiles 136
    - Downloading the Distribution Provisioning Profile 138
    - Verify Your Steps 138
  - Archiving and Validating Your App 139
    - Code Signing Your App 139
    - Review the Archive Scheme Settings 140
    - Creating and Validating an Archive 141
  - Test the Mac Installer Package 143
  - Submitting Your App Using Xcode 144
    - Submitting Your iOS App 144
    - Submitting Your Mac App 147
    - Troubleshooting 149
  - Recap 149

## **Releasing and Updating Your App** 150

Recap 150

**Managing Your App in iTunes Connect** 151

About iTunes Connect User Roles and Privileges 151

Adding iTunes Connect Users 153

Creating an App Record 153

Viewing the Status of Your App 153

Changing the Availability Date of Your App 154

Viewing Crash Reports 155

Viewing Customer Reviews 156

Creating New Versions of Your App 157

Recap 157

**Best Practices for Maintaining Certificates and Provisioning Profiles** 158

About Protecting Your Code Signing Identities 158

Exporting and Importing Certificates and Provisioning Profiles 159

    Exporting Your Developer Profile 159

    Importing Your Developer Profile 160

Removing Certificates from Your Keychain 161

Revoking Certificates 164

Replacing Expired Certificates 166

Installing Missing Intermediate Certificate Authorities 167

Requesting Additional Developer ID Certificates 168

Registering App IDs 169

Deleting App IDs 169

Registering Devices Using Member Center 170

    Locating Device IDs 171

    Registering Individual Devices 172

    Registering Multiple Devices 173

Editing Provisioning Profiles 175

Installing and Removing Provisioning Profiles from Devices 177

Removing Provisioning Profiles from Your Team 179

Renewing Expired Provisioning Profiles 180

Downloading Provisioning Profiles from Member Center 180

Re-Creating Certificates and Updating Related Provisioning Profiles 181

Recap 183

**Managing Your Team** 184

About Apple Developer Program Team Roles and Privileges 184

    Team Roles 184

- Team Privileges 185
- Team Agent 186
- Before You Begin 187
- Inviting Team Members and Assigning Roles 187
  - Inviting Team Members 187
  - Changing Team Roles 189
- Approving Development Certificates 190
- Registering Team Member Devices 192
- Recap 193
  
- Distributing Applications Outside the Mac App Store 194**
- Creating Developer ID-Signed Applications or Installer Packages 194
  - Requesting Developer ID Certificates 194
  - Code Signing Your Application 198
  - Exporting a Developer ID-Signed Application 199
  - Signing an Installer Package 201
- Verify Your Steps 202
  - Enabling and Disabling Gatekeeper 202
  - Testing Gatekeeper Behavior 204
- Recap 207
  
- Troubleshooting 208**
- Certificate Issues 208
  - Your Provisioning Profile Doesn't Appear in the Code Signing Identity Menu 208
  - Duplicate Provisioning Profile Appear in the Devices Organizer 208
  - Your Certificates Are Invalid Because You're Missing Private Keys 208
  - Your Developer ID Certificates Are Invalid Because You're Missing Private Keys 209
  - Your Certificates Are Invalid Because You're Missing an Intermediate Certificate 209
  - Your Certificates Have Trust Issues 209
  - Your Certificates Have Expired 209
  - You're Missing Signing Certificates 209
  - You Have Duplicate Certificates 210
- Provisioning Issues 210
  - Xcode Cannot Install Your App on Your Development Device 210
  - Your Provisioning Profile Has Expired 210
- Build and Code Signing Issues 210
  - Xcode Cannot Find Your Provisioning Profile 211
  - Xcode Doesn't Trust Your Certificate 211
  - The Code Signing Identity Build Setting Doesn't Match Any Certificates 211
  - Your Keychain Contains Duplicate Code Signing Identities 212

- [The App ID of Your Provisioning Profile Doesn't Match Your App's Bundle Identifier](#) 213
  - [Device Is Not Listed as a Run Destination](#) 213
- [Debugging Information Issue](#) 213
  - [Xcode Displays the Unknown iOS Detected Dialog When You Connect a Device](#) 213
- [Document Revision History](#) 214
- [Glossary](#) 215



# Figures and Tables

## **Creating Your Signing Certificates** 24

Table 2-1 Certificate types and names 36

## **Provisioning Your App for Store Technologies** 54

Table 4-1 Tasks you perform to configure store technologies 56

## **Configuring Your Xcode Project for Distribution** 98

Figure 6-1 Common uses for an app's bundle ID 99

## **Managing Your App in iTunes Connect** 151

Table 11-1 iTunes Connect roles and responsibilities 152

Table 11-2 Abbreviated list of iTunes Connect modules, including availability by role 152

## **Best Practices for Maintaining Certificates and Provisioning Profiles** 158

Table 12-1 Team certificate revoking privileges 164

## **Managing Your Team** 184

Table 13-1 Team roles 184

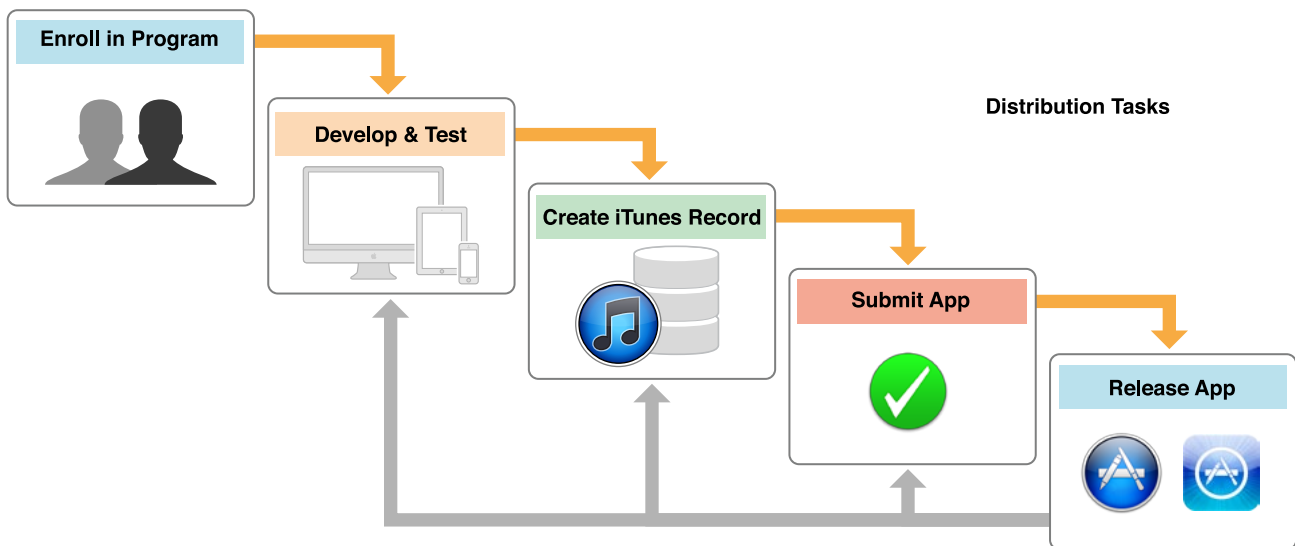
Table 13-2 Privileges assigned to each membership level 185

# About App Distribution

This guide explains how to develop, test, submit, and release your iOS and Mac apps. By understanding your tools and the distribution process, you'll be able to get your new app and updates to your customers faster.

To submit your app to the store, you use Xcode features and several web tools available only to members of an Apple Developer Program. Even before you can develop with technologies, such as iCloud and Game Center, you must join an Apple Developer Program. You should join a program even if you distribute your application outside of the Mac App Store so that customers know your application comes from a known source.

Once you join a program, you can start using store services from your app, testing your app on devices, providing marketing, sales, and contact information, and submitting versions of your app for approval. You iterate the steps of this distribution process, as necessary, until your app is approved and released. Then you repeat some of these steps again for each subsequent update.



## At a Glance

This guide contains everything you need to distribute an app through the App Store or Mac App Store.

- Get step-by-step guidance on enrolling in an Apple Developer Program and building, testing, and submitting your app.
- Configure technologies that are only available to apps submitted to the App Store or Mac App Store.

- Verify that you've prepared your app correctly, and find troubleshooting techniques.
- Learn how to maintain your app and program assets after submission.

## Enroll in an Apple Developer Program to Distribute Your App

To distribute your apps on the App Store and Mac App Store, or to sign apps that you distribute outside the Mac App Store with a Developer ID, you must join an Apple Developer Program. As a member, you'll have access to the resources you need to configure technologies and to submit new apps and updates.

---

**Related Chapter:** [“Enrolling in an Apple Developer Program and Accessing Its Tools”](#) (page 15)

---

## Generate Certificates and Register Your Devices

Apple implements an underlying security model to protect both user data and your app from being modified and distributed without your knowledge. So throughout the development process, you create assets and enter information that Apple will use to identify you, your devices, and your apps. During the lifetime of your Developer Program membership, you'll maintain these assets.

---

**Related Chapters:** [“Creating Your Signing Certificates”](#) (page 24), [“Developing Apps Using the Team Provisioning Profile”](#) (page 38), [“Best Practices for Maintaining Certificates and Provisioning Profiles”](#) (page 158)

---

## Add Store Capabilities to Your App

The store provides advanced, integrated services for certain types of apps, such as games and Newsstand apps, and for additional sources of revenue, such as In-App Purchase and iAd Network. All store technologies require additional configuration—both during development and later, when you are submitting your app to the store. Good examples are Game Center and iCloud. You'll learn how to create a custom provisioning profile to start adding these store capabilities to your app.

---

**Related Chapters:** [“Provisioning Your App for Store Technologies”](#) (page 54), [“Configuring Store Technologies in Xcode and iTunes Connect”](#) (page 79)

---

## Prepare Your App for Distribution

Before you distribute your app for testing or submit it to the store for approval, you need to complete the configuration of your Xcode project. The Xcode project contains required app icons and launch images, contains additional entitlements for technologies you add, and determines which devices and operating system your app supports.

---

**Related Chapter:** [“Configuring Your Xcode Project for Distribution”](#) (page 98)

---

## Test iOS Apps Across Numerous Devices

If you have an iOS app, make sure you test it not only in iOS Simulator but on all the devices and releases that your app supports. Testing on more than one kind of device ensures that your app operates exactly as you thought it would, no matter which device it’s running on. You can register up to 100 devices for use for development and testing. After testing an app yourself, distribute it to testers. You’ll first create a special profile—an ad hoc provisioning profile—to ensure that test versions of your app are not copied and distributed without your knowledge, and then collect device IDs from testers you’ve selected.

---

**Related Chapters:** [“Beta Testing Your iOS App”](#) (page 117), [“Analyzing Crash Reports”](#) (page 132)

---

## Submit and Release Your App

Submitting your app to the store is a multistep process. First, you sign in to iTunes Connect and enter necessary information to change the state of your app record to “Waiting for Upload” or later. If you are selling your app on the store, you provide the information for your reimbursement on iTunes Connect, too. Double-check that you have the certificates for distribution. Then create an archive and sign it with your distribution assets. Last, submit your app using Xcode or Application Loader. When your app is approved, use iTunes Connect to release it by setting the date when the app will be available to customers. If you are distributing your Mac app outside the store, you follow a slightly different process.

**Related Chapters:** [“Submitting Your App”](#) (page 133), [“Releasing and Updating Your App”](#) (page 150), [“Managing Your App in iTunes Connect”](#) (page 151), [“Distributing Applications Outside the Mac App Store”](#) (page 194)

---

## How to Use This Document

Begin by reading the first three chapters in sequence to learn the essential steps and concepts for developing for the store. If you add a store-specific technology to your app, read [“Provisioning Your App for Store Technologies”](#) (page 54) followed by [“Configuring Store Technologies in Xcode and iTunes Connect”](#) (page 79) to learn how to configure that technology. Before you distribute your app, read [“Configuring Your Xcode Project for Distribution”](#) (page 98) to perform final configuration steps. All iOS developers need to read the [“Beta Testing Your iOS App”](#) (page 117) chapter (you should never submit an app to the store without first testing it on many different devices). After thorough testing, read [“Submitting Your App”](#) (page 133) for how to do so. After Apple approves your app, read [“Releasing and Updating Your App”](#) (page 150) for how to set the availability date. If you decide to distribute outside of the Mac App Store, read [“Distributing Applications Outside the Mac App Store”](#) (page 194). After you’ve had your Apple Developer Program account for a while, learn how to maintain your assets by reading [“Best Practices for Maintaining Certificates and Provisioning Profiles”](#) (page 158). And if you enroll as a company, read [“Managing Your Team”](#) (page 184) for a description of the team roles and additional administrative tasks you perform throughout your project. Refer to the glossary for the definitions of terms used in the document.

## See Also

You should already be familiar with the software and tools you use to write code before reading this document. If not, there are a number of platform-specific tutorials you should read first. Then read the technology overview documents followed by the appropriate human interface guidelines for your platform, and most important, the guidelines for submitting your app to the store.

	iOS	Mac
To get started . . .	<i>Start Developing iOS Apps Today</i> <i>App Store Submission Tutorial</i>	<i>Start Developing Mac Apps Today</i>
To learn more about technologies . . .	<i>iOS Technology Overview</i> <i>iOS App Programming Guide</i>	<i>Mac Technology Overview</i> <i>Mac App Programming Guide</i>

	iOS	Mac
To learn about the user interface guidelines . . .	<i>iOS Human Interface Guidelines</i> <i>App Store Review Guidelines for iOS Apps</i>	<i>OS X Human Interface Guidelines</i> <i>App Store Review Guidelines for Mac Apps</i>
To learn more about tools . . .	<i>Xcode User Guide</i> <i>iTunes Connect Developer Guide</i> <i>iOS Simulator User Guide</i>	<i>Xcode User Guide</i> <i>iTunes Connect Developer Guide</i>

# Enrolling in an Apple Developer Program and Accessing Its Tools

**Apple Developer Programs** offer a complete set of technical resources, support, and access to prerelease software—providing everything you need to create innovative applications for iOS and Mac, extensions for Safari, and accessories for iPod, iPhone, and iPad. After you enroll in the iOS Developer Program or Mac Developer Program, you'll have full access to Member Center and iTunes Connect.

## Enrolling in an Apple Developer Program

During enrollment, you'll be asked for basic personal information, including your legal name and address. If you're enrolling as a company or organization, you'll need to provide a few more things, like your legal entity name and D-U-N-S Number, as part of the verification process. Once your information is verified, you'll review license agreements, purchase your program on the Apple Online Store, and receive details on how to activate your membership.

You can always add more Apple Developer Program memberships to your account. For example, you can first join the iOS Developer Program and later add the Mac Developer Program and the Safari Developer Program.

To enroll in an Apple Developer Program, go to [Apple Developer Program Enrollment](#).

## You Enroll as an Individual or a Company

During the enrollment process, you choose whether to enroll as an individual or a company. If you enroll as an **individual**, you are considered a one-person team, one who can perform all the tasks described in this document except manage multiple team members.

If you enroll as a **company**, you may add other persons to your team and grant them privileges to manage your account. All team members must be Registered Apple Developers. Team members have different privileges, so depending on your role, you may not be able to perform all the tasks in this book. To learn about the different roles and privileges, read "[About Apple Developer Program Team Roles and Privileges](#)" (page 184).

## You Can Join Multiple Teams

You can use your Apple ID to join multiple teams but with some restrictions.

Registered Apple Developers are given an **Apple ID** that identifies a person, not a membership in an Apple Developer Program. The Apple ID must have a unique email address associated with it that is verified by Apple. You'll use your Apple ID to sign in to Member Center and iTunes Connect.

A single Apple ID can be associated with multiple Member Center teams, but can only be associated with a single iTunes Connect team. Consequently, developers need to create another Apple ID for different individual or company accounts that they want to manage in iTunes Connect.

## Emails from Apple Contain Further Instructions and Welcome You

When you enroll in an Apple Developer Program or are invited to join a team, you receive a series of emails. For example, if you need to register as an Apple developer, Apple sends you an email requesting that you confirm your email address. Be sure to read and follow the instructions in these emails promptly to streamline the enrollment process.

## Accessing Member Center and iTunes Connect

Although most administrative tasks can be done in Xcode, you may need to use web tools to manage your assets and enter metadata about your app.

### Accessing Member Center

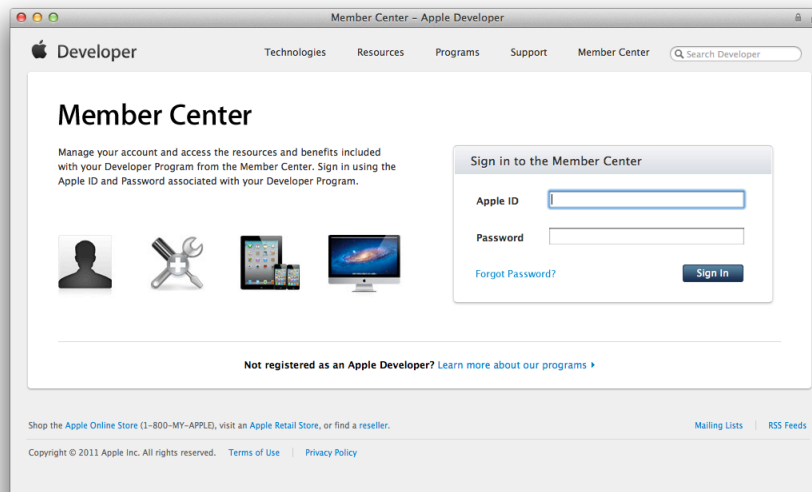
Member Center is a starting point to access other web tools. It's also where you manage your membership account, invite team members, and request technical support. If you have enrolled as a company, you can invite others to join your team and help you perform some of these tasks.

#### To sign in to Member Center

1. Go to [the Apple Developer website](#).
2. Select Member Center in the toolbar.



3. Enter your Apple ID and password, and click Sign In.



4. If you belong to multiple teams, select a team from the Teams menu and click Continue.  
Select a team that is enrolled in the developer program you want to use. For example, if you are developing an iOS app, select a team that belongs to the iOS Developer Program.

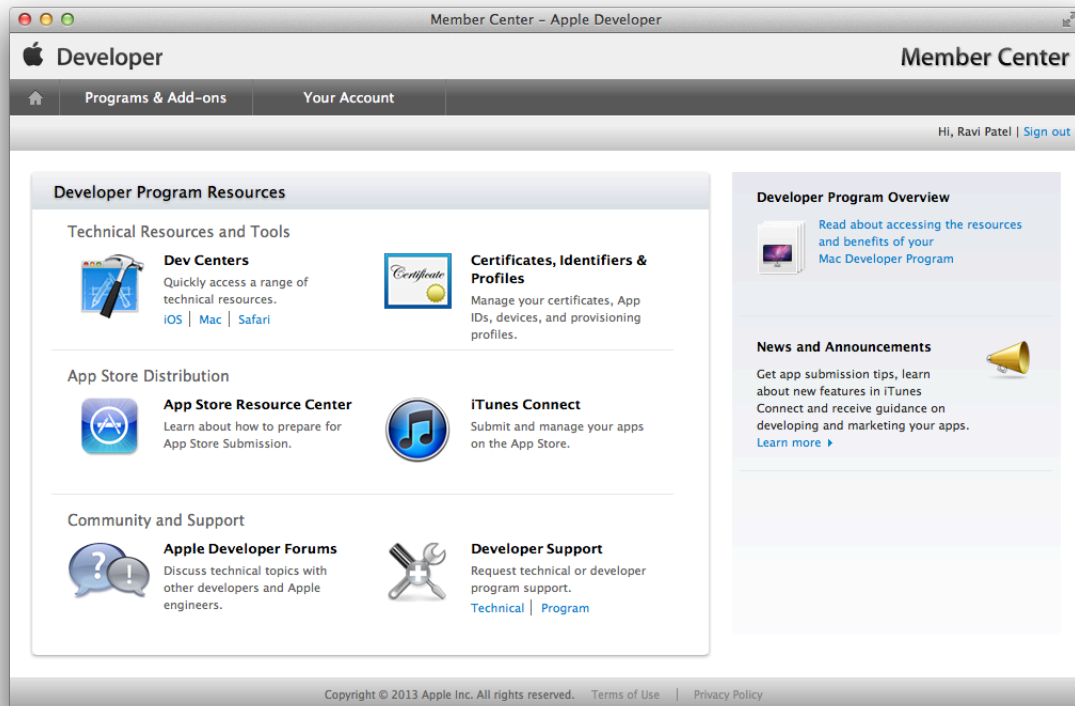
## Managing Your Certificates, Identifiers, and Profiles

Access Certificates, Identifiers & Profiles by signing in to Member Center. Here you can register App IDs and devices, enable app services, and create signing certificates and provisioning profiles for iOS and Mac apps. In the Certificates, Identifiers & Profiles area of Member Center, you can add additional developer programs.

### To access your certificates, identifiers, and profiles

1. Sign in to [Member Center](#).

2. Click the icon or text for Certificates, Identifiers & Profiles under Developer Program Resources.

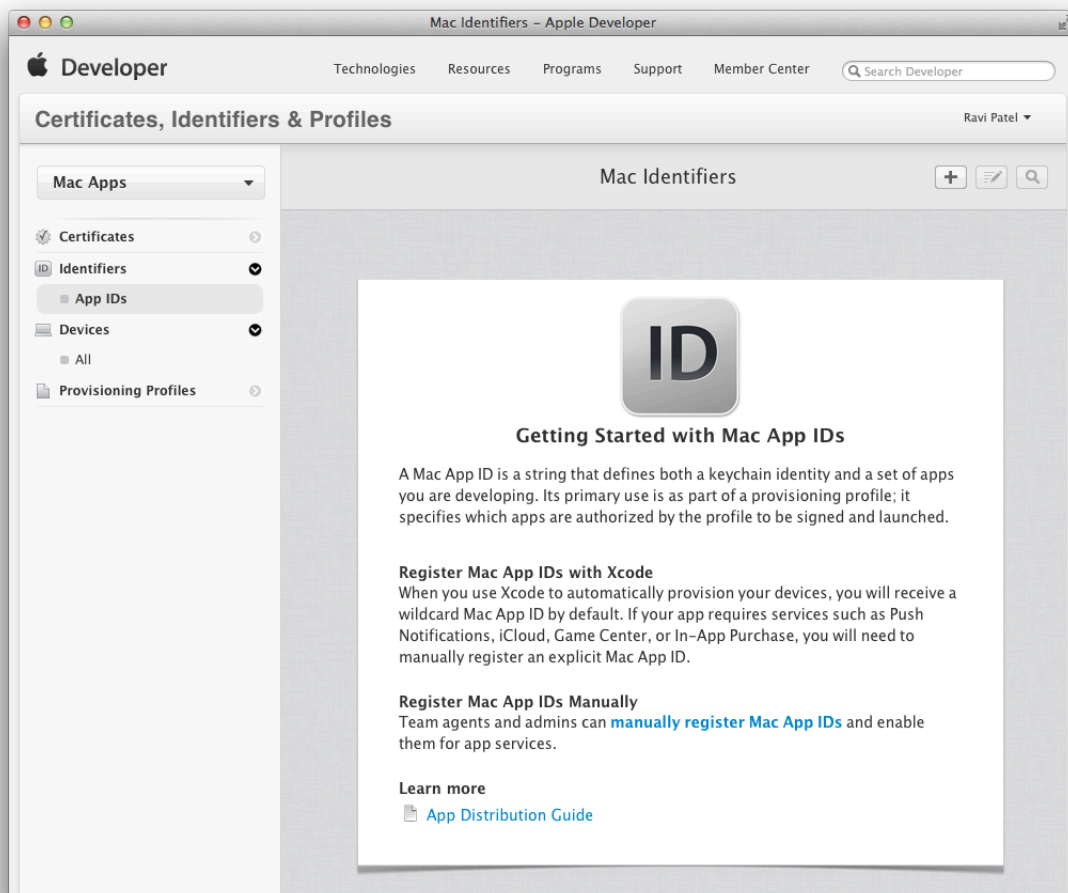


3. Click the assets below iOS Apps or Mac Apps to view them.

For example, click Identifiers under Mac Apps to view your Mac Developer Program identifiers, and click Certificates under iOS Apps to view your iOS Developer Program certificates.



The type of asset you selected will be listed. If you don't have that type of asset associated with your membership, you will see information similar this to help you get started:

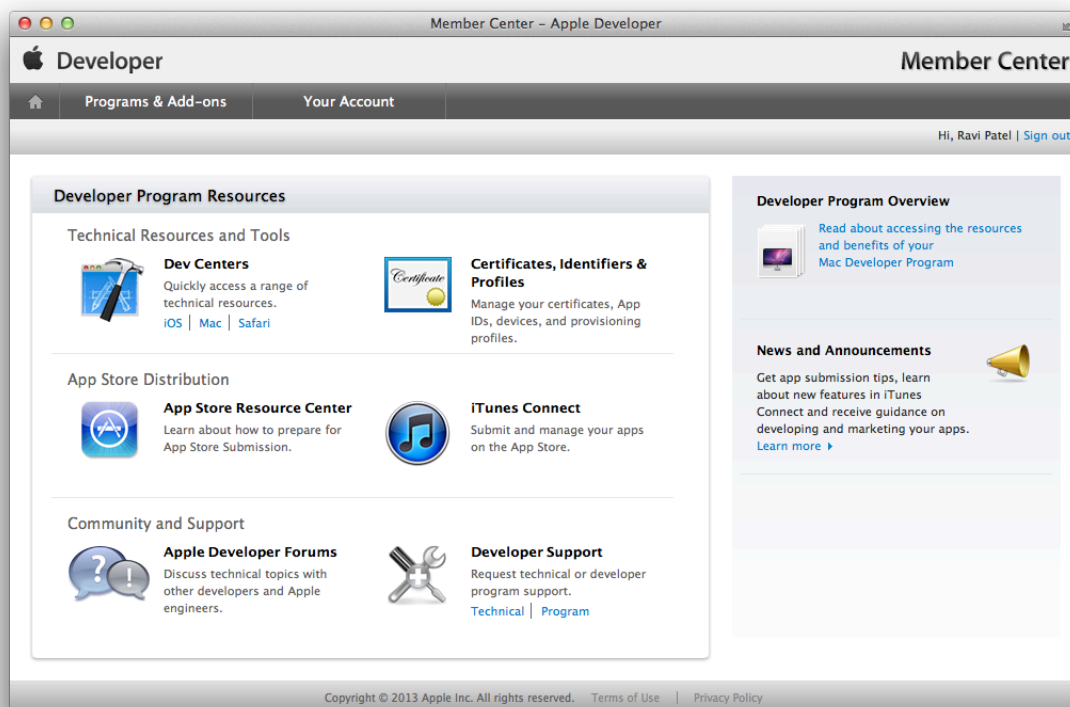


4. If you belong to multiple developer programs and want to switch to another program, select it from the iOS Apps or Mac Apps drop-down menu on the left.
5. Select a menu item from the account button in the upper-right corner to sign out or sign in using another Apple ID.

## To join another developer program

1. Sign in to [Member Center](#).

2. Click the icon or text for Certificates, Identifiers & Profiles under Developer Program Resources.



3. Click the "Join now" button under the program name you want to join.



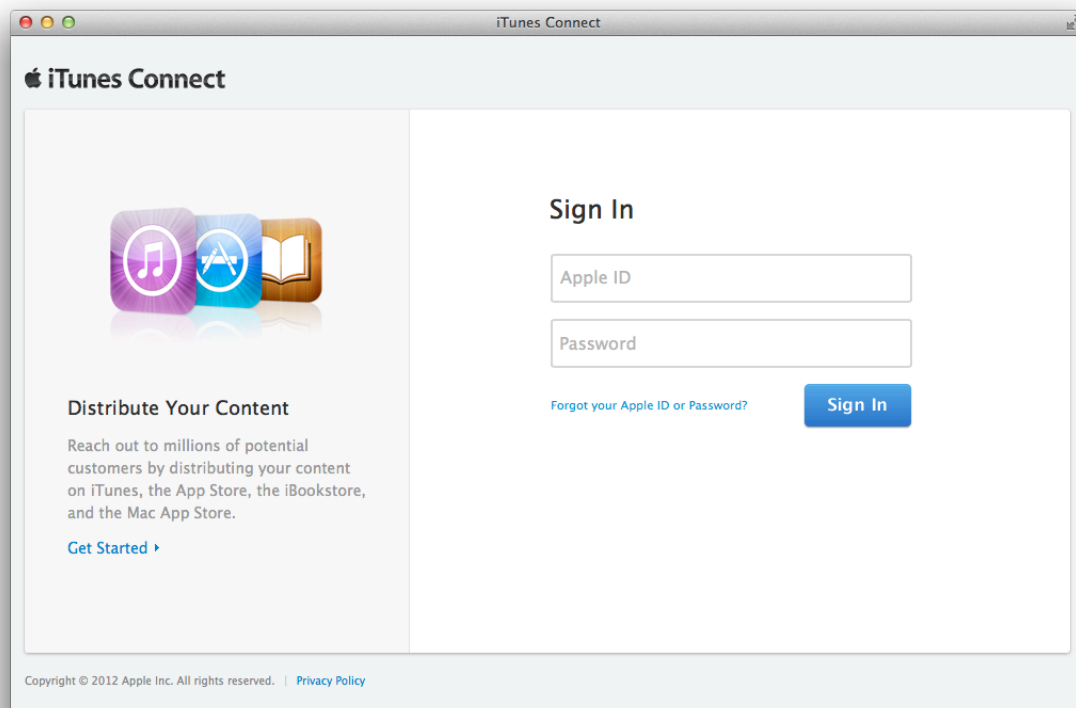
You are taken to the Apple Online Store to complete your purchase and activate your membership.

## Accessing iTunes Connect

iTunes Connect is the repository for all store-related assets, including your app binaries. You use iTunes Connect to market and distribute your app, check the status of your contracts, set up tax and banking information, get sales and finance reports, and manage your app's metadata. You can give another set of users access to your iTunes Connect account. You access iTunes Connect from Member Center or by going directly to the [iTunes Connect](#) website.

### To go to iTunes Connect from Member Center

1. Sign in to [Member Center](#).
2. Click the icon or text for iTunes Connect in the App Store Distribution section under Developer Program Resources.
3. Enter your Apple ID and password, and click Sign In.



## Bookmarking the Web Tools

You can also bookmark these links to go directly to these resources:

- [Member Center](#)
- [Certificates, Identifiers & Profiles](#)
- [iTunes Connect](#)

## Recap

You learned how to enroll in an Apple Developer Program and how to access Member Center and iTunes Connect. You'll use these resources throughout the development process to manage your account assets.

# Creating Your Signing Certificates

Code signing uses cryptographic technology to digitally sign your app and installer package. Code signing your app lets users trust that your app has been created by a source known to Apple and that your code hasn't been modified since you signed it. You must sign your code to submit your iOS and Mac apps to the store. For certain store technologies, you must sign your code during development and testing—well before submitting your app to the store. iOS and OS X verify the signature of your app before allowing it to run on devices or use certain technologies.

You use specialized signing certificates in your keychain to sign an app or installer package. For iOS apps, you need these signing certificates during development and to run your app on devices. For Mac apps, you don't need signing certificates unless you enable App Sandbox (which is required by the Mac App Store) and use store technologies such as iCloud and Game Center. Both iOS and Mac apps need special signing certificates to submit your app to the store. For Mac apps, you need a type of signing certificate to distribute your app outside of the store.

Therefore, the first step to prepare for code signing is to create the certificates specific to your platform:

- For iOS apps, you will create one certificate for each of these tasks:
  - to run an app on an iOS device and use store technologies during development
  - to distribute your app on designated devices for testing or to submit it to the store
- For Mac apps, you will create one certificate for each these tasks:
  - to use store technologies during development and testing
  - to sign your app before submitting it to the store
  - to submit an installer, containing your signed app, to the store
  - to distribute your app outside of the store
  - to distribute an installer, containing your app, outside of the store

## About Code Signing

Code signing your app allows the operating system to identify who signed your app and to verify that your app has not been modified since you signed it. Your app's executable code is protected by its signature because the signature becomes invalid if any of the executable code in the app bundle changes. Note that resources such as images and nib files are not signed; therefore, a change to these files does not invalidate the signature.



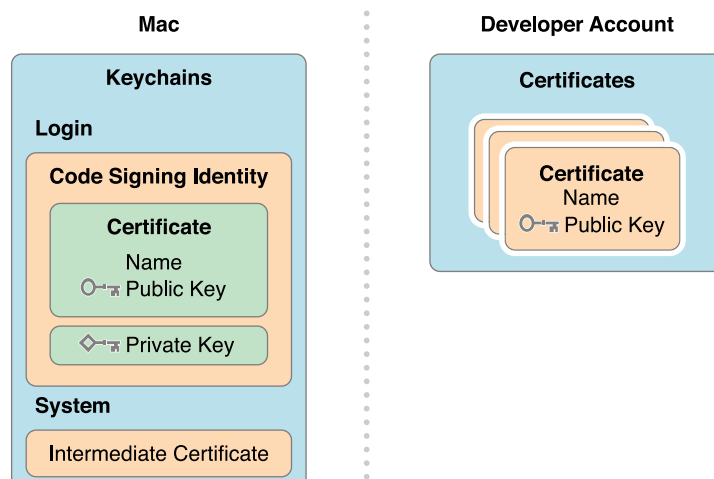
Code signing is used in combination with your App ID, provisioning profile, and entitlements (which you will learn more about later) to ensure that:

- Your app is built and signed by you or a trusted team member.
- Apps signed by you or your team run only on designated development devices.
- Apps run only on the test devices you specify.
- Your app is not using technologies you didn't add to your app.
- Only you can submit revisions of your app to the store.
- If you choose to distribute outside of the store (Mac only), the app can't be modified and distributed by someone else.

Code signing also allows your app's signature to be removed and re-signed by a trusted source. For example, you sign your app before submitting it to the store, but Apple re-signs it before distributing it to customers. Also, you can re-sign and submit a fully tested development build of your app to the store.

Xcode uses your code signing identity to sign your app during the build process. This **code signing identity** consists of a public-private key pair that is issued by Apple. The private key is stored in your keychain and used by cryptographic functions to generate the signature. The certificate contains the public key and identifies you as the owner of the key pair. The certificate is stored both in your keychain on your Mac and in your developer account. An **intermediate certificate** is also required to be in your keychain to ensure that your certificate is issued by a **certificate authority**.

To sign apps, you must have both the code signing identity and the intermediate certificate installed in your keychain. When you install Xcode, Apple's intermediate certificates are installed in your keychain for you. You create your code signing identity and sign your app using Xcode. Thereafter, you use Keychain Access and Member Center to manage your code signing identities.



**Signing certificates** are used to sign your app or installer package. A type of signing certificate used for development is used to identify you in a development provisioning profile that allows apps signed by you to launch on devices.

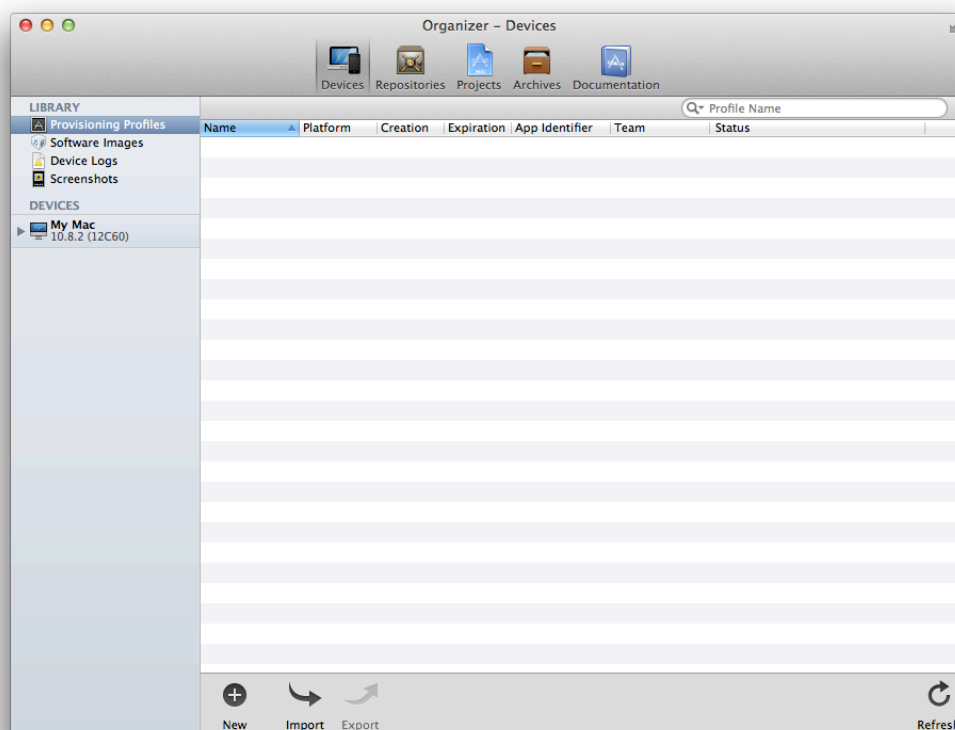
## Requesting Signing Certificates

Before you can code sign your app, you need to create your development certificate. You actually create all the types of signing certificates you'll need, throughout the project life cycle, using Xcode. Xcode requests, downloads, and installs your signing certificates for you.

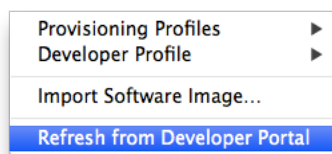
For a company, a team member requests their development certificate using Xcode but downloads and installs it later, after it is approved, as described in [“Approving Development Certificates”](#) (page 190).

### To request signing certificates

1. In Xcode, choose Window > Organizer to open the Organizer window.
2. Click Devices to display the Devices organizer.

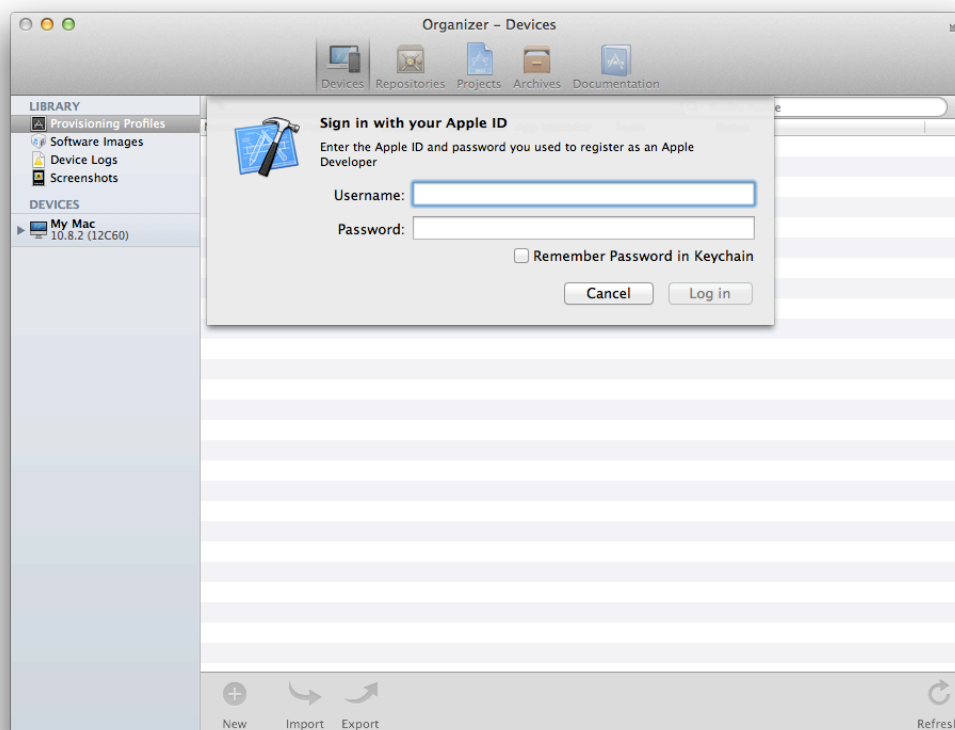


3. Select "Refresh from Developer Portal" from the Editor menu.



4. Enter your Apple ID user name and password, and click "Log in".

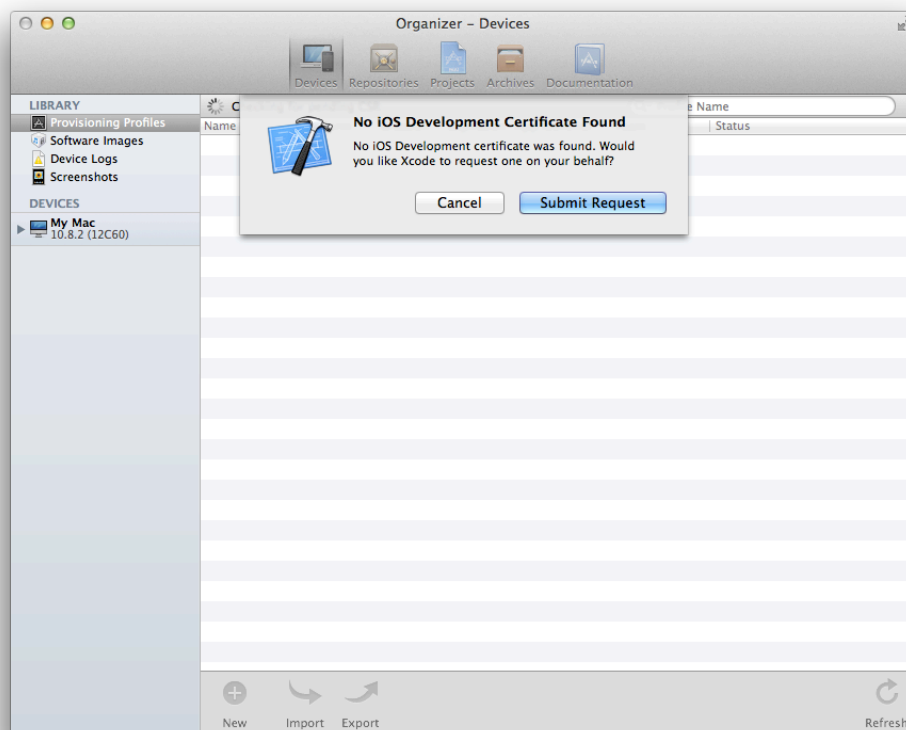
If you don't have a development certificate, Xcode offers to request development certificates on your behalf for the developer programs you are enrolled in. For example, if you are enrolled in the iOS Developer Program, Xcode offers to request an iOS Development certificate.



5. Click Submit Request for each dialog that appears.

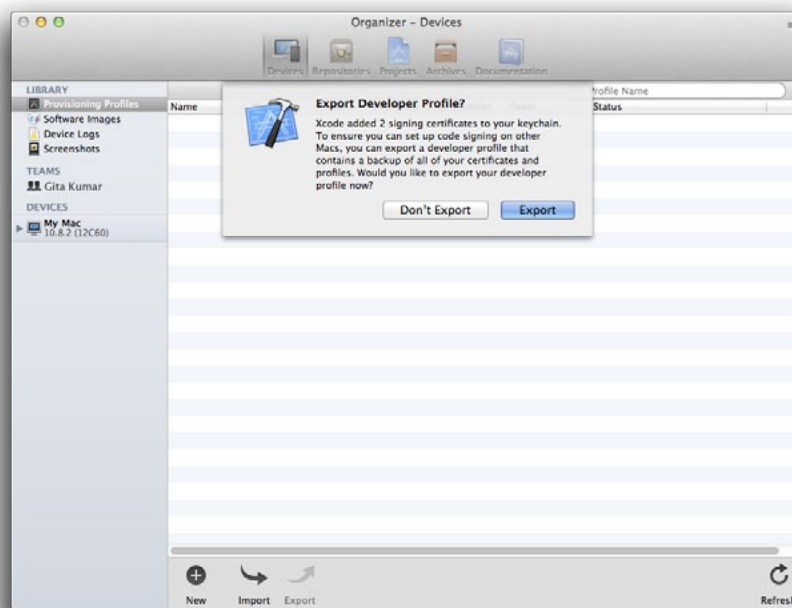
Xcode offers to request development and distribution certificates depending on the type of account you have (whether its an individual or company account) and the developer programs you belong to (iOS or Mac). If you are an individual developer, wait while Xcode submits the requests for each certificate

and they are approved. Refer to [Table 2-1](#) (page 36) for the types of certificates Xcode may request on your behalf. iOS developers need an iOS Development certificate and Mac developers need a Mac Development certificate to proceed.



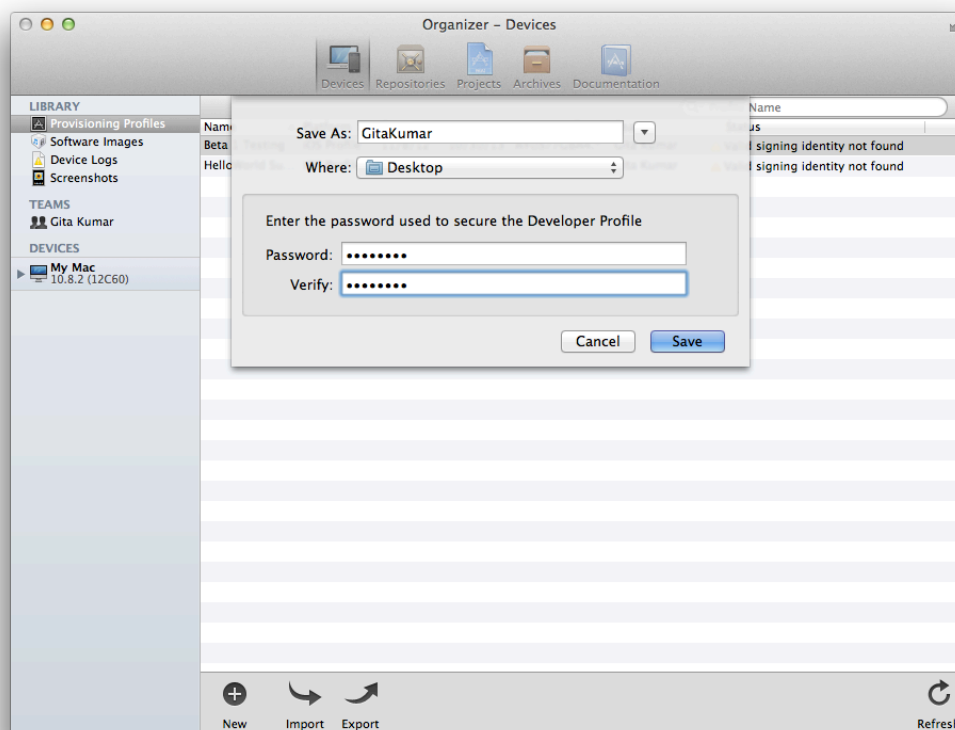
6. At the end of the refresh process, a dialog asks whether you want to export your developer profile. Click Export.

The private keys for your certificates are stored in your keychain, and the certificates, along with their public keys, are stored by Member Center. Therefore, you can't move to another Mac or user account and refresh your provisioning profiles in Xcode to restore your certificates. Instead, you should back up your certificates now, after you create them, and then import them later from another Mac or later if you are missing private keys in your keychain on this Mac.



7. Enter a filename and password, and Click Save.

For your protection, the exported file is encrypted and password protected.



If you need to export or import your developer profile later, follow the steps in [“Exporting and Importing Certificates and Provisioning Profiles”](#) (page 159).

## Verify Your Steps

Verify that your certificates are correct and ready for use. If the certificates shown in Xcode and Keychain Access don't match your certificates in Member Center, follow the instructions in [“Certificate Issues”](#) (page 208), because if the certificates in your keychain are not valid, you won't be able to sign your app.

The first time you verify your certificates, verify them in Xcode, Keychain Access, and Member Center to learn where they are located and how they appear in each tool. Later, you'll use Keychain Access for troubleshooting.

**Note:** The name of the certificate in Keychain is not the same as the certificate type that appears in Xcode and Member Center. Refer to [Table 2-1](#) (page 36) for the mapping between the certificate names and types that appear in the tools.

## Verify Using Xcode

After creating your certificates, you immediately see them displayed in Xcode.

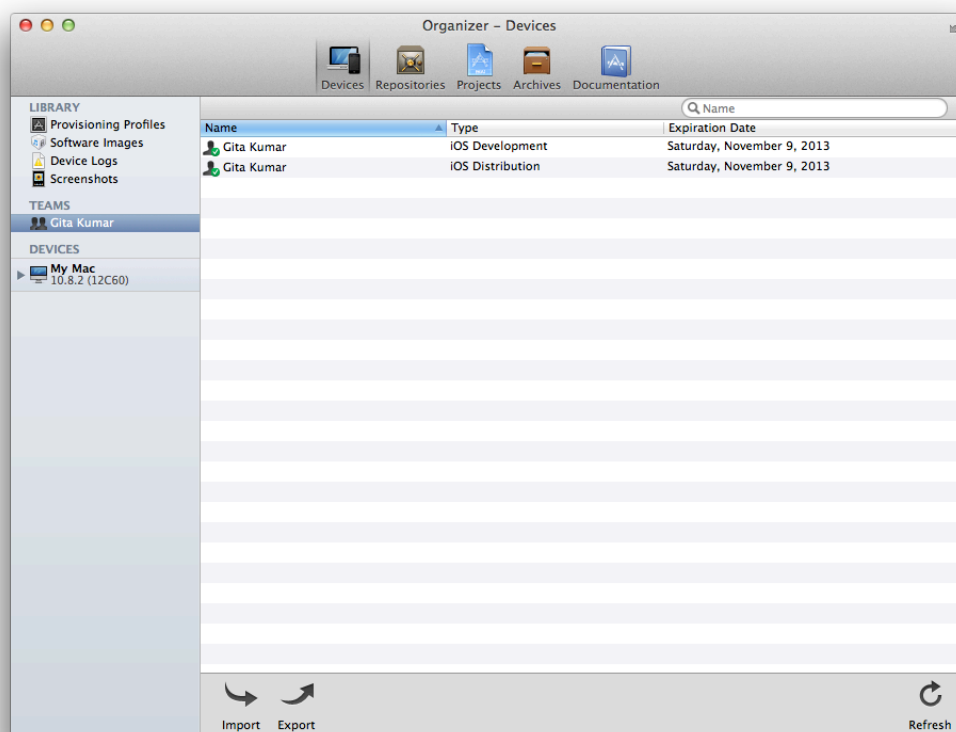
### To verify signing certificates using Xcode

1. In the Devices organizer, select your team in the Teams section.

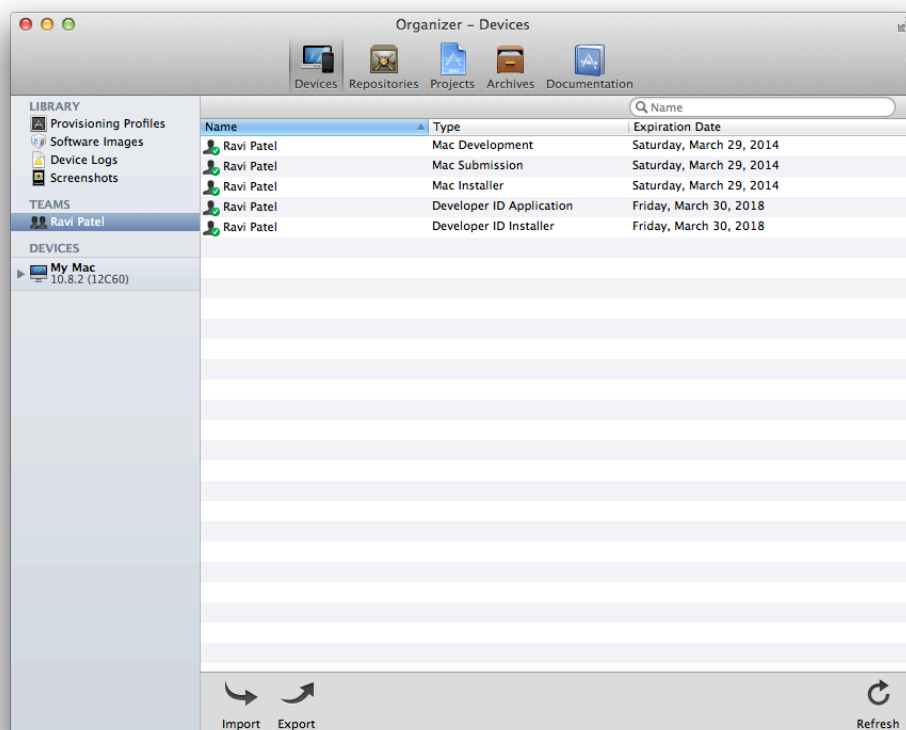
Xcode adds a Teams section to the Devices organizer that displays all your team certificates that are managed by Member Center and appear in your keychain. The list should include all the certificates you recently and previously requested.

**Note:** Individual developers are considered a one-person team.

For iOS apps, two certificates appear in the Teams section:



For Mac apps, five certificates appear in the Teams section:



2. Verify that the certificates are valid.

If the certificate is valid in your keychain, a green circle containing a checkmark badge appears next to the name of your certificate in Xcode. This checkmark means that the certificate authority (for example, Apple) that issued the intermediate certificate authorized your certificate.

## Verify Using Keychain Access

Keychain Access shows the private and public keys for each of your certificates.

### To verify signing certificates using Keychain Access

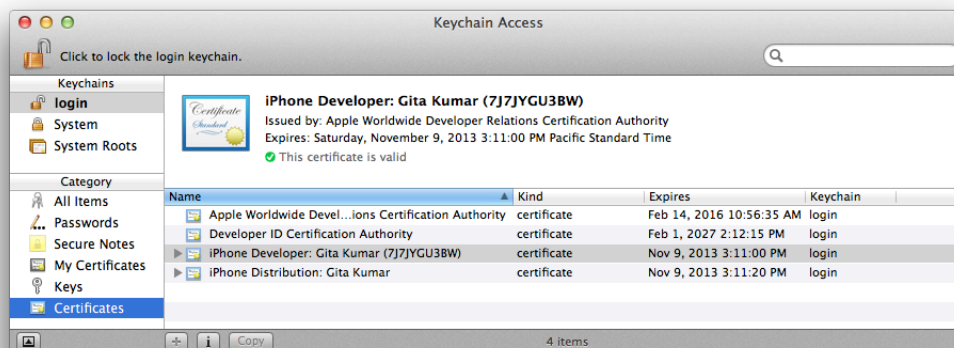
1. Launch Keychain Access located in ~/Applications/Utilities.

When you request a development or distribution certificate using Xcode, the certificate is automatically installed in your login keychain.


2. Select "login" in the Keychains section, and Certificates in the Category section.



The development certificate should appear in the Certificates category in Keychain Access. The name of the development certificate begins with the text “iPhone Developer” for the iOS Developer Program and “Mac Developer” for the Mac Developer Program, followed by your name (development certificates belong to a person).

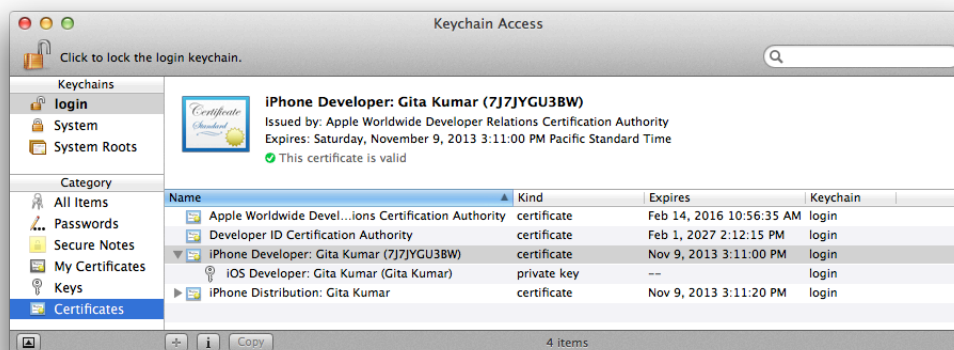


Other types of certificates also appear in the Certificates category of Keychain Access.

 **Tip:** Keep your personal keychain items in your login keychain. If your certificates don't appear in the login keychain, it may not be the default keychain. The default keychain appears in bold in the Keychains column in Keychain Access. If the default keychain is not login, select login in the Keychains column and choose File > Make Keychain “login” Default.

3. Verify that there is a disclosure triangle to the left of the certificate.

If you click the disclosure triangle next to the certificate name, your private key appears. If the disclosure triangle doesn't appear, you are missing your private key.



4. Verify that the certificates are valid.

When you select a certificate, a similar green circle containing a checkmark appears in Keychain Access above the list of certificates. The text next to the checkmark should read “This certificate is valid.”

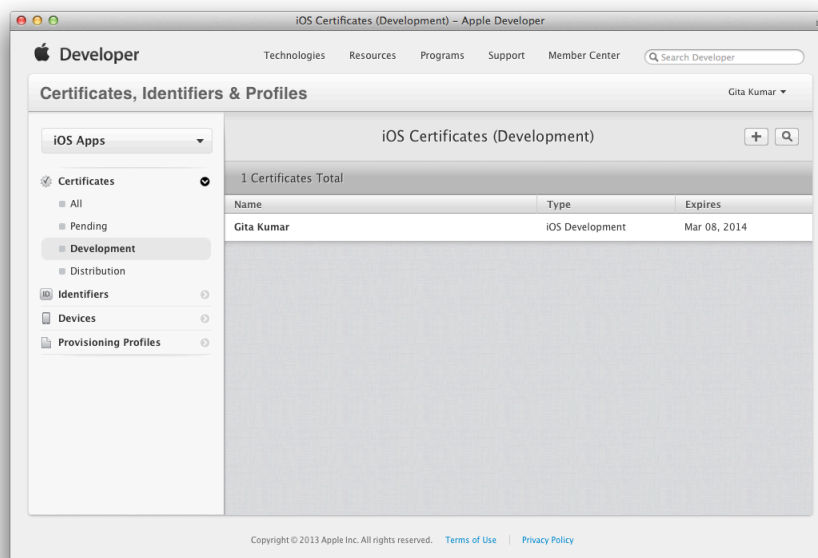
## Verify Using Member Center

[Member Center](#) should show the same certificates you see in Xcode and Keychain Access because it stores the public keys.

### To verify signing certificates using Member Center

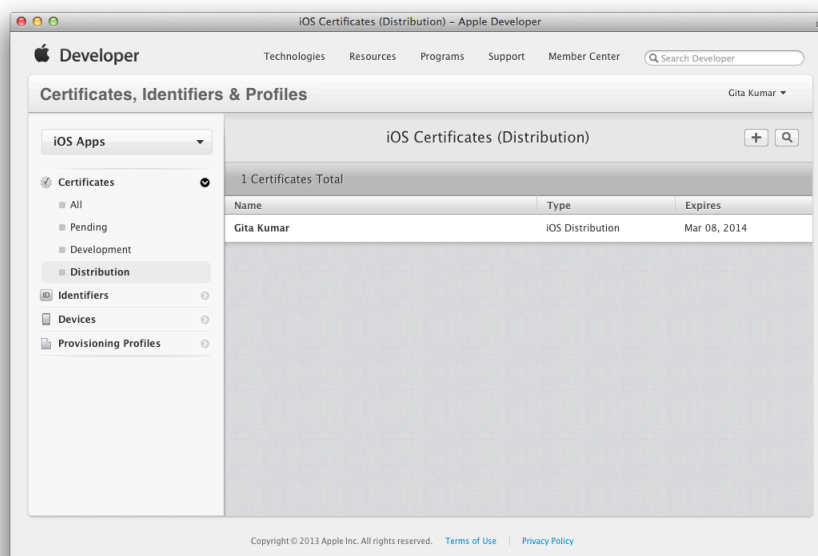
1. In [Certificates, Identifiers & Profiles](#), select Certificates.
2. In the Certificates section, select Development.

The names, types, and expiration dates of the development certificate should match the information that you view in Xcode.



3. In the Certificates section, select Distribution.

The names, types, and expiration dates of the development certificate should match the information that you view in Xcode.



## Troubleshooting

If the certificates shown in Xcode and Keychain Access don't match your certificates in Member Center, read ["Certificate Issues"](#) (page 208) for information about how to resolve the discrepancies.

## Your Signing Certificates in Depth

Your code signing identities, stored in your keychain, represent your iOS and Mac program development and distribution credentials. You should be familiar with the names of these certificates, because they appear in menus, and the types of certificates, because they appear in lists, so that you don't accidentally remove them from your keychain or Member Center.

There are different types of signing certificates for different purposes. **Development certificates** identify a person on your team and are used to run an app on a device. During development and testing, you are required to sign all iOS apps that run on devices and Mac apps that use certain technologies like iCloud and Game Center.

**Distribution certificates** identify the team and are used to submit your app to the store or for a Mac app, distribute it outside of the store. If you are a company, distribution certificates can be shared by team members who have permission to submit your app. There are multiple kinds of distribution certificates, each associated with a specific method of distribution. Different code signing identities are also used for iOS and Mac apps.

Signing certificates are issued and authorized by Apple. You must have the intermediate certificate provided by Apple installed in your system keychain to use your certificate; otherwise, it is invalid. The intermediate certificates provided by Apple and installed by Xcode are:

- **Apple Worldwide Developer Relations Certification Authority.** Used to validate development and store certificates.
- **Developer ID Certification Authority.** Used to validate a Developer ID certificate for distribution outside of the Mac App Store.

Refer to Table 2-1 for the mapping between the type of the certificate and name of the certificate as it appears in Keychain Access, and for the purpose of each.

The Devices organizer in Xcode and Member Center display the team name (or person's name) and type for each certificate. Keychain Access and the Code Signing Identity build setting pop-up menu in Xcode display the name of the certificate.

There's one Mac or iOS development certificate per team member. Therefore, development certificate names contain the person's name. All other types of certificates are owned by the team (shared by multiple team members) and therefore, contain the team name. Individual developers are a one-person team, and so your name and the team name are the same.

**Table 2-1** Certificate types and names

Certificate type	Certificate name	Description
iOS Development	iPhone Developer: <i>Team Member Name</i>	Used to sign an iOS app during development.
iOS Distribution	iPhone Distribution: <i>Team Name</i>	Used to sign an iOS app for ad hoc testing and submission to the App Store.
Mac Development	Mac Developer: <i>Team Member Name</i>	Used to sign a Mac app during development.
Mac App Distribution	3rd Party Mac Developer Application: <i>Team Name</i>	Used to sign a Mac app for submission to the Mac App Store.

Certificate type	Certificate name	Description
Mac Installer Distribution	3rd Party Mac Developer Installer: <i>Team Name</i>	Used to sign a Mac Installer Package for submission to the Mac App Store.
Developer ID Application	Developer ID Application: <i>Team Name</i>	Used to sign a Mac app for distribution outside the Mac App Store.
Developer ID Installer	Developer ID Installer: <i>Team Name</i>	Used to sign a Mac Installer Package for distribution outside the Mac App Store.

## Recap

In this chapter, you learned how to create your development and distribution signing certificates that you'll use in later chapters. You also learned how to identify the different types of certificates in Xcode, Keychain Access, and Member Center.

# Developing Apps Using the Team Provisioning Profile

All iOS apps and most Mac apps require that you use provisioning profiles during development. For iOS apps, you cannot run an app on a device (an iPhone, iPad, or iPod touch) until you provision that device for development. Similarly, you cannot run an app on a Mac that uses certain store technologies until you provision the Mac.

**Provisioning** is the process of preparing and configuring an app to launch on devices and use certain services. During development and testing, you designate the devices that can launch your app. When you submit your app to the store, you just provision your app. Provisioning iOS and Mac apps involves creating certificates, configuring App IDs, creating development and distribution provisioning profiles, and setting entitlements.

This chapter shows you how to use the team provisioning profile, which Xcode manages for you, to simplify the provisioning process during development. Xcode automatically adds your development certificate and all registered devices to the team provisioning profile. For example, you can use the team provisioning profile to run your iCloud app on all of your development devices. You can also use the team provisioning profile to run sample and test apps on devices.

For iOS apps, you can use the team provisioning profile to enable iCloud, data protection, and Passbook. For Mac apps, you can use the team provisioning profile to enable iCloud. To configure other store technologies that require a custom provisioning profile, read [“Provisioning Your App for Store Technologies”](#) (page 54).

## About the Team Provisioning Profile

The **team provisioning profile** is a development provisioning profile that Xcode manages for you. A development provisioning profile allows your app to launch on devices and use certain store technologies during development. For an individual, the team provisioning profile allows all apps signed by you to run on all of your registered devices. For a company, the team provisioning allows *any* app developed by a team, to be signed by *any* team member, and installed on *any* team device. Because the team provisioning profile isn't associated with a specific app, this profile is very useful when you want to install simple test apps on a device. When learning to provision devices, it is easier to start by using the team provisioning profile.

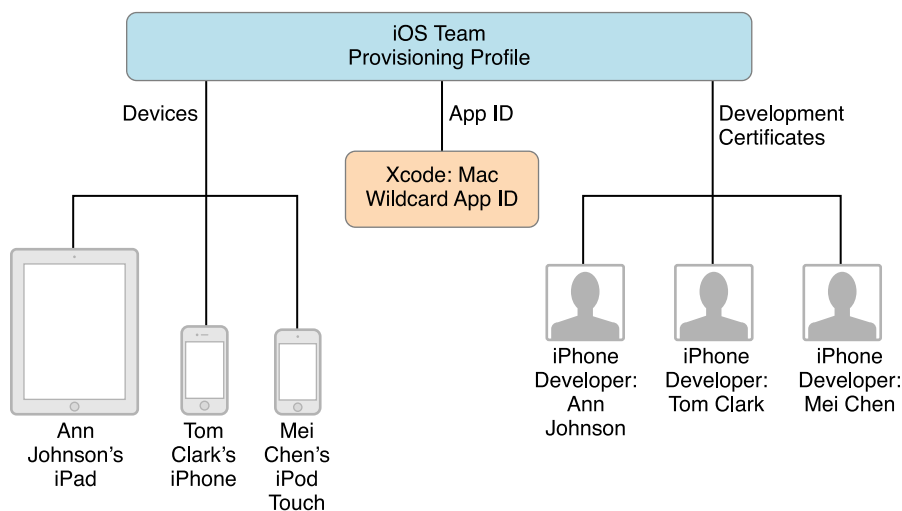
The team provisioning profile contains:

- A wildcard App ID that matches all your team's apps
- All devices associated with the team

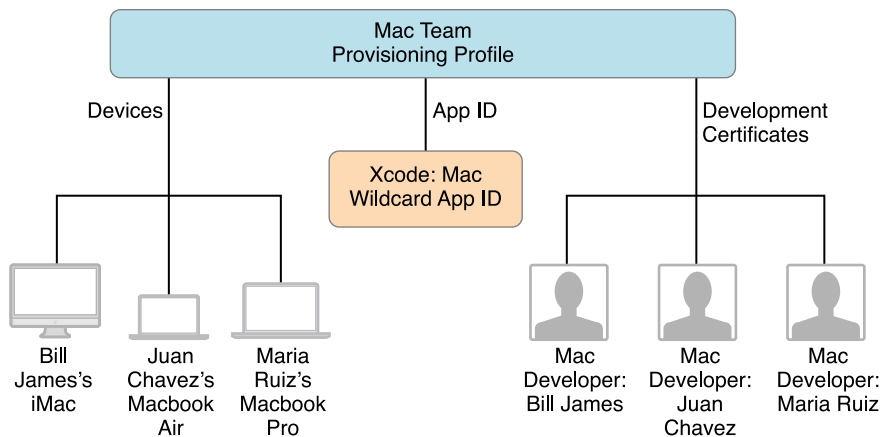
- All development certificates associated with the team

The first time you register a device, Xcode creates the team provisioning profile named *iOS Team Provisioning Profile* or *Mac Team Provisioning Profile* using a wildcard App ID it also creates. Thereafter, Xcode updates the team provisioning profile whenever you register a device, create a development certificate, or refresh provisioning profiles using Xcode. (Changes you make to your team assets using Member Center don't automatically update the team provisioning profile.) Xcode adds all of the devices and development certificates from all team members to this profile. The team provisioning profile can be used for iCloud but not for other store technologies that require an explicit App ID.

This diagram shows an iOS Team Provisioning Profile for a company with three team members.



This diagram shows a similar Mac Team Provisioning Profile for a Mac company.



## Adding Devices to Your Team Provisioning Profile

After creating the signing certificates, the next step is to register your development devices. The first time you register a device, Xcode creates the team provisioning profile containing a wildcard App ID it also creates, and all the iOS Development or Mac Development certificates in your account.

You can register development devices using either Xcode or Member Center. However, if you register your device using Xcode, Xcode updates the team provisioning profile and provisions your device at the same time. The **device is provisioned** when a provisioning profile that contains the device information is installed on the device.

## Registering and Provisioning an iOS Device Using Xcode

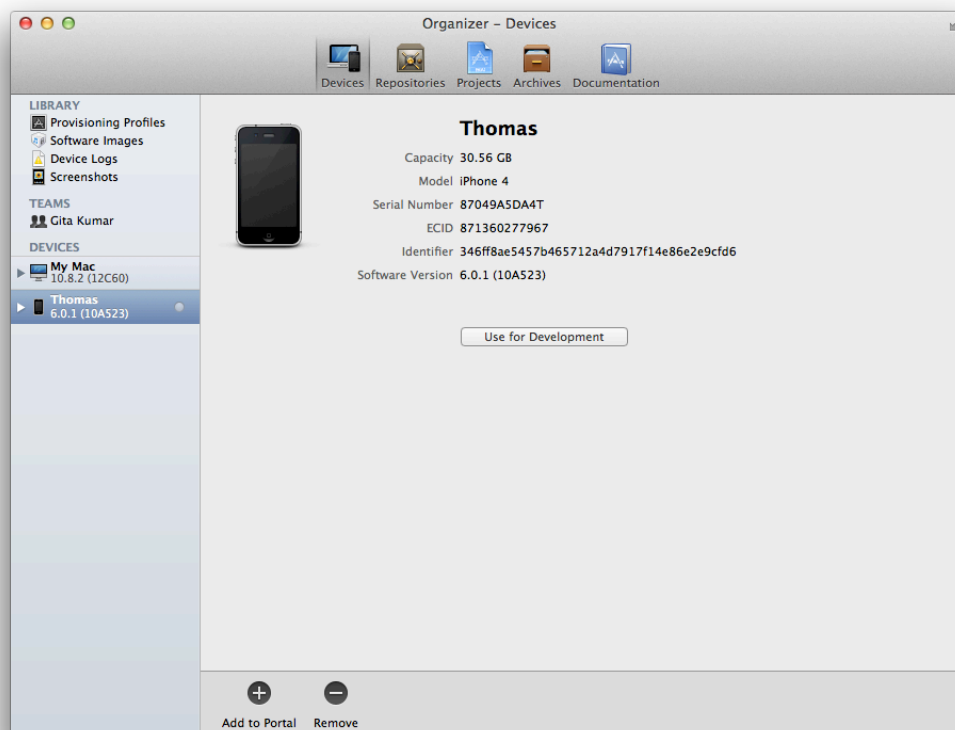
For iOS apps, you register and provision a device in a single operation.

### To provision your iOS device for development using Xcode

1. In Xcode, choose **Window > Organizer**, and click **Devices** to display the **Devices** organizer.
2. Connect your device to your Mac.
3. In the **Devices** section, select your iOS device.
4. Click the **“Use for Development”** or **“Add to Portal”** button.



If the device was previously used for development, the “Use for Development” button does not appear. If this happens, click “Add to Portal” at the bottom of the window instead.



5. Enter your user name and password, and click “Log in.”

Xcode also installs the team provisioning profile on your device, so you can immediately use it for development.

## Registering and Provisioning a Mac Using Xcode

If your Mac app uses specialized technologies such as iCloud and Game Center, you need to provision it for development, too. After you register your Mac and add it to a provisioning profile, you install the provisioning profile on the Mac.

### To provision your Mac for development using Xcode

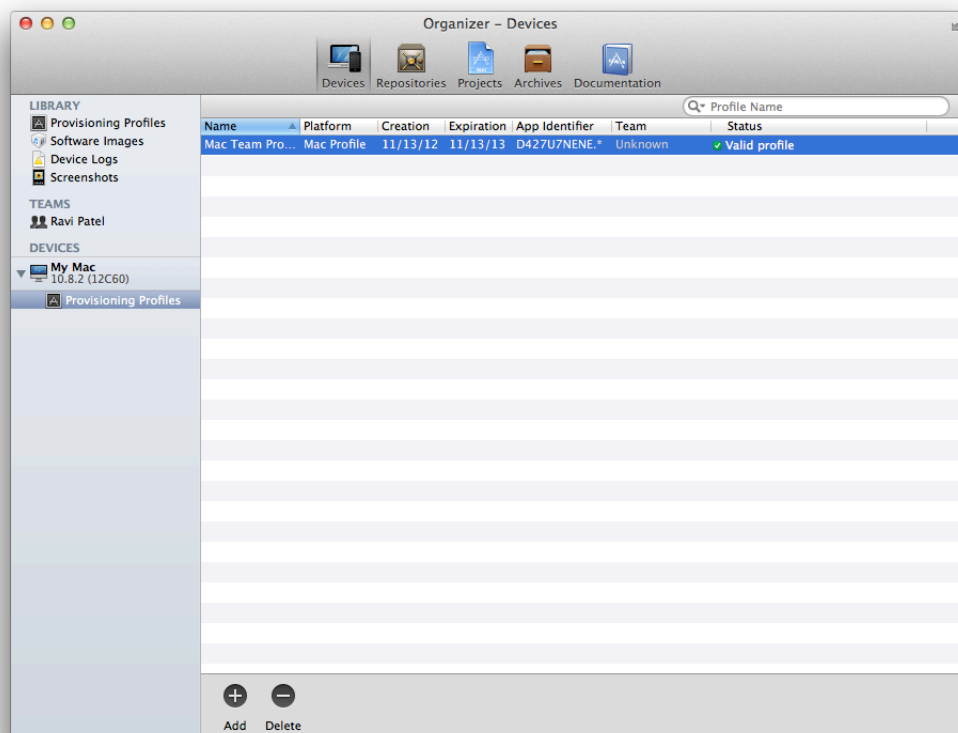
1. In Xcode, choose Window > Organizer, and click Devices to display the Devices organizer.
2. In the Devices section, select your Mac.

3. Click the “Add to Portal” button at the bottom of the window.



4. Enter your user name and password, and click “Log in.”
5. Click Provisioning Profiles under Library.
6. Drag the Mac Team Provisioning Profile to your Mac under Devices.

In the Devices organizer, the team provisioning profile appears in the Provisioning Profiles section on your Mac.



## Verify Your Steps

The first time that you use the team provisioning profile, you should verify that the device ID has been added correctly and that the team provisioning profile is installed on your device.

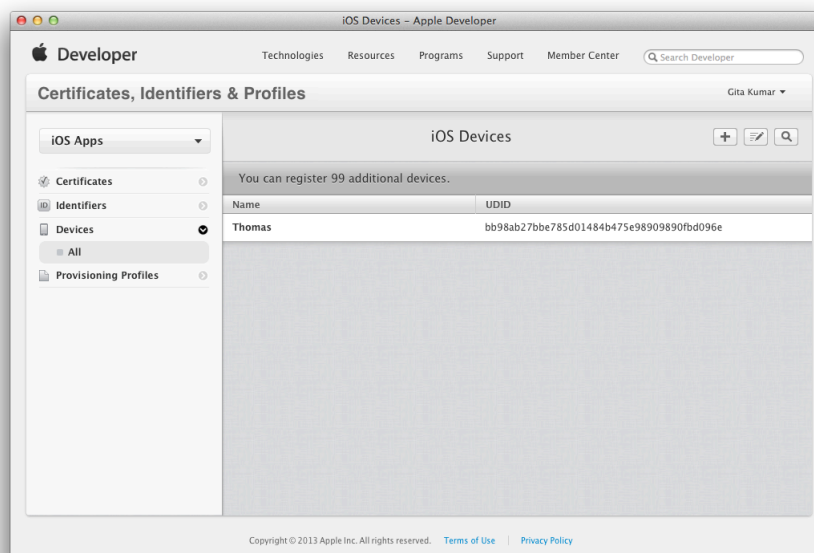
### Verify That Your Device Was Registered and Added to the Team Provisioning Profile

Use [Member Center](#) to view your registered devices and details about the team provisioning profile.

#### To verify that your device is registered

1. In [Certificates, Identifiers & Profiles](#), select Devices.
2. Under Devices, select All.

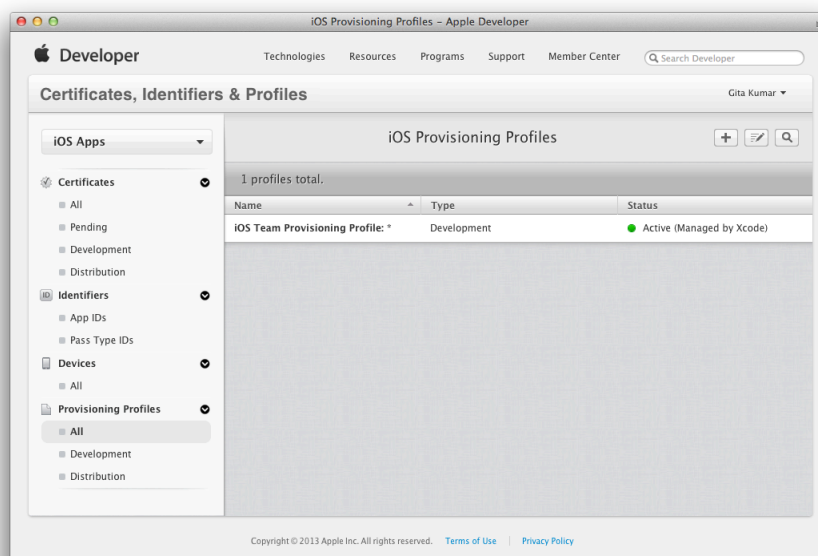
The device you registered should appear enabled in the list. Enabled devices appear in black text and disabled devices appear in gray text.



### To verify that your device is added to the team provisioning profile

1. In [Certificates, Identifiers & Profiles](#), select Provisioning Profiles.
2. Under the Provisioning Profiles section, select All.

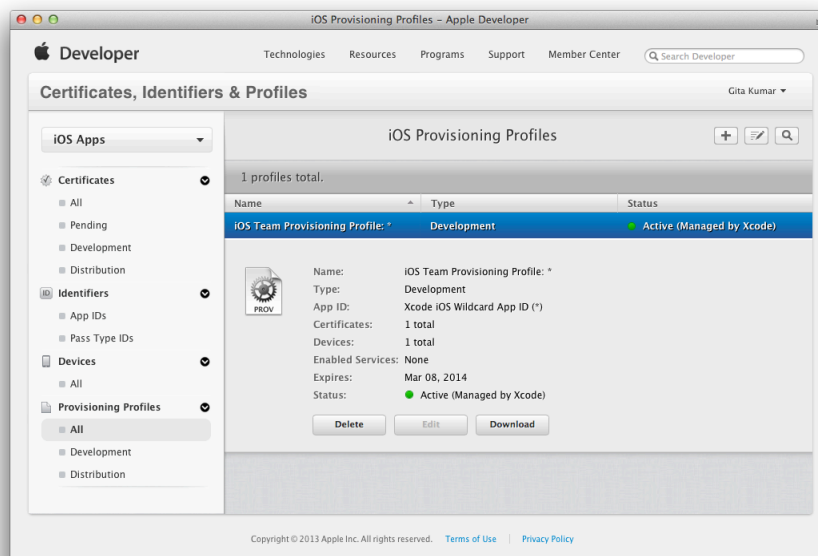
The team provisioning profile, starting with the text “iOS Team Provisioning Profile” or “Mac Team Provisioning Profile,” should appear under iOS Provisioning Profiles or Mac Provisioning Profiles.



3. Click the team provisioning profile to view its details.

The team provisioning profile contains an App ID—either for iOS apps (Xcode iOS Wildcard App ID) or for Mac apps (Xcode Mac Wildcard App ID). The screenshot below shows an iOS Provisioning Profile.

Listed beneath the App ID is the number of development certificates and devices contained in the provisioning profile. The values should equal the total number of iOS Development or Mac Development certificates and registered devices contained in your account. If you are an individual, you should have only one development certificate.



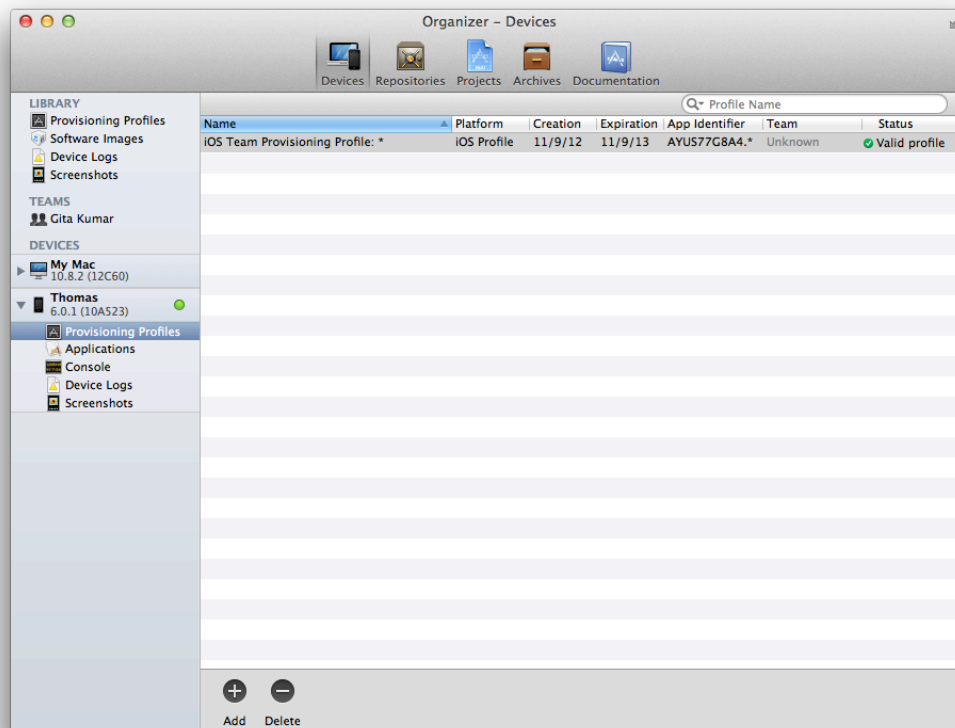
## Verify That Your Team Provisioning Profile Is Installed on Your Device

Use Xcode to examine the provisioning profiles on your device.

### To verify that a provisioning profile is installed on your device

1. In Xcode, choose Window > Organizer, and click Devices to display the Devices organizer.
2. Click the disclosure triangle next to your device under Devices.
3. Select Provisioning Profiles under your device.

Your provisioning profile should be listed in the detail area and the status should be “Valid profile.”

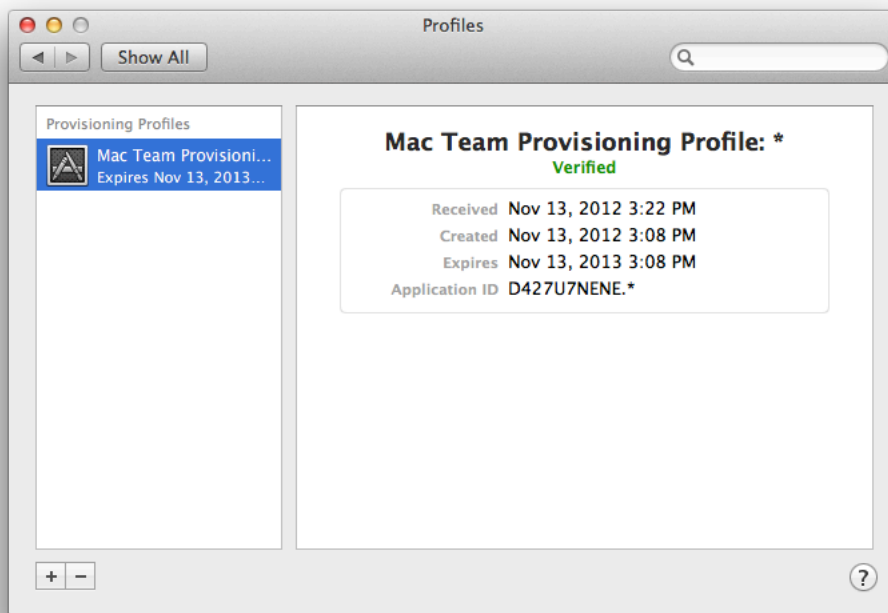


For Mac apps, optionally use System Preferences to view all of the installed provisioning profiles.

### To verify that your provisioning profile is installed on your Mac using System Preferences

1. Launch System Preferences.
2. Select Profiles under System.  
If you have one or more profiles installed, a Profiles preference appears; otherwise, it does not.
3. Select a provisioning profile under Provisioning Profiles.

Verify that your provisioning profile is valid. The provisioning profile is valid if it has not expired and the text “Verified” appears under the name of the provisioning profile.



## Troubleshooting

There are several reasons why Xcode may not create the team provisioning profile. Follow the steps for each situation below and then re-create the team provisioning profile.

- If you have a team provisioning profile but for some reason it is invalid, delete and re-create it. Follow the steps in [“Removing Provisioning Profiles from Your Team”](#) (page 179) to delete the team provisioning profile.
- If your device was previously registered but is disabled, enable the device using Member Center and then refresh your developer profile using Xcode to create the team provisioning profile.
- If your device was previously registered and is enabled, refresh your developer profile using Xcode to create the team provisioning profile.

To create the team provisioning profile using assets already in your account, open the Devices organizer in Xcode and select “Refresh from Developer Portal” from the Editor menu.



## Code Signing Your App Using the Team Provisioning Profile

To run an app on an iOS device and enable store technologies, the app needs to be code signed and provisioned. For example, using the team provisioning profile, you can enable iCloud. When you select your development certificate—contained in the team provisioning profile—as the code signing identity, the app is code signed and the team provisioning profile is embedded in your app’s bundle. The embedded team provisioning profile allows your app to launch and use iCloud.

When you build the app, you code sign it with the signing certificate contained in the provisioning profile you want to use. The possible values for the Code Signing Identity build setting pop-up menu are:

- **Don’t Code Sign.** Don’t sign your app. Selecting this option disables entitlements including sandboxing.
- **Automatic Profile Selector.** Selects an identity that matches your developer or distribution certificate name.
- **Identities without Provisioning Profiles.** Selects a code signing identity that is not in a provisioning profile.
- **Other.** Selects a specific code signing identity. The code signing identities in your default keychain are listed by the name. Expired or otherwise invalid identities are dimmed and cannot be chosen.

A menu item appears in the Code Signing Identity build setting pop-up menu for each provisioning profile to which your development certificate belongs. The default setting is the platform-specific development certificate that appears in the Automatic Profile Selector menu item, which matches your development certificate in the team provisioning profile. Refer to [Table 2-1](#) (page 36) for a description of each type of certificate that may appear in this menu.

Before you begin, decide whether to set the Code Signing Identity build setting at the project or target level. For a single target, you can set this build setting at either the project or target level as long as you are consistent. For multiple targets that use the same code signing identity, you should set this build setting at the project level. For multiple targets that use different code signing identities, you must set this build setting for each individual target. For example, choosing the project level ensures that any helper apps inside of your project are code signed as well as the main app.

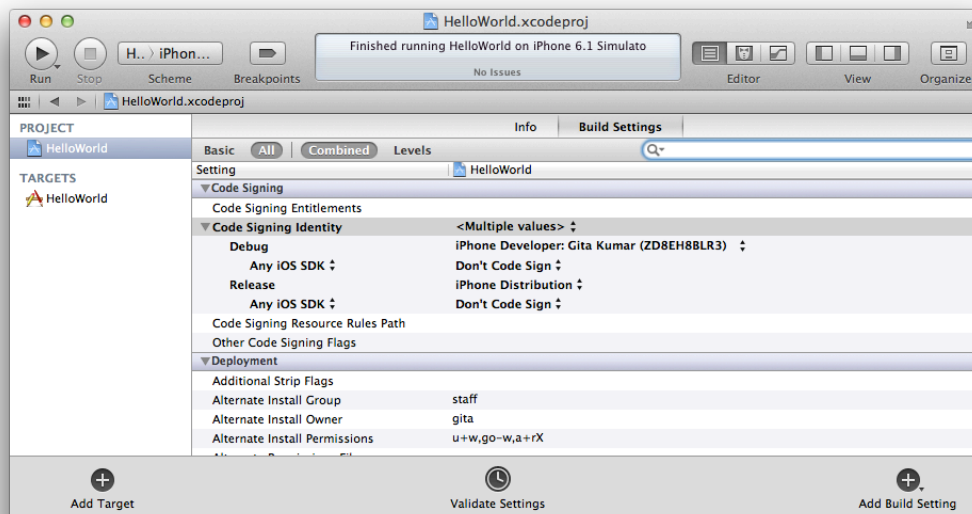
Set the Code Signing Identity to your development certificate contained in the team provisioning profile.

### To set the code signing identity to your development certificate

1. In the Xcode project editor, select the target.

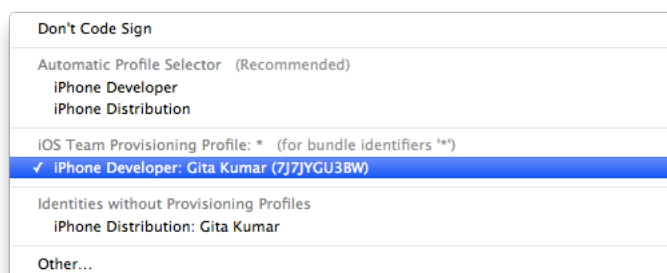
**Important:** If you want to sign multiple targets with the same code signing identity, select the project, not a target.

2. Select the Build Settings tab.

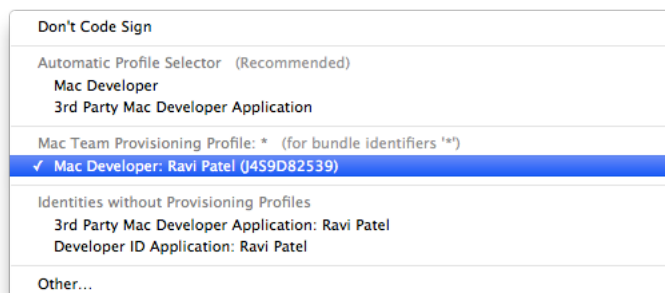


3. Click All.
4. Type Code Signing in the search field in the Build Settings pane of the project editor.
5. From the Code Signing Identity pop-up menu—in your team provisioning profile section—choose your development certificate.

For iOS apps, select the certificate in the iOS Team Provisioning Profile menu item that begins with the text “iPhone Developer:” followed by your name.



For Mac apps, select the certificate in the Mac Team Provisioning Profile menu item that begins with the text “Mac Developer:” followed by your name.



You must build your app to actually code sign it. You can build and run your Mac app by simply clicking the Run button. For an iOS app, follow the steps in [“Launching Your iOS App on the Device”](#) (page 52), to sign your app and launch it on a device.

The first time you sign your app, a dialog appears asking if you want to allow the `codesign` command-line utility to sign your app using the private key in your login keychain. When you see this dialog, click Always Allow. If you click Allow, the dialog appears every time you build and run your app. If a dialog appears when you run the app asking for a Developer Tools Access login, enter an account name and password of a user in this group—for example, a system administrator—and click Continue.

To learn more about Apple’s code signing technology, read *Code Signing Guide*.

## Troubleshooting

If the team provisioning profile doesn’t appear in the Code Signing Identity menu, choose a certificate under Automatic Profile Selector. Then try to choose your development certificate under the team provisioning profile again.

If a code signing error occurs when you build the app, verify that the Code Signing Identity build setting is correct. Also, check whether the Code Signing Identity build setting is set at the project or target level (target settings override project settings). To troubleshoot the Code Signing Identity build setting, read [“Build and Code Signing Issues”](#) (page 210).

## Launching Your iOS App on the Device

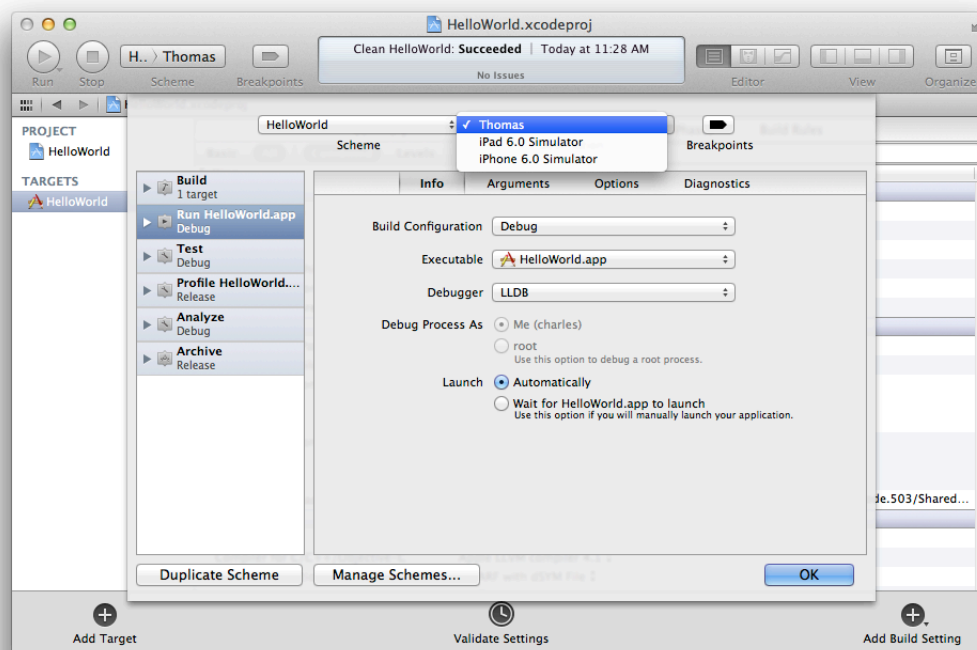
After provisioning your device for development, you can tell Xcode to launch the app on the device. You do this by changing the *run destination* setting in the Scheme pop-up menu before you build the app. When you connect an iOS device with a valid provisioning profile to your Mac, the name of the device appears as an option in the destination Scheme pop-up menu.

You should also test your app in iOS Simulator using Instruments and other tools before distributing your app. Read *iOS Simulator User Guide* for details on how to use iOS Simulator to test your app.

### To launch an iOS app on a device

1. Connect the device to your Mac.
2. Choose Product > Scheme > Edit Scheme to open the scheme editor.
3. Select your device from the Destination pop-up menu.

When you connect an iOS device with a valid provisioning profile into your Mac, its name appears as an option in the destination Scheme pop-up menu.



4. Click OK to close the scheme editor.
5. Click Run.

If a prompt appears asking whether `codesign` can sign the app using a key in your keychain, click **Always Allow**.

## Troubleshooting

There are several reasons why your app may not run on a device. Often the configuration of your project doesn't match the configuration of your device. Assuming that you followed and verified the steps in this chapter, check the Xcode project settings. To verify that the iOS Deployment Target is less than or equal to the iOS software version installed on your device, read "[Setting the Deployment Target](#)" (page 105).

## Recap

In this chapter, you learned how to use the team provisioning profile, which Xcode creates and manages for you, to provision your devices for development and code sign your app. You also learned how to launch your app on an iOS device.

# Provisioning Your App for Store Technologies

Certain technologies—such as iCloud, Game Center, and In-App Purchase—are available only to apps distributed through the store. But simply submitting your app to the store doesn't enable these technologies. You need to provision your app to use these technologies. The first step is to configure App IDs and provisioning profiles to enable the technologies you want to use. Later, you'll complete the configuration of some technologies using Xcode and iTunes Connect.

The store technologies available for both iOS and Mac apps are:

- **Apple Push Notification service (APNs).** Allows an app that is not running in the foreground to notify the user that it has information for the user. (This document and some tools refer to this technology as *push notifications*.)
- **Game Center.** Apple's social gaming network that allows players to connect to the service and exchange information with other players.
- **iCloud.** Allows you to share the user's data among multiple instances of your app running on different iOS and Mac OS X devices.
- **In-App Purchase.** Embeds items to purchase directly into your app by allowing you to connect to the store and securely process payments from the user.

The additional store technologies available for iOS apps are:

- **Data protection.** Adds a level of security to files stored on-disk by your app.
- **Newsstand.** Allows you to deliver magazine and newspaper subscription content to users.
- **Passbook.** Presents digital representations of information—such as a coupon, ticket for a show, or boarding pass—that allow users to redeem a real-world product or service.
- **Routing apps.** Allows apps that can display point-to-point directions to make those directions available to Maps and other apps.

Follow these steps to provision your app to use these technologies:

1. If necessary, create an App ID.
2. Enable store technologies for your App ID.
3. Use the team provisioning profile or create your own development provisioning profile containing your App ID.

4. Set your bundle ID to match your App ID.
5. Provision your development devices.
6. Build and sign your app using your development provisioning profile.

After performing these steps, follow any additional technology-specific steps described in [“Configuring Store Technologies in Xcode and iTunes Connect”](#) (page 79).

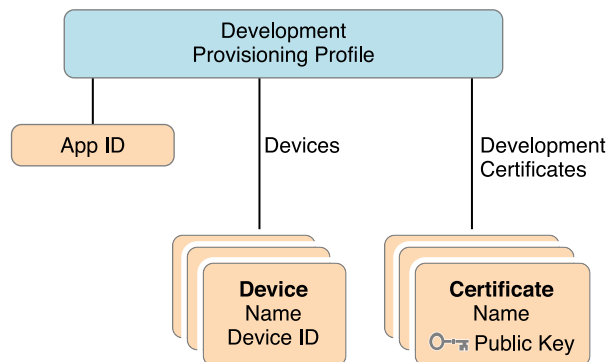
The goal of this chapter is to create a development provisioning profile that is customized for your app.

## About Development Provisioning Profiles

You use a **development provisioning profile** to authorize your app to launch on devices and use certain store technologies during development. It is one of the two types of provisioning profiles you create in the lifetime of your app. Later, you’ll use the other type of provisioning profile, a **distribution provisioning profile**, for testing and submitting your app to the store.

In general, your app needs to be authorized by Apple to run on an iOS device. Your app also needs to be authorized by Apple to use certain iOS and OS X technologies. You enable and configure technologies for your app by setting **entitlements**. Some entitlements are enabled for an App ID (which identifies a set of apps created by your team) and others are set in the Xcode project. You also enter your App ID into iTunes Connect for additional security checks. A **provisioning profile** contains the App ID and other assets required for that type of profile.

A development provisioning profile authorizes a specific set of apps to run, devices to run those apps, and development certificates to sign those apps. Consequently, a development provisioning profile is comprised of these assets: a single App ID, a set of devices, and a set of development certificates. Your development provisioning profiles reside in Member Center, but since Xcode downloads them, you can install them onto devices.



Each Mac and iOS device in a provisioning profile is identified by its unique **device ID (UDID)**. The devices you register and add to a development provisioning profile are stored by Member Center. Each individual or company can register up to 100 devices for development and testing.

If you use certain technologies that require an explicit App ID (that is, an App ID that matches a single bundle ID), you will create a custom development provisioning profile that contains the explicit App ID. (Later, you'll create a distribution provisioning profile containing the App ID.)

## Before You Begin

Before you begin, decide on what type of App ID and provisioning profile you need for the technologies you want to use. It's convenient to use the team provisioning profile that Xcode manages for you, as described in [“Developing Apps Using the Team Provisioning Profile”](#) (page 38), but you can't use the team provisioning profile for all technologies.

Table 4-1 shows what needs to be configured for each of the store technologies. You configure one App ID to enable all the technologies you want to use for one or more apps. If you don't need an explicit App ID, you can configure the Xcode wildcard App ID and use the team provisioning profile. Otherwise, you need to create an explicit App ID and corresponding development and distribution provisioning profiles containing the App ID that you manage yourself.

Table 4-1 Tasks you perform to configure store technologies

	Create Explicit App ID	Enable App ID	Set Entitlements	Edit Info.plist	Configure iTunes Connect
APNs	✓	✓			
Game Center	✓	✓			✓
iCloud		✓	✓		
In-App Purchase	✓	✓			✓
Data protection		✓			
Newsstand				✓	✓
Passbook		✓			
Routing apps				✓	✓



For example, if your iOS app uses iCloud, data protection, Newsstand, Passbook, or routing apps, you can use the team provisioning profile. For Mac apps, if your app uses iCloud only, you can use the team provisioning profile. However, if you add push notifications, In-App Purchase, or Game Center to your app, you need to create an explicit App ID and custom provisioning profiles for both development and distribution. If you develop multiple apps that use different sets of technologies, you need specialized App IDs and provisioning profiles, too.

Table 4-1 also shows which technologies require additional configuration (setting entitlements and editing the information property list) in the Xcode project. Some technologies also require configuration in iTunes Connect. “[Configuring Store Technologies in Xcode and iTunes Connect](#)” (page 79) describes the additional steps you perform after configuring App IDs and provisioning profiles.

Decide on a naming convention now for the App IDs and provisioning profiles you’ll create, especially if you develop multiple apps, to help you identify them later in Member Center, Xcode, and on devices.

## Creating App IDs

You’ll create an App ID if you can’t use the Xcode wildcard App ID and team provisioning profile for the technologies you want to use. When you create an App ID, you specify whether it’s an explicit or wildcard App ID and optionally, enable technologies. For iOS apps, Game Center and In-App Purchase are enabled by default for an explicit App ID. For Mac apps, In-App Purchase is enabled by default for an explicit App ID. (You can also enable technologies after you create an App ID.)

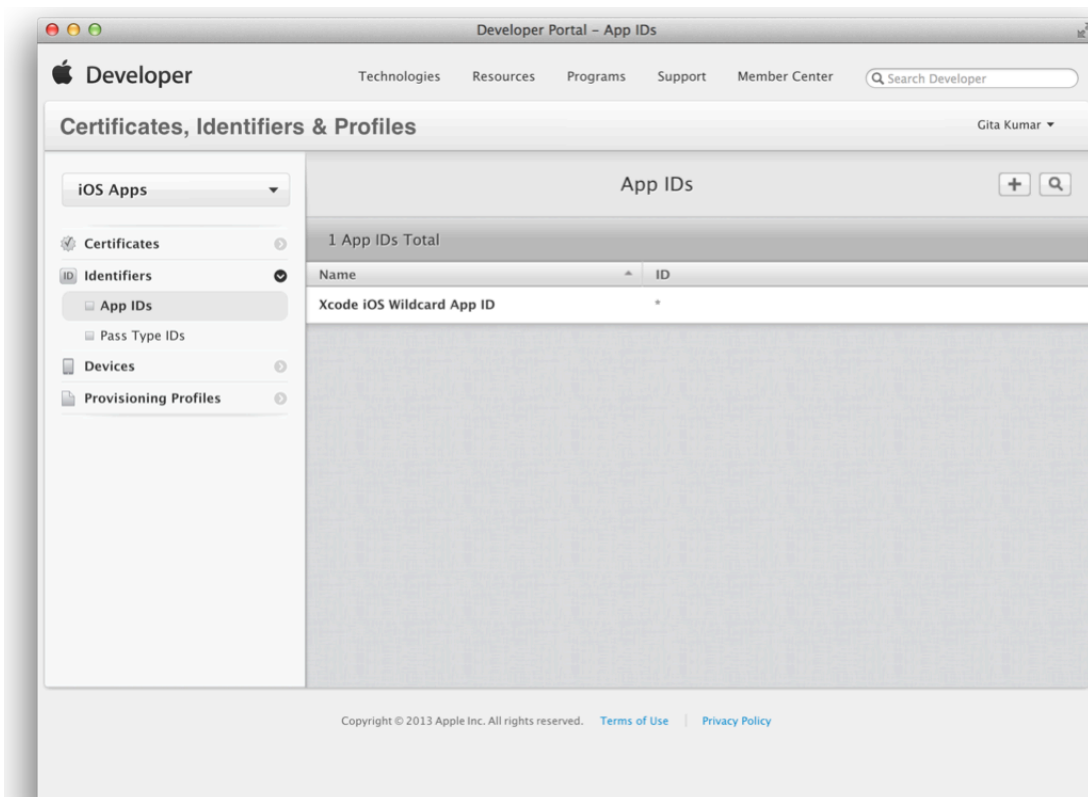
## Registering an App ID

The steps to enable each technology are similar with a few variations for specific technologies. For iOS apps, you select a level of protection when you enable data protection.

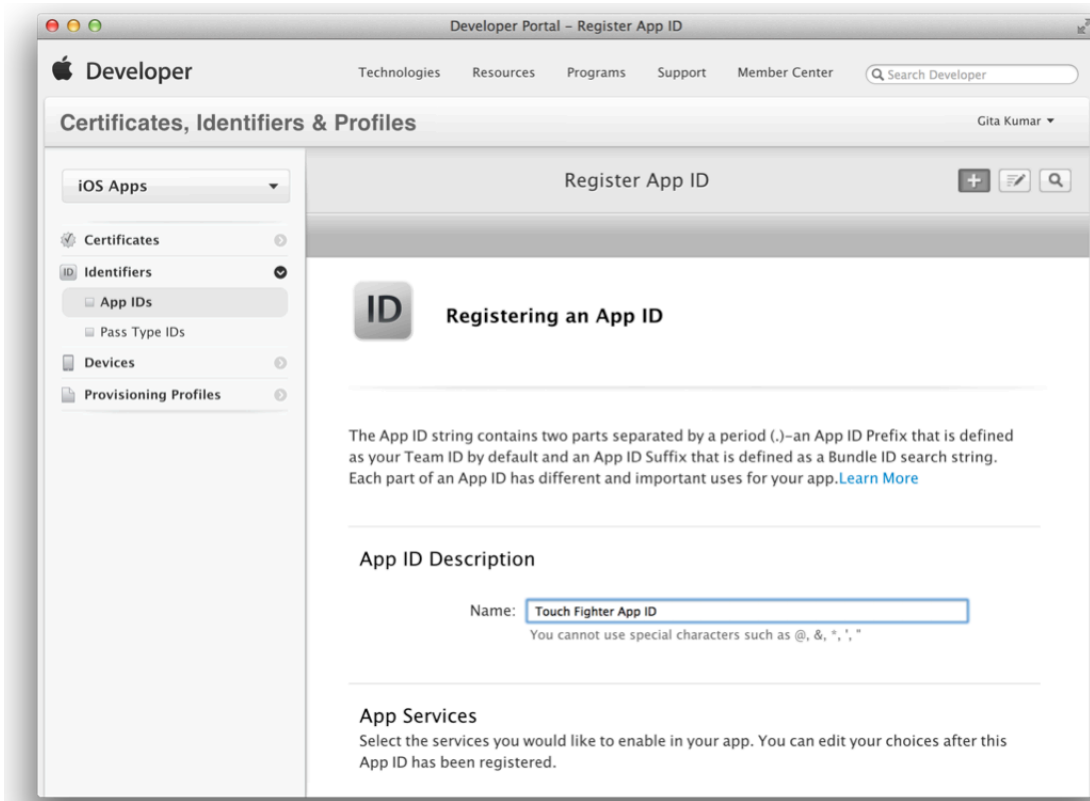
### To register an App ID

1. In [Certificates, Identifiers & Profiles](#), select Identifiers.
2. Under Identifiers, select App IDs.

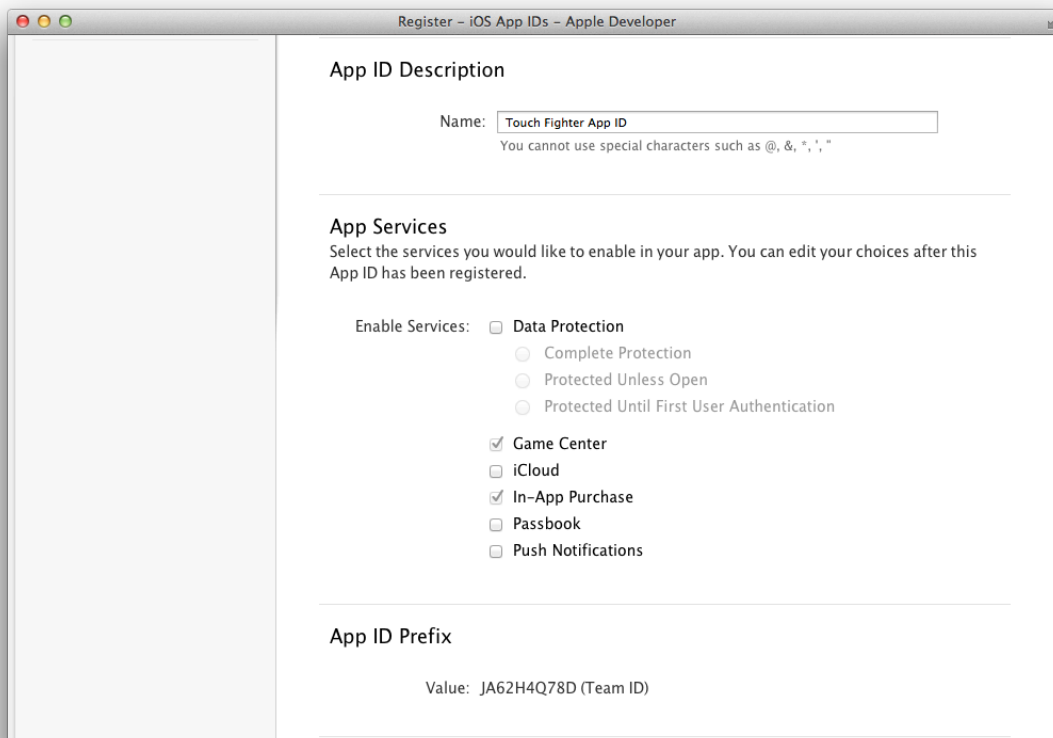
3. Click the plus button (+) in the upper-right corner.



4. Enter a name or description.



5. Click the corresponding checkboxes to enable the technologies you want to use.



**Tip:** If a technology is enabled and the checkbox is disabled, then that technology is automatically enabled for the type of App ID you are creating. If a technology is disabled and the checkbox is disabled, then that technology is not available for the type of App ID you are creating.

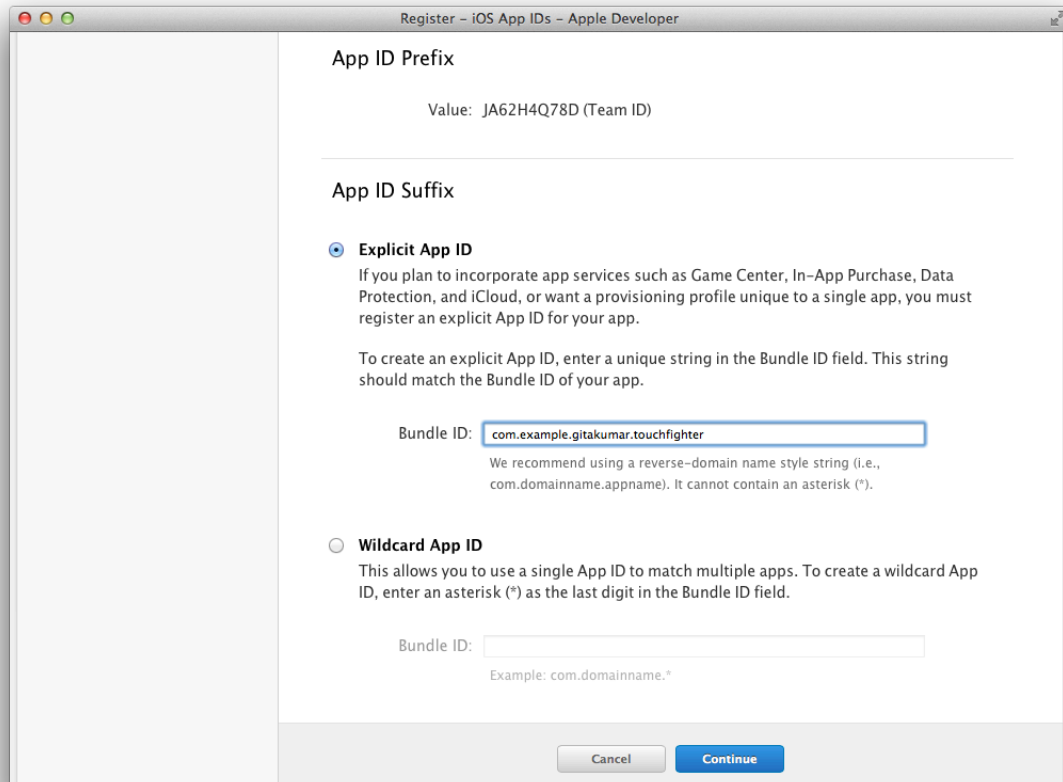
6. For iOS apps, if you select Data Protection, select the default level of protection.

You select one of the following data protection levels for your app:

- **Complete Protection.** The file is encrypted and inaccessible while the device is locked.
- **Protected Unless Open.** The file is encrypted. A closed file is inaccessible while the device is locked. After the user unlocks the device, your app can open the file and use it. If the user locks the device while the file is open, though, your app can continue to access it.
- **Protected Until First User Authentication.** The file is encrypted and inaccessible until after the device has booted and the user has unlocked it once.

7. To create an explicit App ID, select Explicit App ID under App ID Suffix and enter the bundle ID in the Bundle ID text field.

An explicit App ID exactly matches the bundle ID of an app you are building—for example, `com.example.gitakumar.touchfighter`. An explicit App ID cannot contain an asterisk (\*).



The App ID you enter should match your bundle ID that appears in the target's Summary pane in Xcode.

8. To create a wildcard App ID, select Wildcard App ID under App ID Suffix and enter a bundle ID search string in the Bundle ID text field.

A wildcard App ID ends with an asterisk (\*). For example, the bundle ID search string `com.example.*` matches all apps whose bundle IDs start with `com.example`.

9. Click Continue.
10. Review the registration information, then click Submit.
11. Click Done.

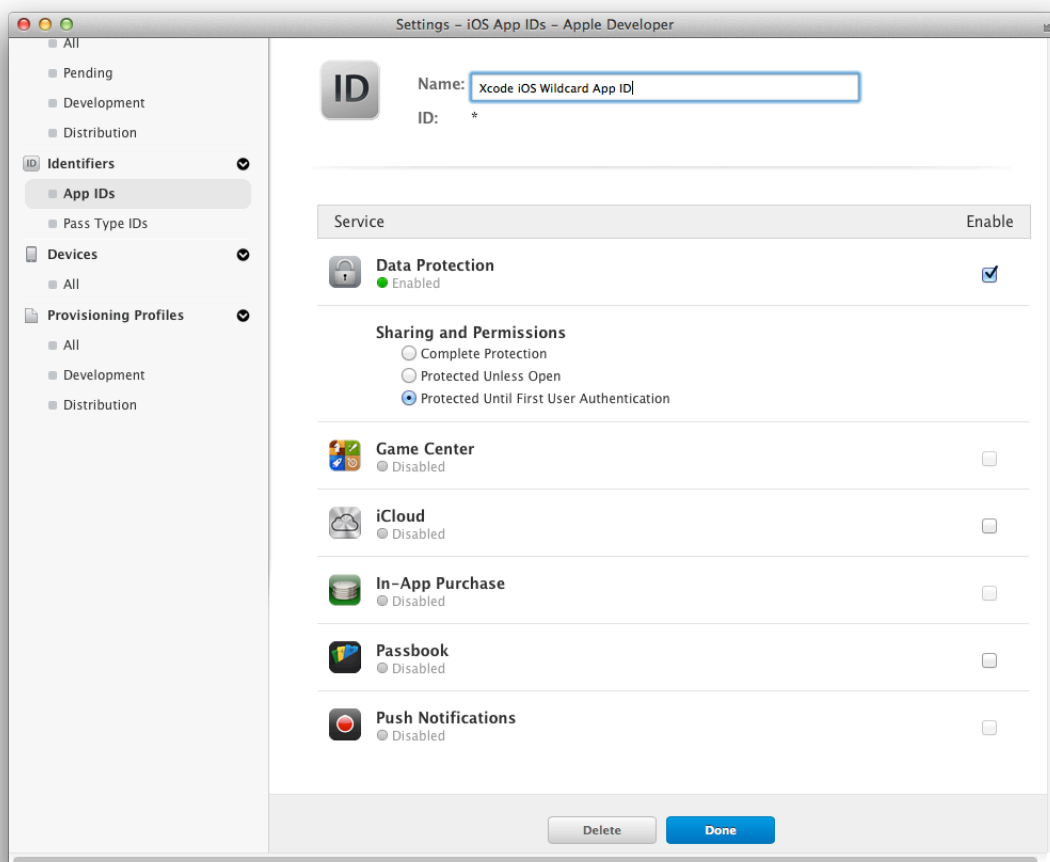
## Enabling Store Technologies

If you have an existing App ID, including the Xcode wildcard App ID, you can modify the settings to add other capabilities to your app. For iOS apps, Game Center and In-App Purchase are enabled by default for an explicit App ID. For Mac apps, In-App Purchase is enabled by default for an explicit App ID.

If you use the team provisioning profile, edit the settings of the wildcard App ID Xcode creates for you. For iOS apps, this App ID is called *Xcode iOS Wildcard App ID*. For Mac apps, this App ID is called *Xcode Mac Wildcard App ID*.

### To enable technologies for an existing App ID

1. In [Certificates, Identifiers & Profiles](#), select Identifiers.
2. Under Identifiers, select App IDs.
3. Select the App ID you want to change, and click Settings.
4. Click the corresponding checkboxes to enable the technologies you want to use.





**Tip:** If a technology is enabled and its checkbox is disabled, that technology is automatically enabled for the type of App ID. If a technology is disabled and its checkbox is disabled, that technology is not available for the type of App ID.

5. For iOS apps, if you select Data Protection, select the default level of protection under Sharing and Permissions.

You select one of the following data protection levels for your app:

- **Complete Protection.** Files are encrypted and inaccessible when the device is locked.
- **Protected Unless Open.** Files are encrypted. A closed file is inaccessible when the device is locked. After the device is unlocked, your app can open and use the file. If the user has a file open and locks the device (for example, by pressing the sleep button), your app can continue to access the file.
- **Protected Until First User Authentication.** Files are encrypted and inaccessible until after the device has booted and has been unlocked once.

6. If a warning dialog appears, click OK.

Later, you'll regenerate the provisioning profiles that use the App ID (including the team provisioning profile if you use it).

7. Click Done.

For iOS apps, you can also set the level of protection programmatically for files created by your app, as described in "Protecting Data Using On-Disk Encryption" in *iOS App Programming Guide*.

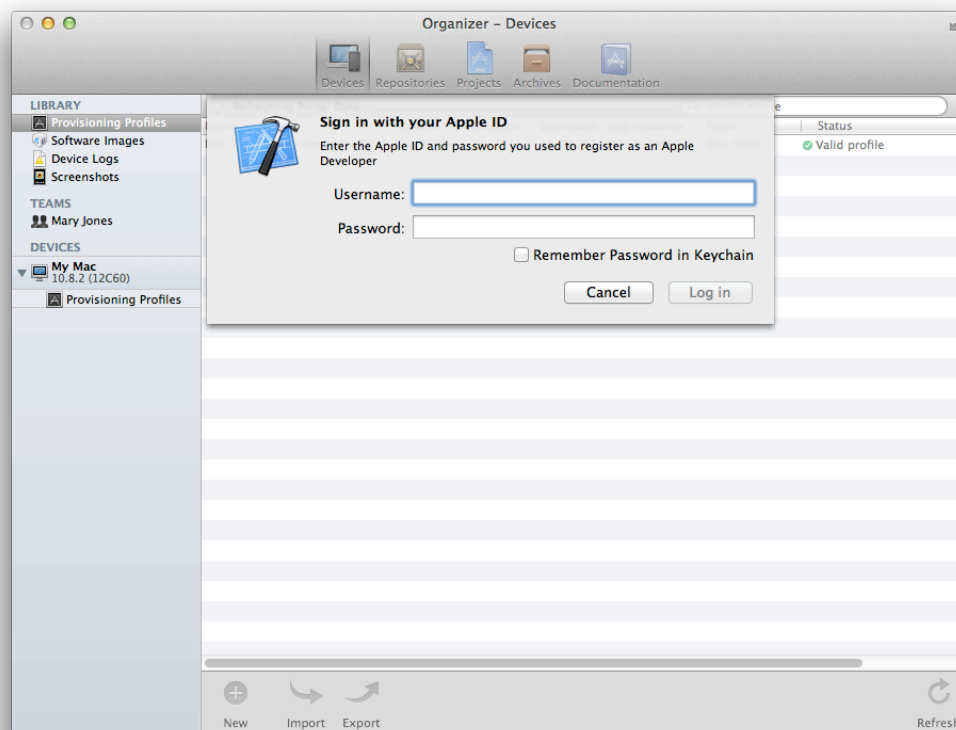
## Creating Development Provisioning Profiles

If you created an App ID, use Xcode to create a development provisioning profile containing your new App ID. You can also use the Xcode wildcard App ID to create a variation of the team provisioning profile (containing a subset of development certificates and devices).

To create a development provisioning profile, you select an App ID, one or more development certificates, and one or more devices. If you need to create your development certificates, read "[Creating Your Signing Certificates](#)" (page 24). If you need to register devices, read "[Developing Apps Using the Team Provisioning Profile](#)" (page 38).

## To create a development provisioning profile

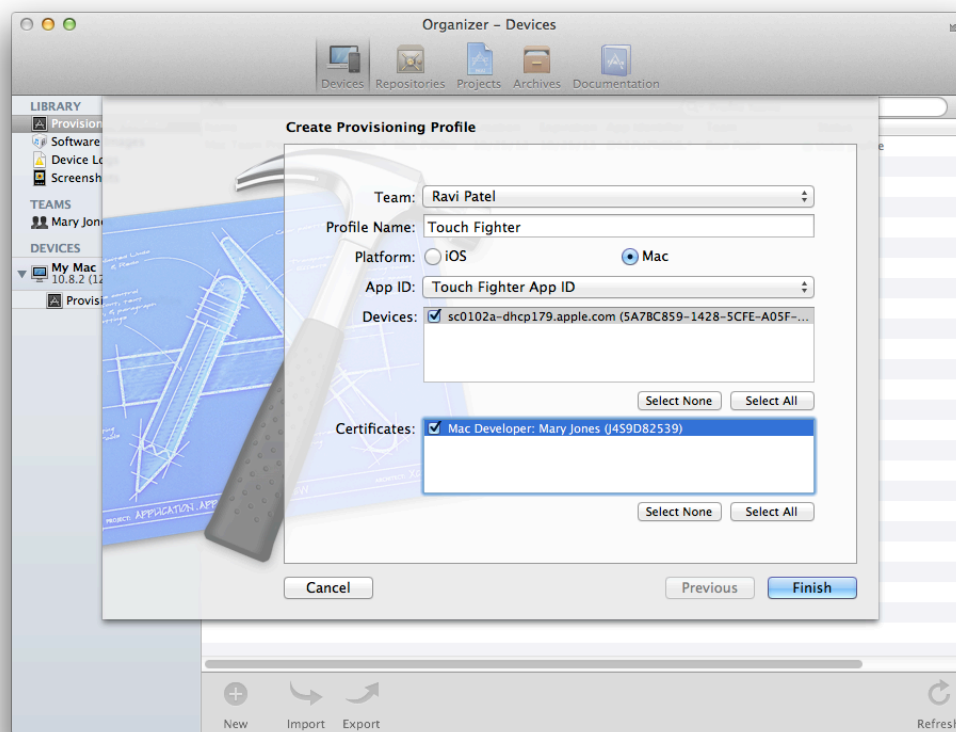
1. In Xcode, choose Window > Organizer to open the Organizer window.
2. Click Devices to display the Devices organizer.
3. Select Provisioning Profiles under Library.
4. Click New (+) at the bottom of the window.
5. Enter your user name and password, and click "Log in."



6. If you belong to multiple teams, select your team from the Team pop-up menu.



7. Enter a provisioning profile name.



8. Select iOS to create an iOS provisioning profile, or select Mac to create a Mac provisioning profile.
9. Select your App ID from the App ID menu.
10. Select your devices from the Devices list.
11. Select your development certificates from the Certificates list.
12. Click Finish.

The provisioning profile should appear in the Devices organizer as valid.

## Regenerating the Provisioning Profile

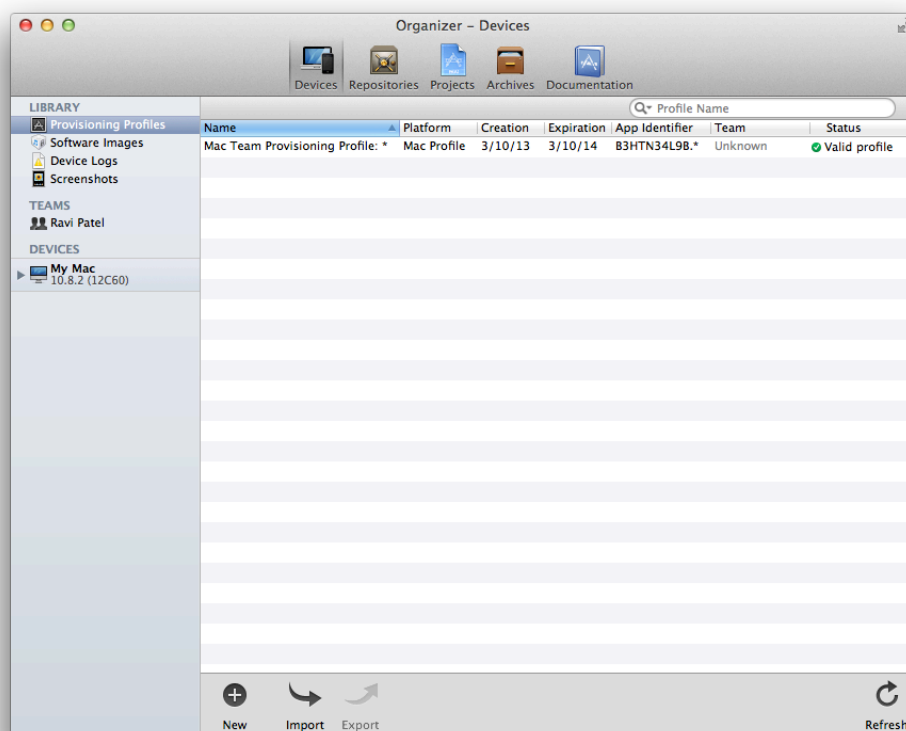
If you edit an App ID, provisioning profiles that contain that App ID become invalid until you regenerate them. For example, if you edit the Xcode wildcard App ID, any provisioning profile containing it—including the team provisioning profile—needs to be regenerated. After you regenerate a provisioning profile, update it on all devices you previously installed it on. The steps to regenerate the team provisioning profile are different from the steps to regenerate a provisioning profile you created.

## Regenerating the Team Provisioning Profile

The team provisioning profile, which is managed by Xcode, is not regenerated immediately when you make changes in Member Center. This is true even when you register devices using Member Center, not Xcode. To regenerate the team provisioning profile, you refresh the provisioning profiles or make some other change to the team provisioning profile using Xcode. Xcode regenerates the team provisioning profile and, for iOS apps, automatically replaces it on any connected iOS devices. For Mac apps, you need to replace the team provisioning profile in System Preferences yourself.

### To regenerate the team provisioning profile managed by Xcode

1. In Xcode, open the Devices organizer.
2. Select Provisioning Profiles under Library.



3. Click Refresh at the bottom of the window.

The team provisioning profile containing your changes should now appear in the Devices organizer as valid. For iOS apps, Xcode replaces the team provisioning profile on any registered iOS devices.

For Mac apps, you need to replace the team provisioning profile on your Mac. (Xcode updates the team provisioning profile automatically only on an iOS device.)

### To replace the team provisioning profile on a Mac

1. In Xcode, open the Devices organizer.
2. Click the disclosure triangle next to your device under Devices.
3. Select Provisioning Profiles under your device.
4. Select the team provisioning profile listed in the detail area.
5. Press Delete (on the keyboard) and click the Delete button.
6. Select Provisioning Profiles under Library.
7. Drag the team provisioning profile to your device under Devices.

## Regenerating Provisioning Profiles Managed By You

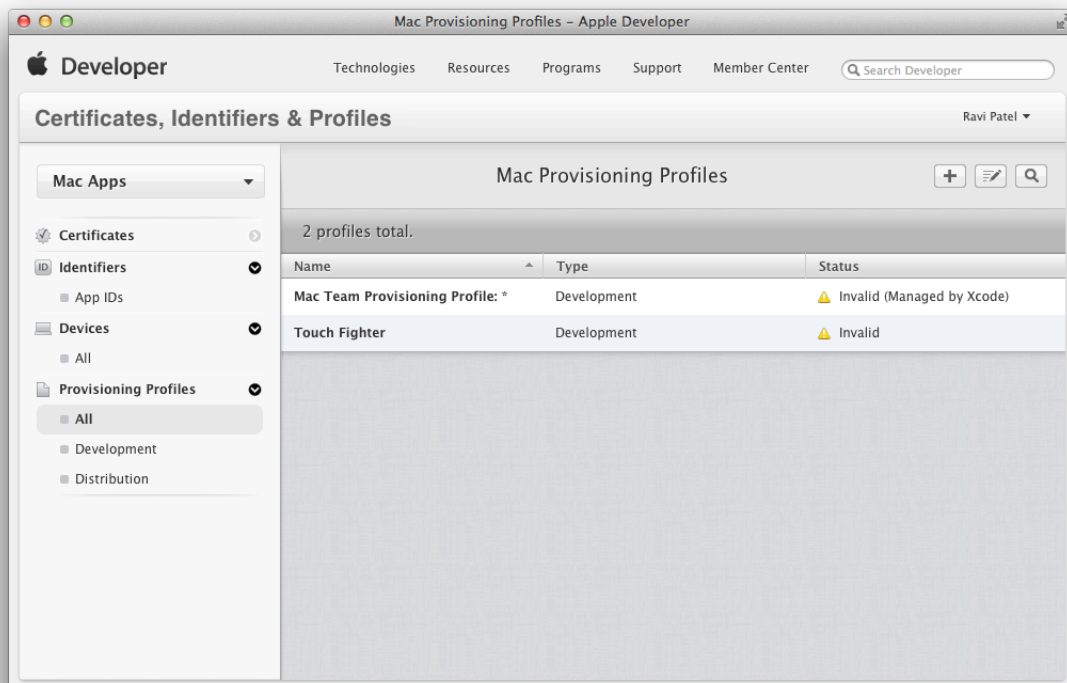
If a provisioning profile you created is invalid because you edited its App ID, regenerate the provisioning profile manually using Member Center.

### To regenerate a provisioning profile you created

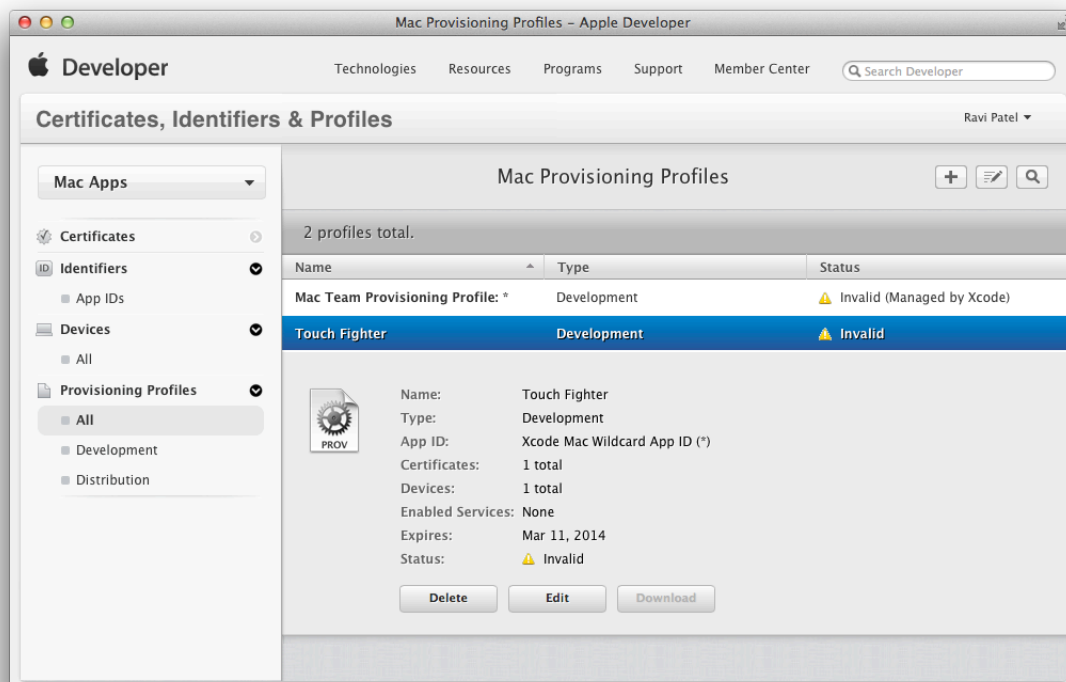
1. In [Certificates, Identifiers & Profiles](#), select Provisioning Profiles.
2. Under the Provisioning Profiles section, select All.

Provisioning profiles that have an Invalid status need to be regenerated.

3. Select the invalid provisioning profile.



4. Click Edit.



5. Scroll to the bottom of the page, and click Generate.
6. Click Done.

## Provisioning Your Development Devices

Except for the iOS team provisioning profile, after you create or regenerate a provisioning profile, you need to install it on your device. The provisioning profile you use to sign your app needs to be installed on the device before you can run your app on the device. If you previously installed a provisioning profile on a device, you need to replace it with the current version. First, you download the latest provisioning profiles from Member Center, and then you update the provisioning profiles on the device.

If you use the team provisioning profile, you completed this step when you regenerated it, as described in [“Regenerating the Team Provisioning Profile”](#) (page 66).

## Refreshing Your Provisioning Profiles Using Xcode

Changes you make in Member Center are not automatically downloaded until you refresh provisioning profiles in Xcode. To keep the Xcode copies up to date, always perform this step after you regenerate provisioning profiles in Member Center.

### To refresh provisioning profiles

1. In Xcode, open the Devices organizer.
2. Select Provisioning Profiles under Library.
3. Click Refresh at the bottom of the window.
4. Enter your user name and password, and click “Log in.”

The provisioning profile containing your App ID should now appear in the Devices organizer as valid.

## Updating Provisioning Profiles on Your Device

It’s your responsibility to maintain provisioning profiles that you manage on your devices. When provisioning profiles become invalid, you need to remove them from the device. A provisioning profile becomes invalid on a device if it doesn’t match the provisioning profile in Xcode. To install and remove provisioning profiles, read [“Installing and Removing Provisioning Profiles from Devices”](#) (page 177).

## Setting the Bundle ID to Match Your App ID

Next, set your app’s bundle ID in Xcode to match your App ID. If you use the team provisioning profile or the Xcode wildcard App ID in your provisioning profile, you can skip this step because the Xcode wildcard App ID matches all your apps.

### To set the bundle ID in your project

1. In Xcode, select the target in the project editor.
2. Select the Info tab.
3. Enter the bundle ID in the Value column of the “Bundle identifier” row.

The Xcode project template uses the Product Name build setting, which defaults to your app name, as the suffix for the default bundle ID. So if you use the app name in your explicit App ID, you can just replace the surrounding text to set the bundle ID. However, unlike domain names, App IDs and bundle IDs are case sensitive. If the App ID is lowercase and your app name is not, you need to replace the entire bundle ID text to match the App ID exactly.

## Signing Your App Using Your Development Provisioning Profile

Your last step is to set the Code Signing Identity build setting to your development certificate that resides in your development provisioning profile. By selecting the development certificate in the provisioning profile, you are instructing Xcode to install the provisioning profile in your app's bundle. The operating system uses the embedded provisioning profile to determine whether your app can launch on the device and use certain services. Consequently, your app won't be able to access store technologies if it's not configured correctly.

Before continuing, review the tasks you should have previously performed in this chapter:

	Task
<input checked="" type="checkbox"/>	Enable the App ID to use store technologies.
<input checked="" type="checkbox"/>	Regenerate the provisioning profile.
<input checked="" type="checkbox"/>	Install the provisioning profile on your device.

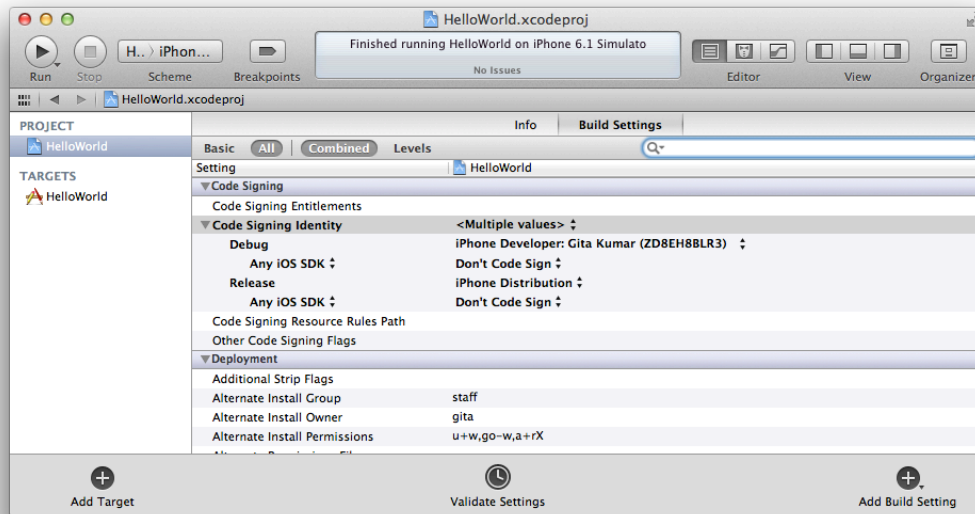
Then set the code signing identity and run your app.

### To set the code signing identity to your development certificate in a provisioning profile

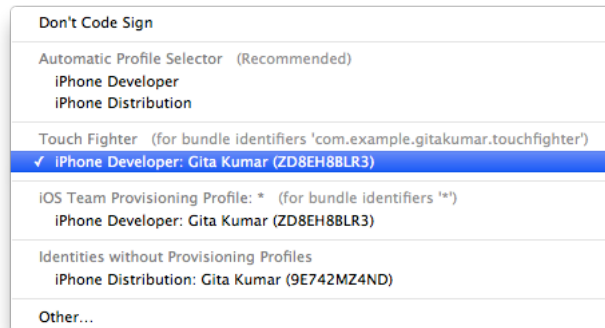
1. In the Xcode project editor, select the target.

**Important:** If you want to sign multiple targets with the same code signing identity, select the project, not a target.

2. Select the Build Settings tab.



3. Click All.
4. Type Code Signing in the search field in the Build Settings pane of the project editor.
5. From the Code Signing Identity pop-up menu, in the development provisioning profile section, choose your development certificate.



To run your iOS app on your device, connect your device to your Mac, select your device from the Scheme pop-up menu, and click Run. To run your Mac app, click Run.



## Verify Your Steps

You should verify that code signing works, that the App ID is configured correctly, and that the provisioning profile embedded in your app is correct. However, some technologies won't be fully configured until you follow the steps in [“Configuring Store Technologies in Xcode and iTunes Connect”](#) (page 79). Nevertheless, you should troubleshoot any code signing and provisioning problems early.

Later, you can verify the entitlements of the embedded provisioning profile in your app bundle, as described in [“Verify Your Steps”](#) (page 92).

## Verify Code Signing

To verify that there are no code signing errors, build and run your app. For iOS apps, connect your device to your Mac, select it from the Destination pop-up menu, and click the Run button to build, sign, and launch your app. (For more details on launching your iOS app, read [“Launching Your iOS App on the Device”](#) (page 52).) For Mac apps, click the Run button. If a code signing error occurs, read [“Troubleshooting Code Signing Errors”](#) (page 76).

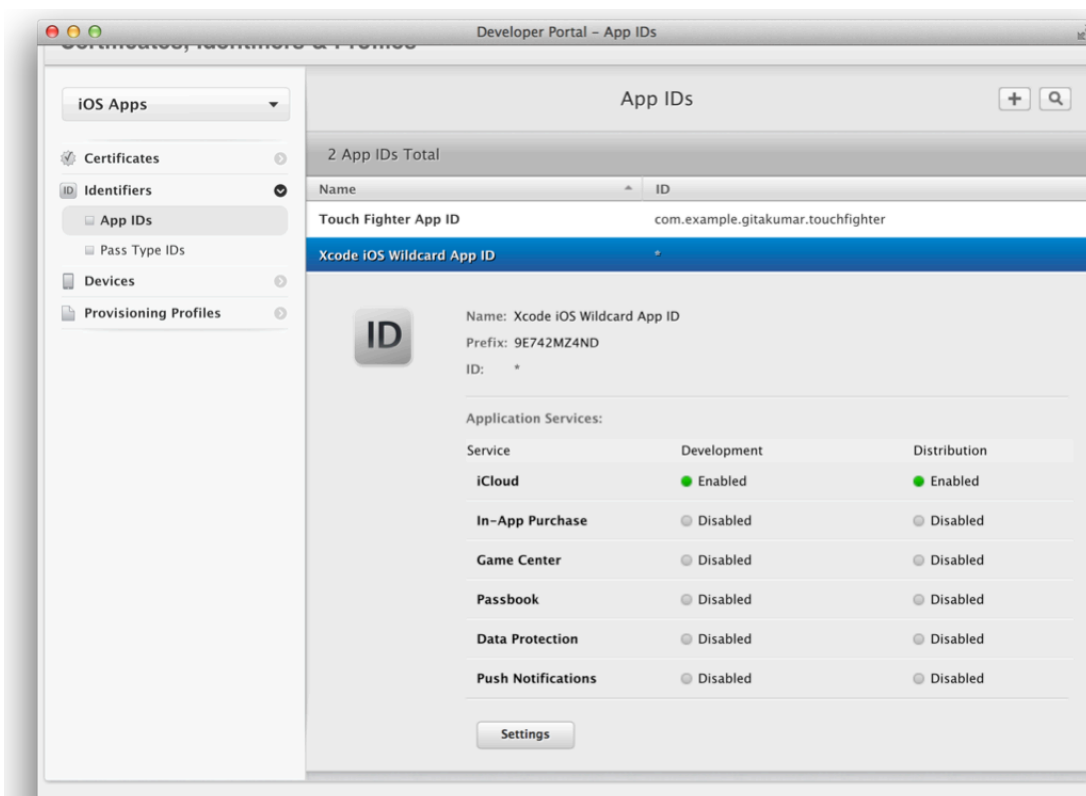
## Verify the App ID Settings in Member Center

The first time you enable store technologies, verify that the App ID and provisioning profile you are using are configured correctly.

### **To verify that an App ID is configured for store technologies**

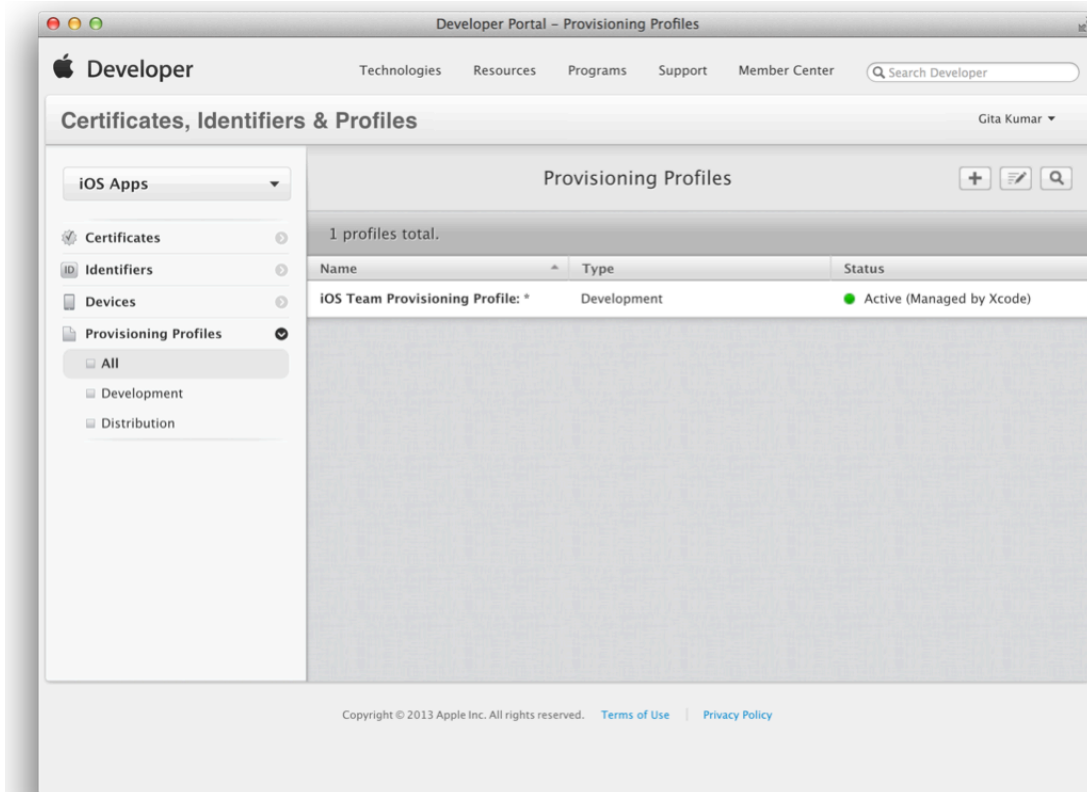
1. In [Certificates, Identifiers & Profiles](#), select Identifiers.
2. Under the Identifiers section, select App IDs.
3. Click the row for the App ID you want to verify.

Green circles followed by the word Enabled should appear in the Development and Distribution columns for each technology you want to use.



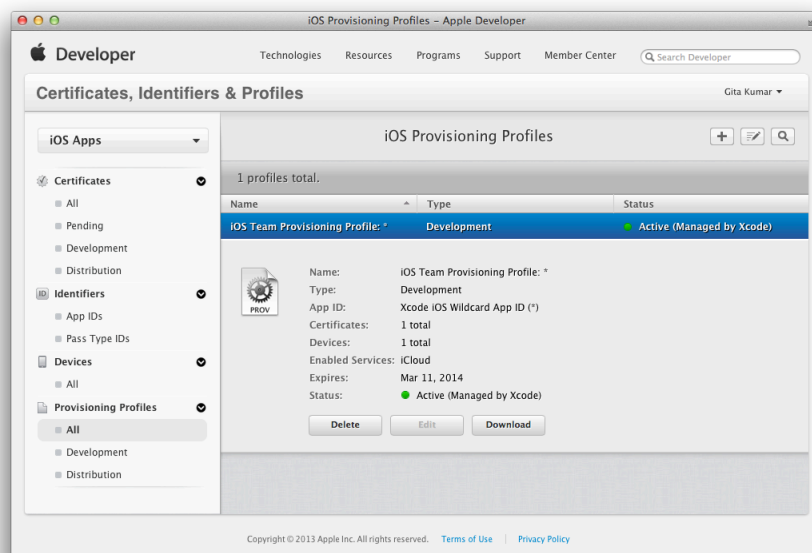
4. Under the Provisioning Profiles section, select All.

The status for the provisioning profile should be Active, as shown below.



5. Select the provisioning profile you are using.

Details about the provisioning profile are displayed. The technologies you want to use should be listed in the Enabled Services row.



## Troubleshooting

If you encounter problems code signing, building, and launching your app, follow the steps in this section.

### Troubleshooting Code Signing Errors

If your development certificate or team provisioning profile doesn't appear in the Code Signing Identity menu (in the Build Settings pane of the project editor), read ["Your Provisioning Profile Doesn't Appear in the Code Signing Identity Menu"](#) (page 208).

If a code signing error occurs when you build the app, verify that the Code Signing Identity build setting is correct. Also, check whether the Code Signing Identity build setting is set at the project or target level (target settings override project settings).

Read ["Certificate Issues"](#) (page 208) to troubleshoot other code signing problems.

### Troubleshooting Failure to Launch

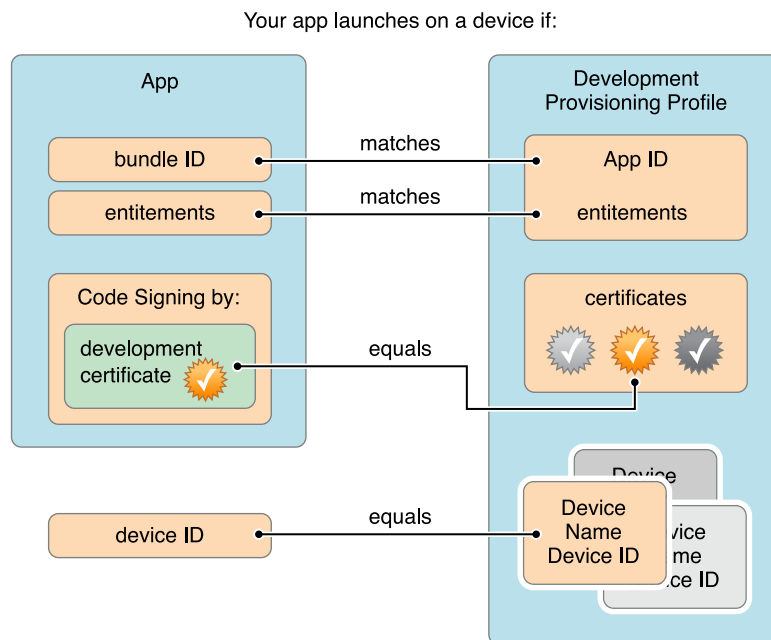
If you build the app but it fails to launch on the device, the provisioning profile installed on the device may be wrong. Replace the provisioning profile on the device with the new provisioning profile.

## To replace a provisioning profile on a device

1. In Xcode, open the Devices organizer.
2. Click the disclosure triangle next to your device under Devices.
3. Select Provisioning Profiles under the name of your device.
4. Select the old provisioning profile listed in the detail area.
5. Enter Delete, and click the Delete button.
6. Select Provisioning Profiles under Library.
7. Drag the new provisioning profile to your device under Devices.

## Development Provisioning Profiles in Depth

An app needs to match the installed development provisioning profile to launch. When you are ready to run your app, you sign it using your development certificate contained in the development provisioning profile. Then you install the provisioning profile and the app on the device. The app successfully launches if the app's bundle ID matches the App ID, the signature matches a development certificate, and the device is in the device list contained in the provisioning profile.



Therefore, a development provisioning profile allows a team member to build an app on his or her Mac and share it with other team members, who can then run that app on their devices. For small teams, you might have one development provisioning profile (for example, the team provisioning profile) containing all team member devices and certificates. For larger teams, you can create multiple development provisioning profiles for specific purposes and groups.

When you create a development provisioning profile, you can limit any of these three areas to provide additional security:

- With a wildcard App ID, you can limit the profile to a subset of your apps; with an explicit App ID, you can limit the profile to a single app.
- You can restrict the developers allowed to sign an app using the profile.
- You can restrict the devices that the app runs on.

## Recap

In this chapter you learned how to enable store technologies for an App ID and create or regenerate a development provisioning profile containing the App ID using Member Center. These same steps work for wildcard and explicit App IDs. The next chapter describes additional configuration of store technologies using Xcode and iTunes Connect.

# Configuring Store Technologies in Xcode and iTunes Connect

This chapter describes the additional setup required to use store technologies. For example, you need to enable iCloud entitlements in your Xcode project to use it. Newsstand and routing apps require some configuration in the information property list. You need to add frameworks to use some of the technologies. Finally, some technologies require additional setup in iTunes Connect. This chapter covers the setup for all store technologies, even those that don't require any provisioning.

[Table 4-1](#) (page 56) shows which assets need to be configured to use each store technology. For the steps to edit your App ID and provisioning profile in Member Center, read [“Provisioning Your App for Store Technologies”](#) (page 54).

## About Entitlements

An **entitlement** is a single right granted to a particular app, tool, or other executable that gives it additional permissions above and beyond what it would ordinarily have. The term *entitlement* is most commonly used in the context of a sandbox, and to a lesser degree for an App ID. Regardless of the location, an entitlement is a piece of configuration information included in your app's code signature—telling the system to allow your app to access certain resources or perform certain operations. In effect, an entitlement extends the sandbox and capabilities of your app to allow a particular operation to occur. You set some entitlements for an App ID in Member Center—for example, by enabling store technologies—and others in the Xcode project.

## Configuring iCloud

iCloud storage allows you to share the user's data among multiple instances of your app running on different iOS and Mac OS X devices. Your app needs to be provisioned to use iCloud, which includes setting entitlements in your Xcode project.

Before continuing, review the tasks that should be complete before you configure iCloud in Xcode:

	Task
<input checked="" type="checkbox"/>	Enable the App ID to use iCloud.
<input checked="" type="checkbox"/>	Create or regenerate a development provisioning profile containing the App ID.

	Task
<input checked="" type="checkbox"/>	Provision your development device.
<input checked="" type="checkbox"/>	Code sign your app using the development provisioning profile.
<input type="checkbox"/>	Set iCloud entitlements in your Xcode project.

You should have enabled your App ID to use this technology, as described in “[Provisioning Your App for Store Technologies](#)” (page 54). Because iCloud doesn’t require an explicit App ID, you can use the team provisioning profile during development. After you do this, launch Xcode to set the entitlements.

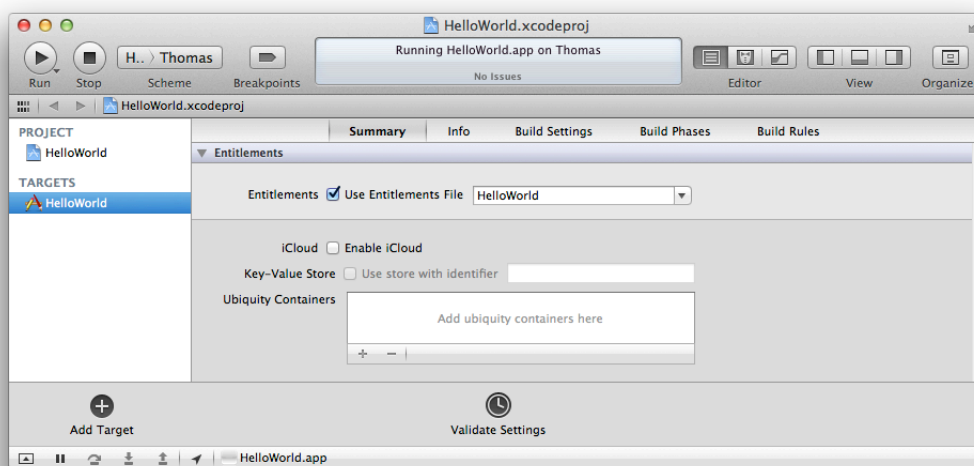
To learn more about using iCloud storage, read *Your Third iOS App: iCloud and iCloud Design Guide*.

## Enabling iCloud Entitlements

Before you can configure iCloud key-value storage or iCloud document storage, you need to enable iCloud entitlements in Xcode.

### To enable iCloud entitlements

1. In Xcode, select the target in the project editor.
2. Select the Summary tab and scroll down to the Entitlements section.



3. If entitlements are not enabled, select Use Entitlements File.
4. In the iCloud section, select Enable iCloud.



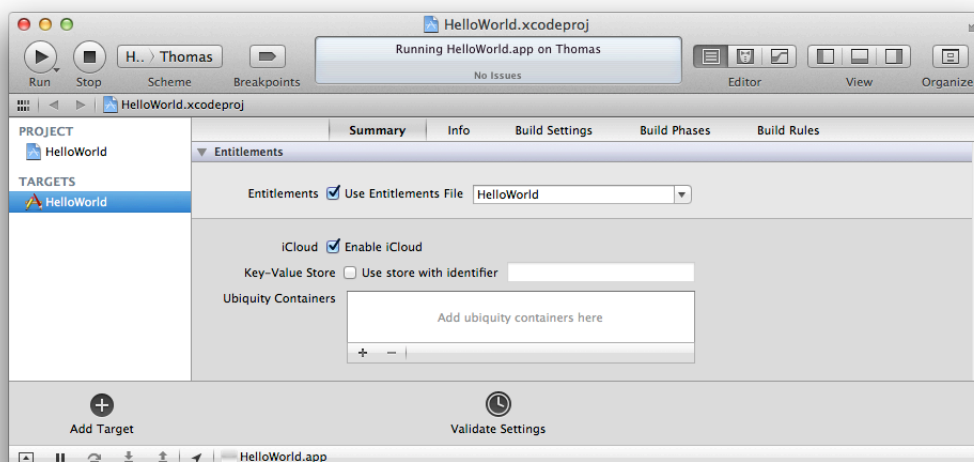
## Configuring iCloud Key-Value Storage

iCloud key-value storage allows an app to share small amounts of data with other instances of itself running on the user's other devices.

### To configure iCloud key-value storage

1. In the iCloud entitlements area on the Summary tab in the project editor, select “Use store with identifier.”

The identifier defaults to your bundle ID.



2. If you want to change the identifier, enter it in the text field adjacent to the “Use store with identifier” checkbox.

To learn how to use iCloud key-value storage for preferences, read *iCloud Design Guide*.

## Configuring iCloud Document Storage

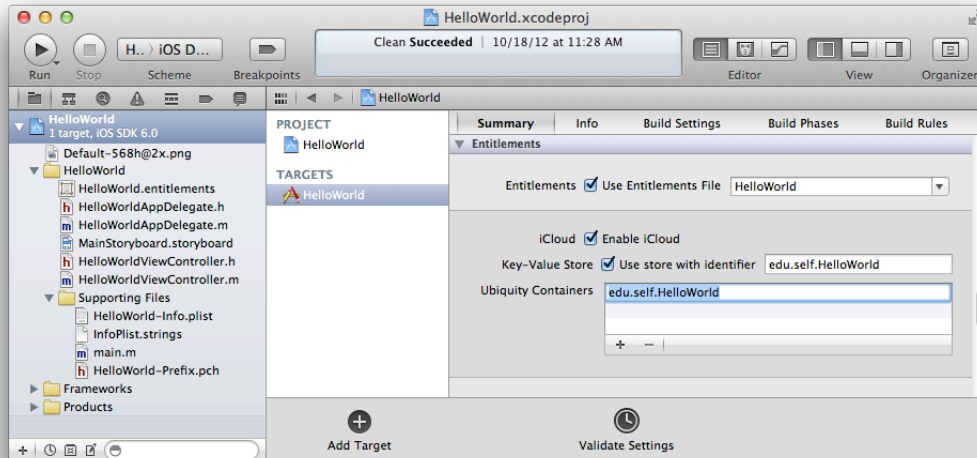
iCloud document storage is used to store user documents and app data in the user's iCloud account. Each app has a container in the user's iCloud account identified by its App ID. An app can access containers belonging to other apps created by your team as well.

To configure iCloud document storage, add one or more iCloud containers. Add your bundle identifier to the container list or add a wildcard App ID to match a set of App IDs. The first container identifier cannot be a wildcard App ID. For guidance on selecting iCloud containers, read *iCloud Design Guide*.

## To add an iCloud container

1. In the iCloud entitlements area on the Summary tab in the project editor, click the Add button (+) at the bottom of the Ubiquity Containers field.

The first time you do this, the bundle ID is added to the list and appears highlighted.



2. Enter the App ID for the container you want to add.

To delete a container, select it from under the Ubiquity Containers field, and click the Delete button (-).

## Configuring Push Notifications

Apple Push Notification service (APNs) allows an app that is not running in the foreground to notify the user that it has information for the user. Your app needs to be provisioned to use push notifications and you can only download client SSL certificates after you register your App ID.

Before continuing, review the tasks that should be complete before you configure push notifications in Xcode:

	Task
<input checked="" type="checkbox"/>	Create or edit the settings of an explicit App ID.
<input checked="" type="checkbox"/>	Create or regenerate a development provisioning profile containing the explicit App ID.
<input checked="" type="checkbox"/>	Provision your development device.

	Task
<input checked="" type="checkbox"/>	Code sign your app using the development provisioning profile.
<input type="checkbox"/>	Generate a client SSL certificate for development.
<input type="checkbox"/>	Install the client SSL certificate and key on your server.

First, you need to create an explicit App ID and provisioning profile, as described in “[Provisioning Your App for Store Technologies](#)” (page 54). After you register your explicit App ID, you can generate client SSL certificates and install them on your server.

To learn more about push notifications, read *Local and Push Notification Programming Guide*.

## Creating Push Notification Client SSL Certificates

You use Member Center to generate your push notification client SSL certificates. A **client SSL certificate** allows your notification server to connect to the APNs. Each App ID is required to have its own client SSL certificate. Similar to signing certificates, you use separate client SSL certificates for development and distribution.

---

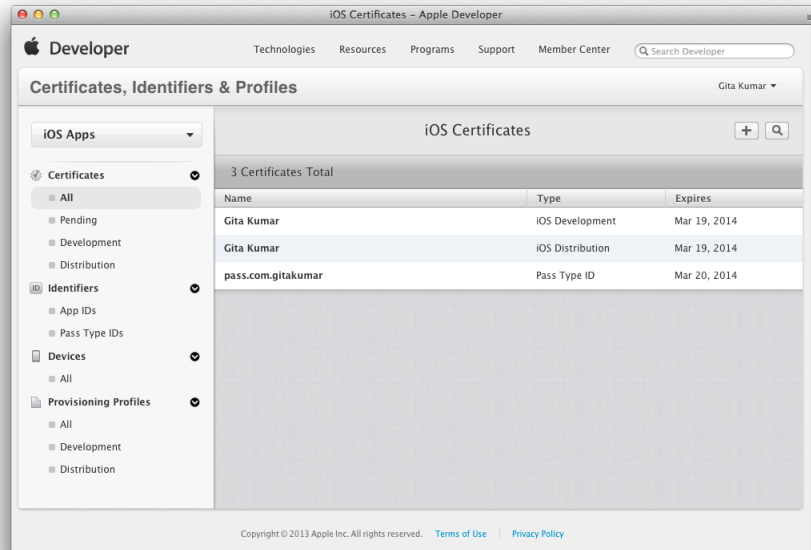
**Note:** The option to create an APNs client SSL certificate is not available if you don't have an App ID that enables APNs.

---

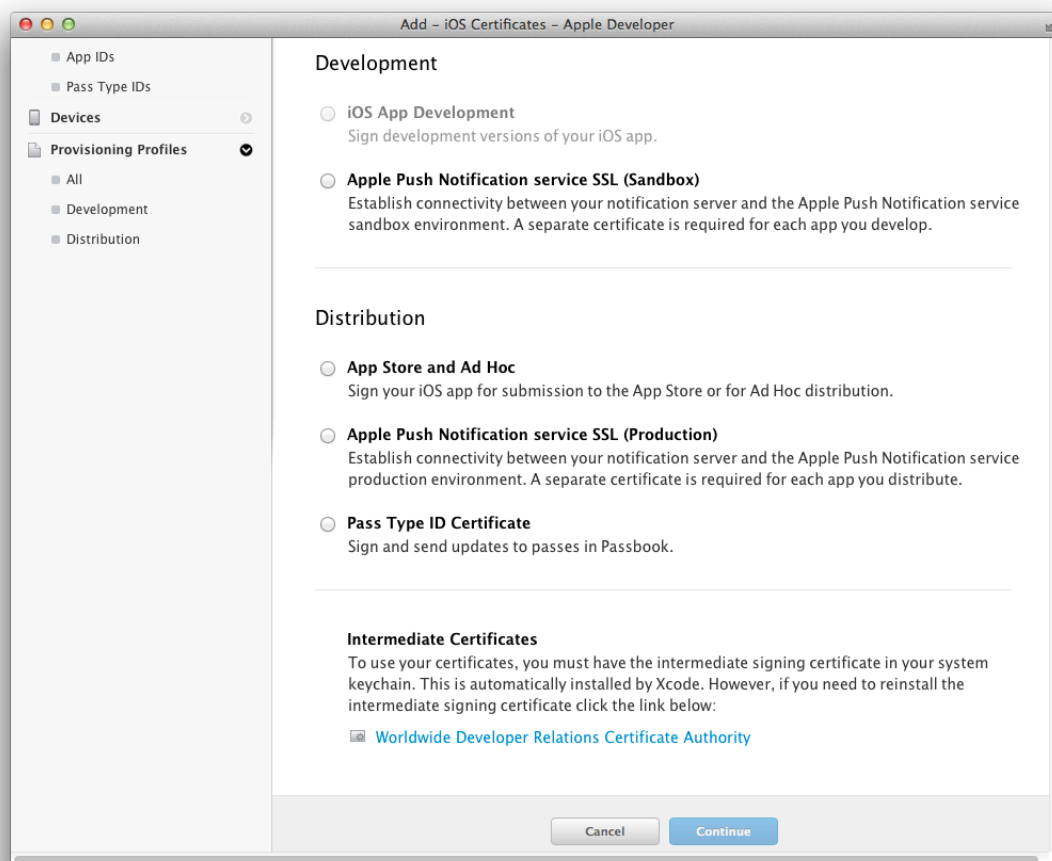
### To generate client SSL certificates

1. In [Certificates, Identifiers & Profiles](#), select Certificates.

2. Click the plus button (+) in the upper-right corner.

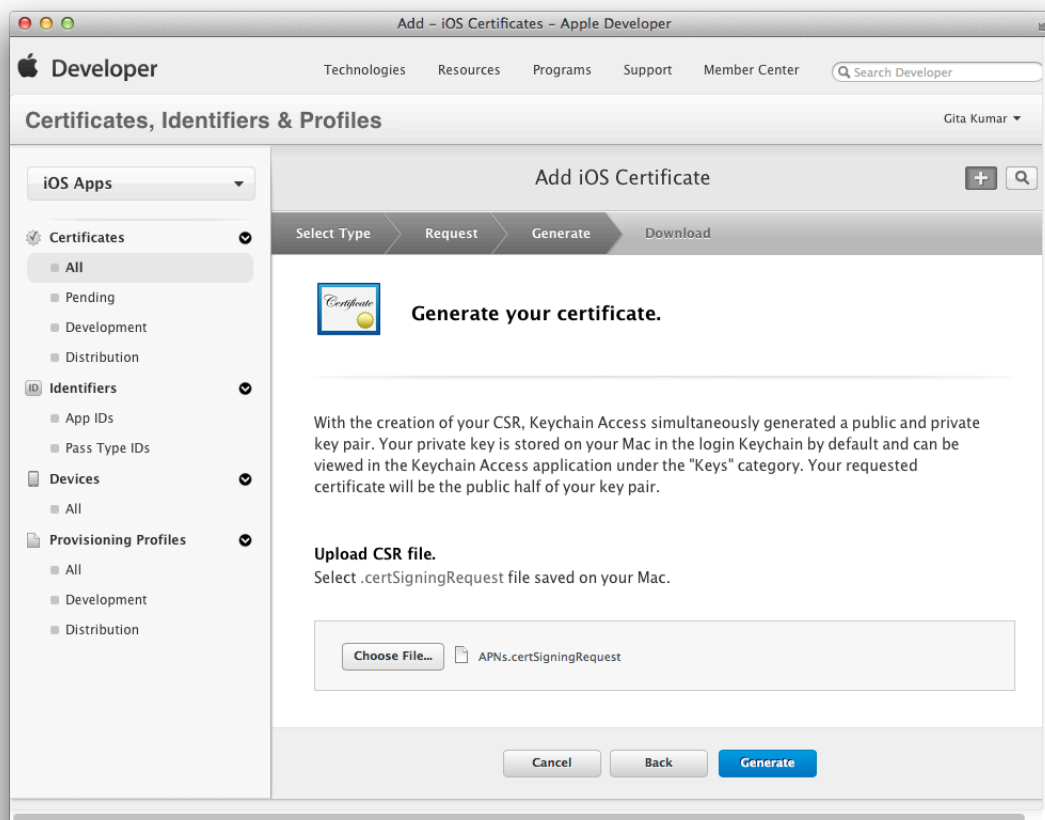


3. Select the checkbox next to “Apple Push Notification service SSL” either under Development or Distribution, and click Continue.



4. Select an App ID from the App ID pop-up menu and click Continue.
5. To create a certificate request, follow the instructions on the page that appears, and click Continue.
6. Click Choose File.
7. In the dialog that appears, select the certificate request file (with a `.certSigningRequest` extension) and click Choose.

8. Click Generate.



## Installing Client SSL Certificates

For how to install the client SSL certificate and key on a server, read “Provisioning Procedures” in *Local and Push Notification Programming Guide*.

## Configuring Game Center

Game Center is Apple’s social gaming network. It allows players to connect their devices to the Game Center service and exchange information. Before your app can use Game Center, it must first be provisioned to use Game Center and configured in iTunes Connect. You also add the Game Kit framework to your Xcode project and for Mac apps, set some network entitlements.

Before continuing, review the tasks that should be complete before you configure Game Center in Xcode:

	Task
<input checked="" type="checkbox"/>	Create or edit the settings of an explicit App ID.
<input checked="" type="checkbox"/>	Create or regenerate a development provisioning profile containing the explicit App ID.
<input checked="" type="checkbox"/>	Provision your development device.
<input checked="" type="checkbox"/>	Code sign your app using the development provisioning profile.
<input type="checkbox"/>	For Mac apps, set the network entitlements in your Xcode project.
<input type="checkbox"/>	Link to the Game Kit framework.
<input type="checkbox"/>	Configure your game app in iTunes Connect.

You need to create an explicit App ID and provisioning profile, as described in [“Provisioning Your App for Store Technologies”](#) (page 54). For iOS apps, Game Center is automatically enabled for all explicit App IDs. For Mac apps, you need to specifically enable Game Center when you create the App ID.

Next, you configure Game Center in your Xcode project. For Mac apps, you may need to set the network entitlements in the App Sandbox section, located in the Summary pane in Xcode. If your app opens network connections, it needs to allow outbound network connections; for this, set the `com.apple.security.network.client` entitlement. If your app listens for network connections, it needs to allow incoming network connections; for this, set the `com.apple.security.network.server` entitlement.

Then link to the Game Kit framework and begin writing code that uses Game Kit APIs. To learn more about using Game Center, read *Game Center Programming Guide*.

To configure your app in iTunes Connect, read [“Creating an App Record”](#) (page 153) to create the app record (enter your explicit App ID), and read [“Game Center”](#) in *iTunes Connect Developer Guide* to configure it.

## Configuring In-App Purchase

In-App Purchase embeds a store directly into your app by allowing you to connect to the store and securely process payments from the user. You can use In-App Purchase to collect payment for enhanced functionality or additional content usable by your app. You need to provision your app to use In-App Purchase, add the Store Kit framework to your Xcode project, and configure it in iTunes Connect. You also use iTunes Connect to create your in-app purchases.

Before continuing, review the tasks that should be complete before you use In-App Purchase:

	Task
<input checked="" type="checkbox"/>	Create or edit the settings of an explicit App ID.
<input checked="" type="checkbox"/>	Create or regenerate a development provisioning profile containing the explicit App ID.
<input checked="" type="checkbox"/>	Provision your development device.
<input checked="" type="checkbox"/>	Code sign your app using the development provisioning profile.
<input type="checkbox"/>	Link to the Store Kit framework.
<input type="checkbox"/>	Enter the App ID in iTunes Connect and create in-app purchases for testing.

If you have not already done so, create an explicit App ID, as described in [“Provisioning Your App for Store Technologies”](#) (page 54). In-App Purchase is automatically enabled for all explicit App IDs.

Then link to the Store Kit framework and refer to *In-App Purchase Programming Guide* for how to write your code.

To create an app record and enter the explicit App ID in iTunes Connect, read [“Creating an App Record”](#) (page 153). To create in-app purchases, read “In-App Purchase” in *iTunes Connect Developer Guide*.

## Configuring Passbook for iOS Apps

Presents digital representations of information—such as a coupon, ticket for a show, or boarding pass—that allow users to redeem a real-world product or service. You can use Passbook in several ways:

- To create, distribute, and update **passes**, register a pass type identifier and request a pass-signing certificate. You don't need an app or an entitlement to do this. For details, read *Passbook Programming Guide*.
- To let users add passes to the Passbook from your app, use PassKit framework. You don't need to set Passbook entitlements to do this.
- To access the user's passes in your app, follow the steps below.

Before continuing, review the tasks that should be complete before you configure Passbook:

	Task
<input checked="" type="checkbox"/>	Enable the App ID to use Passbook.



	Task
<input checked="" type="checkbox"/>	Create or regenerate a development provisioning profile containing the App ID.
<input checked="" type="checkbox"/>	Provision your development device.
<input checked="" type="checkbox"/>	Code sign your app using the development provisioning profile.
<input type="checkbox"/>	Link to the PassKit framework.
<input type="checkbox"/>	If needed, register a pass type identifier.
<input type="checkbox"/>	Set Passbook entitlements.

You should have enabled your App ID to use this technology, as described in [“Provisioning Your App for Store Technologies”](#) (page 54). Because Passbook doesn’t require an explicit App ID, you can use the team provisioning profile during development.

After performing these steps, link to the Pass Kit framework and read *Passbook Programming Guide* for how to set Passbook entitlements. To create a pass type identifier, read “Requesting a Pass Type Identifier” in *Passbook Programming Guide*.

## Configuring Data Protection for iOS Apps

Data protection adds a level of security to files stored on-disk by your app. Data protection uses the built-in encryption hardware present on specific devices to store files in an encrypted format on disk. Your app needs to be provisioned to use data protection.

Before continuing, review the tasks that should be complete before you use data protection:

	Task
<input checked="" type="checkbox"/>	Enable the App ID to use data protection.
<input checked="" type="checkbox"/>	Create or regenerate a development provisioning profile containing the App ID.
<input checked="" type="checkbox"/>	Set the default level of data protection.
<input checked="" type="checkbox"/>	Provision your development device.

	Task
<input checked="" type="checkbox"/>	Code sign your app using the development provisioning profile.
<input type="checkbox"/>	Set the level of protection programmatically for files created by your app.

You should have enabled your App ID to use data protection, as described in “[Provisioning Your App for Store Technologies](#)” (page 54), and selected a default level of protection. Data protection doesn’t require an explicit App ID so you can use the team provisioning profile during development. No other configuration is required in your Xcode project to use data protection but you can in addition change this setting in your code.

You can programmatically set the level of protection for files created by your app, as described in “Protecting Data Using On-Disk Encryption” in *iOS App Programming Guide*. Your app may override these App ID settings:

- **Complete Protection.** Files are encrypted and inaccessible when the device is locked.
- **Protected Unless Open.** Files are encrypted. A closed file is inaccessible when the device is locked. After the device is unlocked, your app can open and use the file. If the user has a file open and locks the device (for example, by pressing the sleep button), your app can continue to access the file.
- **Protected Until First User Authentication.** Files are encrypted and inaccessible until after the device has booted and has been unlocked once.

## Configuring Routing Apps for iOS Apps

Apps that are able to display point-to-point directions can register as routing apps and make those directions available to Maps and other apps. Use Xcode and iTunes Connect to configure your routing app.

Before continuing, review the tasks that you need to perform to configure a routing app:

	Task
<input type="checkbox"/>	Write the code to provide routing directions.
<input type="checkbox"/>	Enable this feature in Xcode.
<input type="checkbox"/>	Create an app record in iTunes Connect.
<input type="checkbox"/>	Upload a binary of your app to the store.
<input type="checkbox"/>	Upload your app’s geographic coverage file in iTunes Connect.

The geographic coverage file defines the regions that your app supports. You can upload the geographic coverage file when you first create the iTunes Connect app record or later after you submit a binary.

## Providing Routing Directions

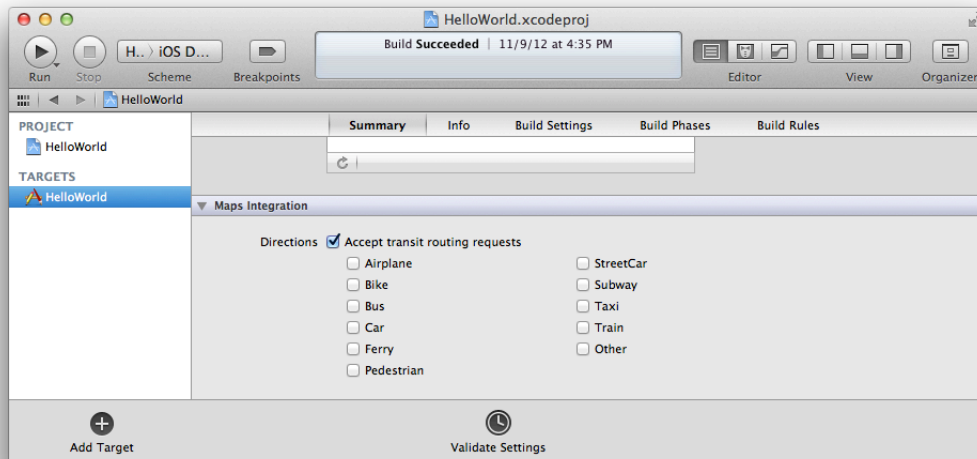
Read “Providing Directions” in *Location Awareness Programming Guide* to learn how to create a routing app.

## Enabling Routing Apps in Xcode

Enable routing apps in your Xcode project and select one or more supported modes.

### To configure Map Integration settings in the Xcode project

1. In Xcode, select the project and your target to display the project editor.
2. Click the Summary tab.
3. Select “Accept transit routing requests” to enable the routing app feature.



4. Select the supported modes from the checkboxes below.  
You are required to select one or more supported modes.

## Creating an App Record in iTunes Connect

Follow the steps in “Creating an App Record” (page 153). Optionally, you can upload the geographic coverage file when you create the app record, or later when you upload a binary.

## Submitting a Binary to the Store

Follow the steps in [“Submitting Your App”](#) (page 133) to upload a binary to iTunes Connect.

## Uploading the Geographic Coverage File to iTunes Connect

If you enable routing apps and submit a binary, Apple will not start the approval process until you upload the geographic coverage file.

### To upload the geographic coverage file after you submit your binary

1. Log in to [iTunes Connect](#).
2. On the iTunes Connect homepage, click Manage Your Applications.
3. Locate the app you want to edit, and click the large icon or app name.
4. Click View Details for the version of your app that you want to edit.
5. Click the Edit button that appears next to the Metadata and Uploads section.
6. Click the Choose File button under Routing App Coverage File.
7. Locate the file and click Choose.
8. Click Upload File.

If the file is not formatted correctly, a message appears at the top of the page.

## Configuring Newsstand for iOS Apps

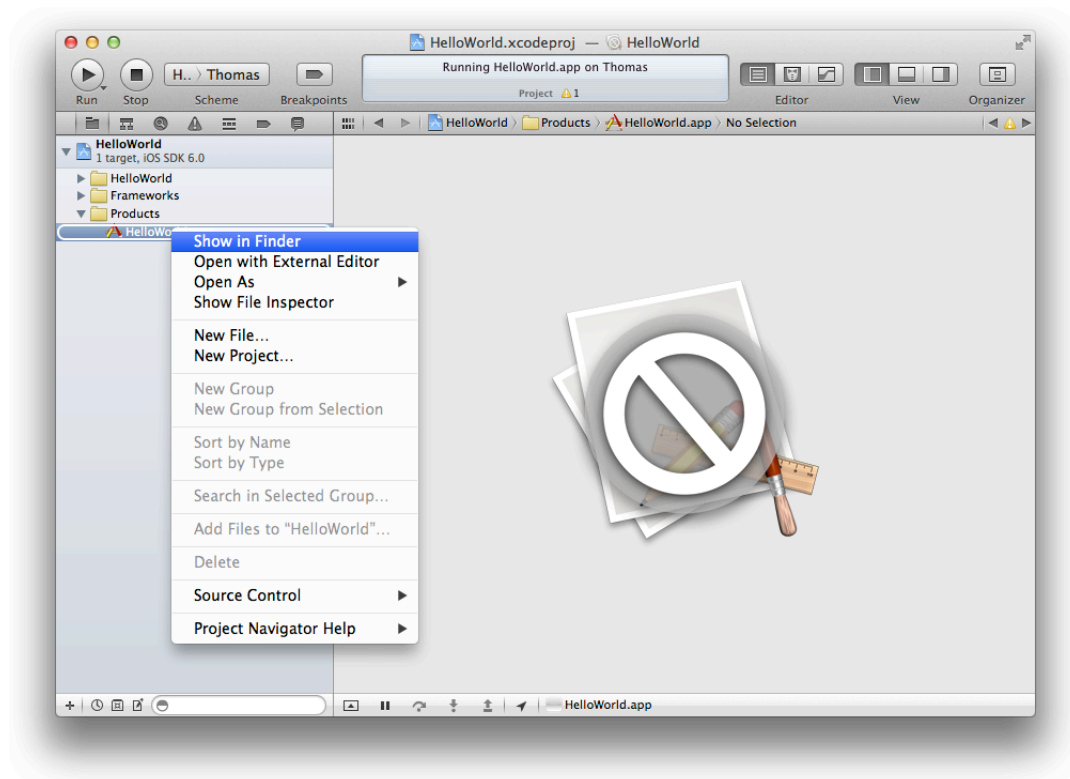
Newsstand enables an app to organize a user’s magazine and newspaper app subscriptions into a folder. To use Newsstand, just add some keys to the information property list and add artwork to your Xcode project. Refer to [Newsstand for Developers](#) for more information on creating a Newsstand app. Read “Newsstand Icons” in *iOS Human Interface Guidelines* for how to add Newsstand cover icons to your Xcode project.

## Verify Your Steps

Some entitlements (for example, App Sandbox entitlements) are set in your Xcode project and others are set in the provisioning profile. You can check if the signed app has the correct entitlements by examining the app’s signature. If there are any discrepancies, you can examine the embedded provisioning profile located in the app binary.

## To verify the entitlements of a signed app

1. In Xcode, select your project in the project navigator.
2. Click the disclosure triangle next to the project to reveal its contents.
3. Click the disclosure triangle next to Products to reveal the binary.
4. Control-click the binary and select “Show in Finder” to go to the Xcode build location in the Finder.



5. Launch Terminal (located in ~/Applications/Utilities) and enter this text followed by a space character (do not press Return):

```
codesign -d --entitlements -
```

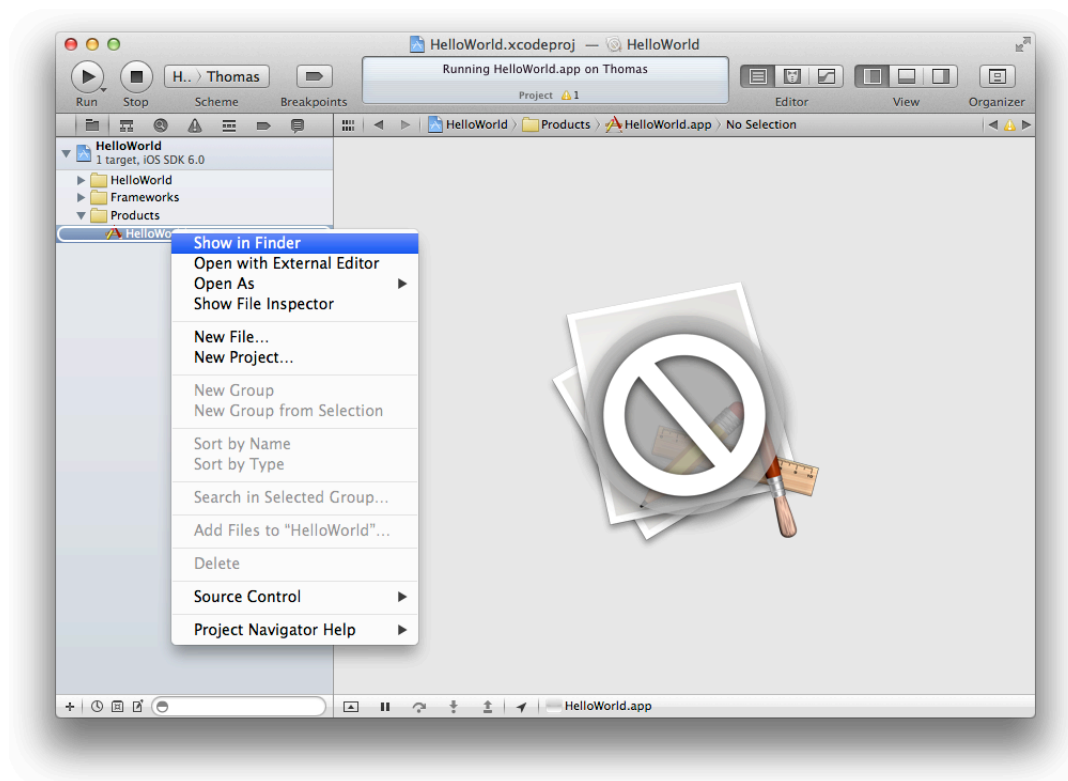
6. In the Finder, drag the app binary to Terminal.
7. Press Return.

For example, the output for an iOS app that is enabled for iCloud key-value storage contains the `keychain-access-groups` entitlement key. The output for a Mac app that is enabled for App Sandbox contains the `com.apple.security.app-sandbox` entitlement key.

If the app's entitlements are different than what you have configured, verify that the embedded provisioning profile is correct. First, you need to locate the embedded provisioning profile.

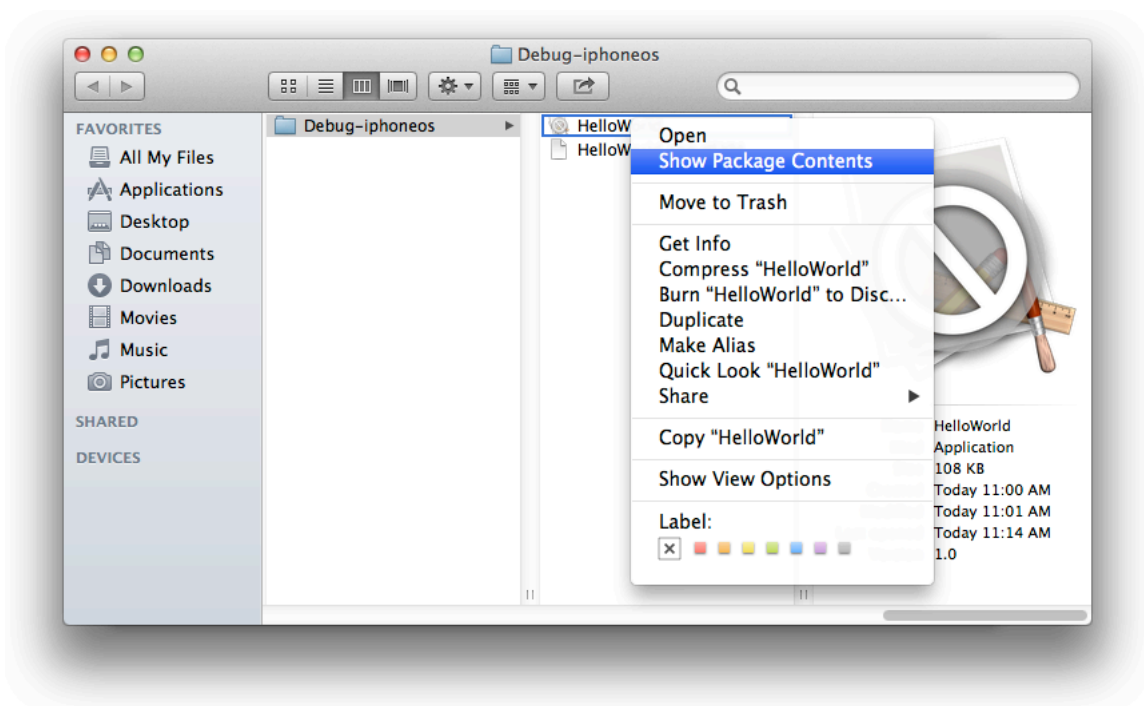
### To locate the embedded provisioning profile in the app binary

1. In Xcode, select your project in the project navigator.
2. Click the disclosure triangle next to the project to reveal the contents.
3. Click the disclosure triangle next to Products to reveal the binary.
4. Control-click the binary and select "Show in Finder" to go to the Xcode build location in the Finder.



5. In the Finder, Control-click the binary and select Show Package Contents from the menu.

For iOS apps, a provisioning profile called `embedded.mobileprovision` appears in the Finder window. For Mac apps, the embedded file is called `embedded.provisionprofile`.

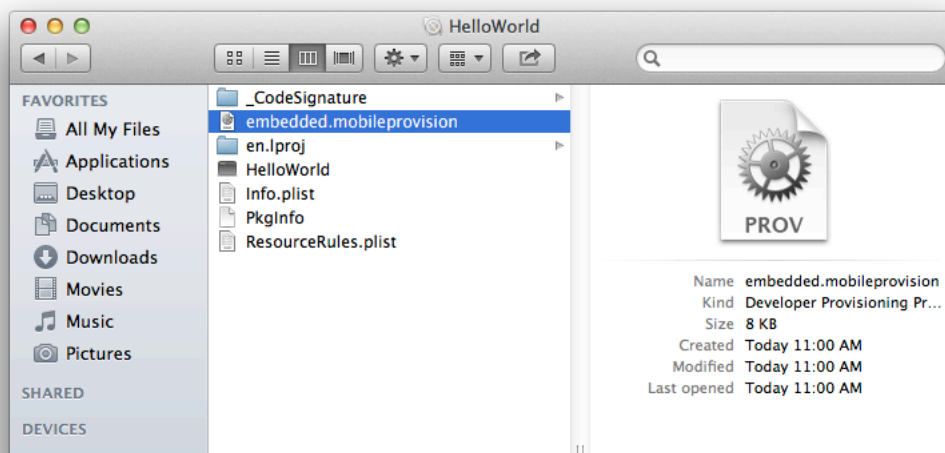


### To verify the entitlements of the embedded provisioning profile

1. Launch Terminal (located in ~/Applications/Utilities) and enter this text (do not press Return):

```
security cms -D -i
```

2. In the Finder, drag the provisioning profile in the app binary to Terminal.



3. Press Return.

This command outputs a property list in XML format.

4. Locate the `Entitlements` key in the output and verify that the `application-identifier` key has the correct entitlements.

For example, the following listing shows an iOS app that is enabled for data protection, Passbook, and iCloud. iCloud entitlements begin with the text `com.apple.developer.ubiquity`.

```
<key>Entitlements</key>
  <dict>
    <key>application-identifier</key>
    <string>AYUS77G8A4.com.gitakumar.touchfighter</string>
    <key>com.apple.developer.default-data-protection</key>
    <string>NSFileProtectionComplete</string>
    <key>com.apple.developer.pass-type-identifiers</key>
    <array>
      <string>AYUS77G8A4.*</string>
    </array>
    <key>com.apple.developer.ubiquity-container-identifiers</key>
    <array>
      <string>AYUS77G8A4.*</string>
    </array>
  </dict>
</key>
```



```
</array>
<key>com.apple.developer.ubiquity-kvstore-identifier</key>
<string>AYUS77G8A4.*</string>
<key>get-task-allow</key>
<true/>
<key>keychain-access-groups</key>
<array>
  <string>AYUS77G8A4.*</string>
</array>
</dict>
```

See [codesign](#) and [security](#) for more ways to use these commands.

## Recap

In this chapter you learned how to configure store technologies in Xcode and iTunes Connect. Some technologies required that you set entitlements and edit the information property list in your Xcode project, add a framework to your Xcode project, or configure your app in iTunes Connect.

# Configuring Your Xcode Project for Distribution

Before you distribute your app for testing or submit it to the store, complete the configuration of your Xcode project. When you create a new Xcode project, you choose a template that most closely corresponds to the kind of app you want to make, and you customize some initial settings for the app. Later, you can change required app information and assets as needed for the store. You set entitlements depending on the capabilities of your app. For example, all Mac apps submitted to the store need to have sandboxing enabled. Some Apple technologies require a combination of code you write and data you configure to operate properly. Some of the data you configure about your app is distributed between your Xcode project, Member Center, and iTunes Connect. This chapter explains the properties stored in the Xcode project that should be set before you distribute your app.

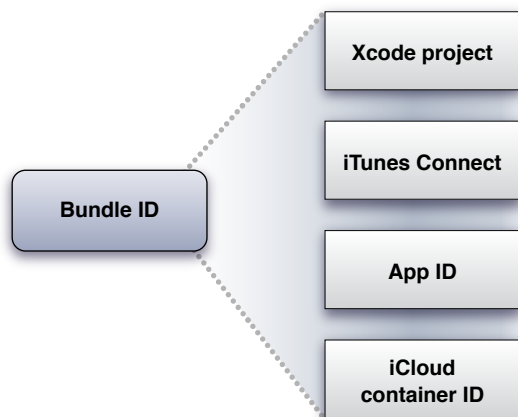
## About Bundle IDs

A **bundle ID** precisely identifies a single app. A bundle ID is used during the development process to provision devices and by the operating system when the app is distributed to customers. For example, the preferences system uses this string to identify the app for which a given preference applies. In contrast, Launch Services uses the bundle ID to locate an app capable of opening a particular file, using the first app it finds with the given identifier. The bundle ID is also used to validate an app's signature.

The bundle ID string must be a uniform type identifier (UTI) that contains only alphanumeric characters (A–Z, a–z, 0–9), hyphen (–), and period (.). The string should be in reverse-DNS format. For example, if your company's domain is `Ajax.com` and you create an app named Hello, you could assign the string `com.Ajax.Hello` as your app's bundle ID.

During the development process, you use an app's bundle ID in many different places to identify the app. Figure 6-1 shows the most common places where an app's bundle ID is used during the development process.

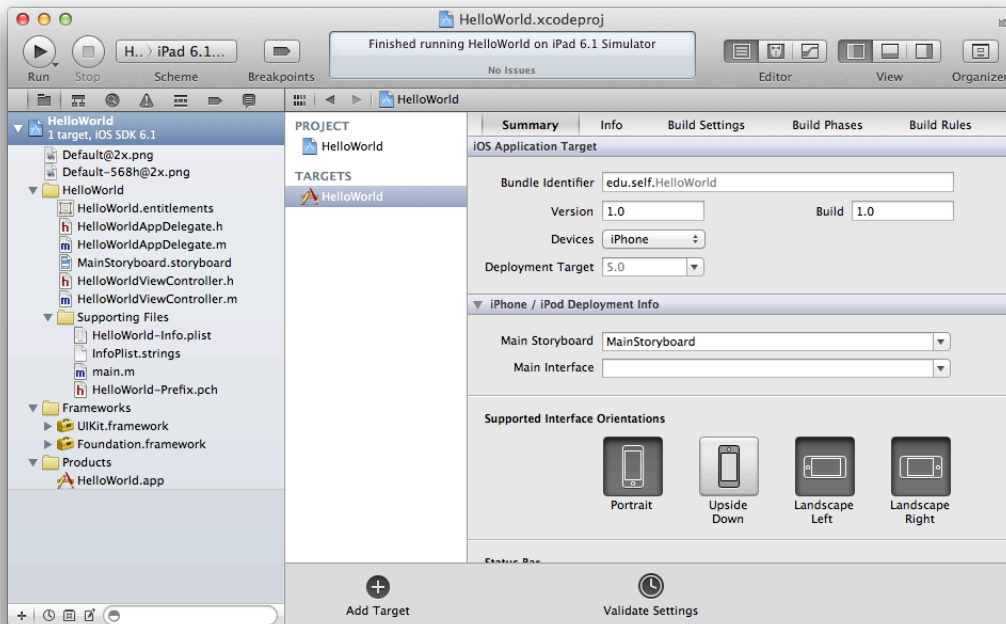
Figure 6-1 Common uses for an app's bundle ID



- The bundle ID itself is stored in the information property list file (`Info.plist`) inside your project. This file is later copied into your app's bundle when you build the project.
- When you are ready to publish an app, you use the bundle ID to identify the app in iTunes Connect. The store submission process correlates the bundle ID from the app you submit with the data you provide in iTunes Connect.
- When developing your app, you need a development provisioning profile with an App ID that is compatible with the app's bundle ID. However, unlike domain names, bundle IDs are case sensitive. If the App ID is lowercase, your bundle ID needs to be lowercase, too.
- When you implement iCloud support in your app, the container IDs you specify are based on the bundle IDs of one or more apps.

## Before You Begin

Most of the options discussed in this chapter, including enabling entitlements, are located in the project editor for your target. In Xcode, choose View > Navigators > Show Project Navigator to open the project navigator. Select your target in the Targets section of the second sidebar to display the project editor. Click the Summary, Info, or Build Settings tab to view these properties.



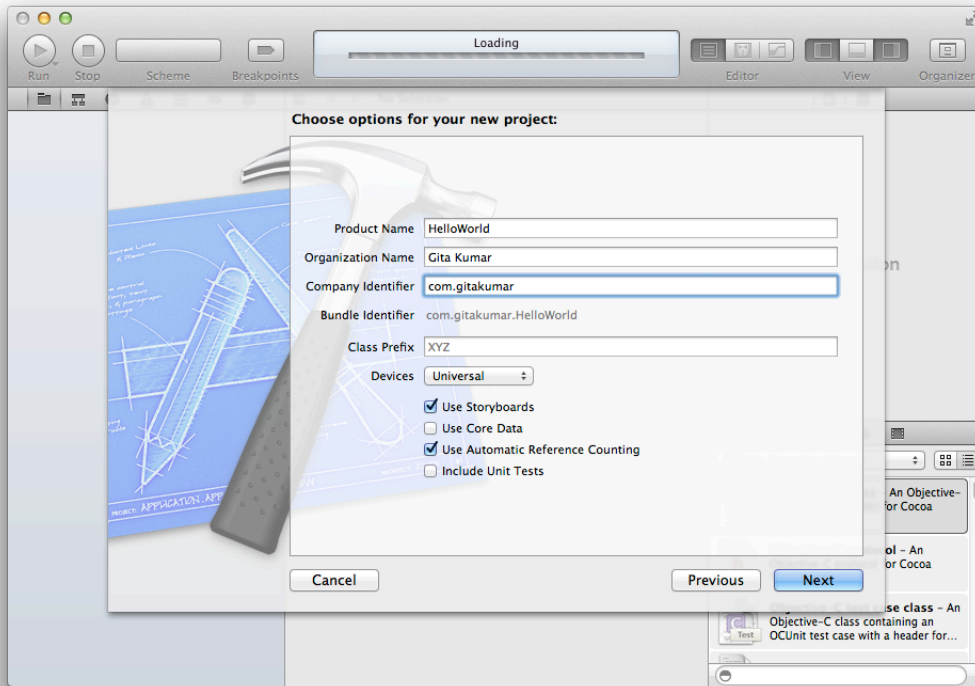
## Setting Properties When Creating Your Xcode Project

An assistant guides you through the process of creating an Xcode project. First, you select a template for your project. Starting with the right template helps speed the development process. The assistant also prompts you to enter information about your app that is used to determine your app's capabilities and distribute it to customers. If you do not have this information when you create the project, you can set these properties later. Some of the data in the Xcode project is similar to the data you enter in iTunes Connect, but only the bundle ID needs to match the bundle ID you enter in iTunes Connect before you can submit your app to the store.

**Note:** If you do not have an Xcode project, read the tutorial in *Start Developing iOS Apps Today* or *Start Developing Mac Apps Today* to learn how to create one. You cannot perform the tasks in this book without first creating an Xcode project that builds and runs an app.

---

For iOS apps, you see a dialog similar to this when creating an Xcode project from a template:



The **product name** is the name of your app as it will appear to customers and should match the app name you enter later in iTunes Connect. It is the name that will appear in Springboard when the app is installed. The product name cannot be longer than 255 bytes and can be no fewer than 2 characters and should not contain any spaces.

The **organization name** is an attribute of the Xcode project and is used in boilerplate text throughout your project folder. For example, the organization name is used in the source and header file copyright strings. The organization name in your Xcode project is not the same as the company name that you enter later in iTunes Connect.

The product name and **company identifier** you enter are concatenated together to create the default bundle ID using reverse domain name (reverse DNS) notation. The **bundle ID** needs to be unique to your app.

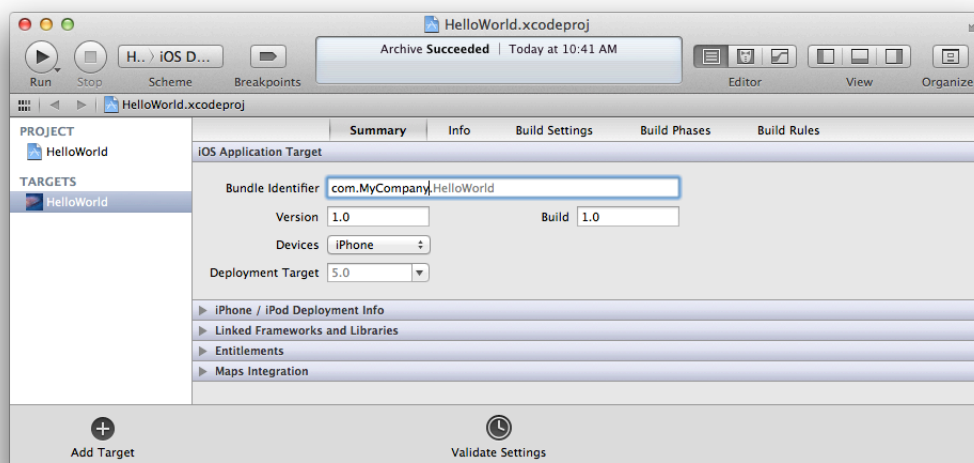
For iOS apps, you can select the types of devices you support from the Devices pop-up menu. For Mac apps, you can select the Mac App Store categories from a pop-up menu.

The other values used by the Xcode template are sufficient for building and running your app locally, but you'll need to finalize properties, such as the bundle ID, later. Also, the assistant doesn't set all required properties for the store. You will need to complete the basic store configuration before you submit. Ideally, you will complete this configuration before you distribute your app for testing too.

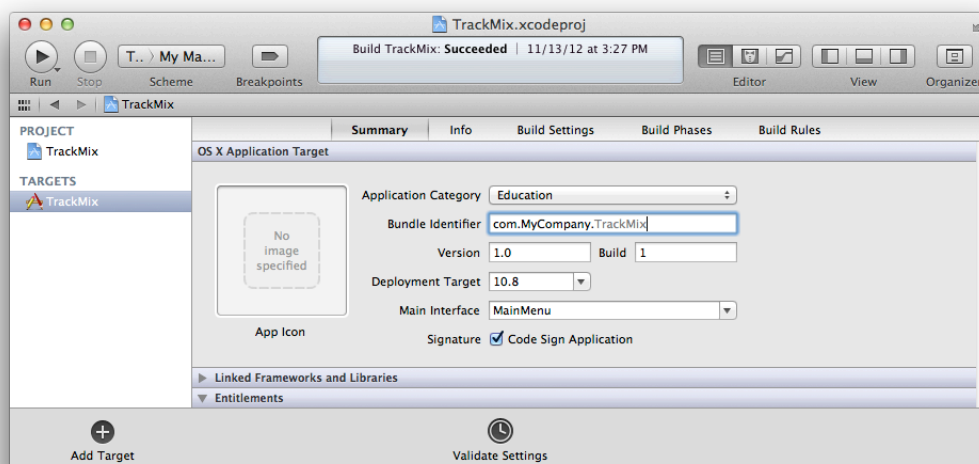
After your app is released, you cannot change some of this metadata, so you want to choose your settings carefully. Refer to "Editing and Updating App Information" in *iTunes Connect Developer Guide* to learn which app states cause some properties to be locked in iTunes Connect.

## Configuring Application Target Settings

The application target settings appear in the target's Summary pane. To display the Summary pane, select the target in the project navigator and click the Summary tab. For iOS apps, the application target settings appear in the iOS Application Target section.



For Mac apps, they appear in the OS X Application Target section.



## Setting the Mac Application Category

Set the category under which your app will be listed on the Mac App Store. The category you select should match the category you later select in your [iTunes Connect](#) app record.

### To set the application category

1. In the project navigator, select the project and your target to display the project editor.
2. Click the Summary tab.
3. Choose the category from the Application Category pop-up menu.

iOS app categories are set in [iTunes Connect](#) only. Read “Choosing Primary and Secondary Categories” in *iTunes Connect Developer Guide* for more details on app categories.

## Setting the Bundle ID

The bundle ID (called a *bundle identifier* in Xcode) is used by Xcode, the operating system, and the store to uniquely identify an app. A Mac app and iOS app cannot share the same bundle ID either. The bundle ID is also used to match a team’s App ID and any associated provisioning profiles. Later, the bundle ID needs to be the same as the bundle ID you enter in iTunes Connect.

### To set the bundle ID

1. In the project navigator, select the project and your target to display the project editor.

2. Click the Info tab.
3. Enter the bundle ID in the Value column of the “Bundle identifier” row.

The default bundle ID in your Xcode project is a string formatted as a reverse-domain—for example, `com.MyCompany.MyProductName`. The Xcode project template uses the Product Name build setting, which defaults to your app name, as the product name in this string. For example, the bundle ID for an app called TrackMix resolves to `com.MyCompany.TrackMix`. So it is sufficient to replace the `com.MyCompany` in the “Bundle identifier” row with your domain name.

For Mac apps, ensure that every bundle ID is unique within your app bundle. For example, if your app bundle includes a helper app, ensure that you do not include two copies of a framework that is used by both your app and the helper app.

## Setting the Version Number and Build String

The version number is a two-period-separated list of positive integers, as in `4.5.2`. The first integer represents a major revision, the second a minor revision, and the third a maintenance release.

The version number is shown in the store and that version should match the version number you enter later in iTunes Connect. The build string represents an iteration (released or unreleased) of the bundle and can contain a mix of characters and numbers. For example, the build is shown in the About window of a Mac app. The user clicks the version number on the About window to toggle between the version number and the build string. For details on possible values, see “`CFBundleShortVersionString`” in *Information Property List Key Reference* and “`CFBundleVersion`” in *Information Property List Key Reference*.

For iOS apps, you should update the build string whenever you distribute a new build of your app for testing. iTunes will recognize that the build string changed and properly sync the new iOS App Store Package to the device. Read “[Beta Testing Your iOS App](#)” (page 117) for how to configure your app for testing.

Set the version number and build string in the Summary pane in the project editor.

### To set the version number and build string

1. In the project navigator, select the project and your target to display the project editor.
2. Click the Summary tab.
3. Enter the version number in the Versions text field, and enter the build string in the Build text field.

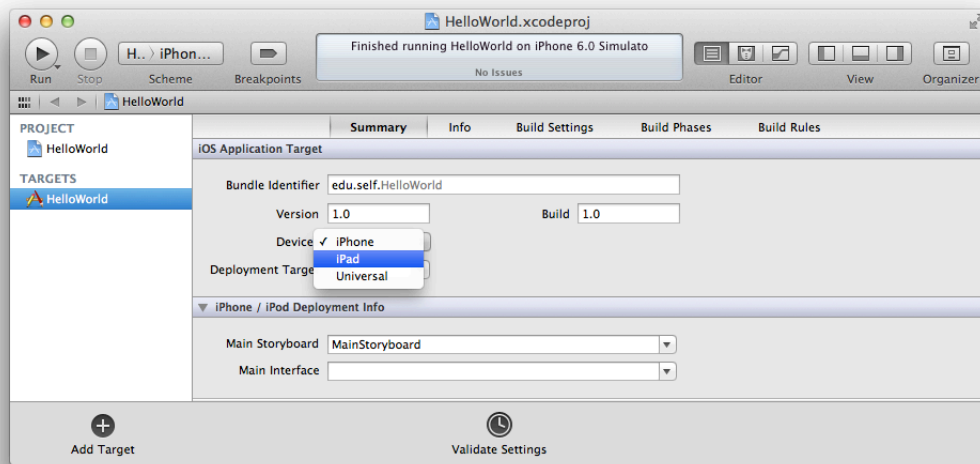


## Setting the Target iOS Devices

The Devices setting identifies the type of devices you want an iOS app to run on. There are two device types: iPhone and iPad. The iPhone type includes iPhone and iPod touch devices. The iPad type includes all iPad and iPad mini devices.

### To set the target devices

1. In the project navigator, select the project and your target to display the project editor.
2. Click the Summary tab.
3. From the Devices pop-up menu, choose iPhone, iPad, or Universal (to target both families).



For more information on configuring your app for iPhone, iPad, or both device families, see “Advanced App Tricks” in *iOS App Programming Guide*.

## Setting the Deployment Target

The deployment target setting specifies the lowest operating-system version that your app will run on. For example, the lowest available setting for iPad apps is iOS 3.2.

There are several strategies for choosing the deployment target when developing your app. Each version of iOS or OS X includes features and capabilities not present in earlier versions. As new versions are published, some users may upgrade immediately, while other users may wait before moving to the latest version. You can target the latest version taking full advantage of all the new features but limiting the app to only users

running the latest version. Or you can target an earlier version making your app available to more users but limiting the features you can use in the app. Another approach is to target an earlier version but use weak linking to determine at runtime whether later version features are available before using them.

For details on weak linking, read “Weak Linking and Apple Frameworks” in *SDK Compatibility Guide*.

### To set the target version

1. In the project navigator, select the project and your target to display the project editor.
2. Click the Summary tab.
3. Choose the version you want to target from the Deployment Target pop-up menu.

Xcode sets the Minimum System Version key in the app’s information property list to the deployment target you choose. When you publish your app to the store, it uses this property value to indicate which versions your app supports.

---

**Note:** The SDK version, not the deployment target, determines which features you can use in an app. If the SDK you’re using to build the app is more recent than the app’s deployment target, Xcode displays build warnings when it detects that your app is using a feature that’s unavailable in the deployment target.

You must also ensure that the symbols you use are available in the app’s runtime environment. To check for their availability, use the techniques described in *SDK Compatibility Guide*.

---

## Adding App Icons and Launch Images

App icons and launch images are stored in the app bundle, not uploaded as separate assets to iTunes Connect. The operating system uses these images in various locations on a device to represent your app. In general, artwork displayed by the operating system resides in the bundle, and artwork displayed by iTunes is uploaded to iTunes Connect.

For all artwork, keep the file size as small as possible for a positive purchase experience for your users. For iOS apps, see “Custom Icon and Image Creation Guidelines” in *iOS Human Interface Guidelines* for the sizes of all required app icons, launch images, and other icons. See “Creating Great Icons for Any Resolution” in *OS X Human Interface Guidelines* for all the required Mac app icons. This table includes icon sizes that may be used on the Mac App Store. To take advantage of Retina displays, provide high-resolution images for each device you support.

Read “Before You Begin” in *iTunes Connect Developer Guide* for the specification of screenshots that you upload later using iTunes Connect.

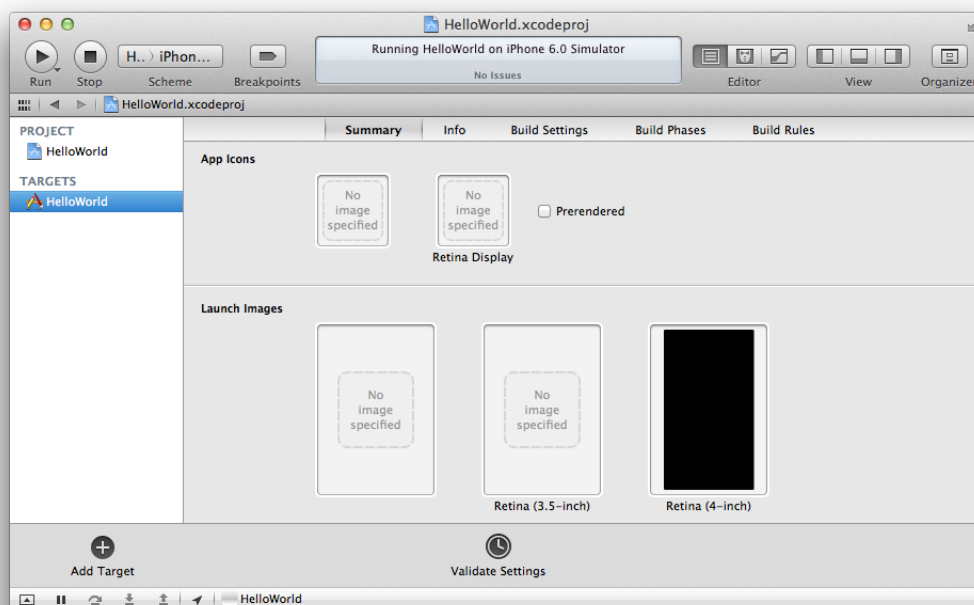
## Setting App Icons

Your app needs an app icon to represent it and pass validation tests. For iOS apps, read “App Icons” in *iOS App Programming Guide* to create separate icon files for different device resolutions. For Mac apps, read “Provide High-Resolution Versions of All App Graphics Resources” in *High Resolution Guidelines for OS X* to create your icon file, which needs to be in ICNS format.

### To add an app icon

1. In the project navigator, select the project and your target to display the project editor.
2. Click the Summary tab.
3. Drag the icon file to the App Icon image well.

For iOS apps, the App Icon image wells are located in the iPhone/iPod Deployment Info section in the Summary pane. For Mac apps, the single App Icon image well is located in the OS X Application Target section at the top of the pane.



## Creating and Setting iOS Launch Images

Launch images are displayed while your app is launching on iOS. A launch image matching the device resolution appears as soon as the user taps your app icon. Use screenshots to create your app’s launch images.

**Note:** Although the launch image includes the status bar as it looked when the screenshot was captured, iOS replaces it with the current status bar when your app launches.

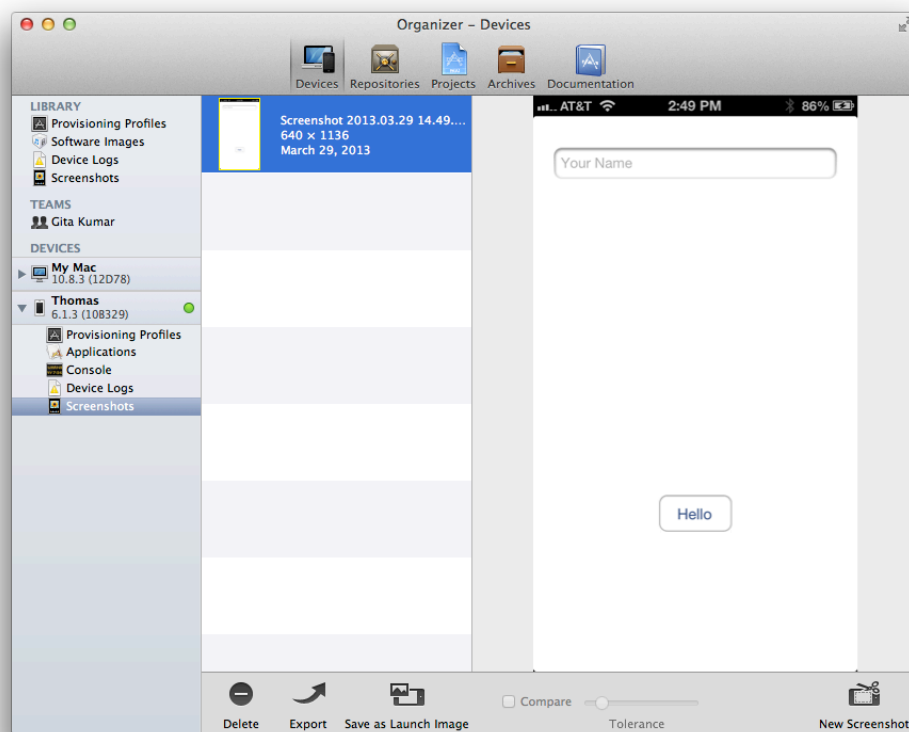
---

## Capturing Screenshots on Your Device and Setting Launch Images Directly in Xcode

Follow these steps to capture screenshots of your app while it's connected to your Mac.

### To capture a screenshot on your connected iOS device

1. On the device, configure the screen the way you want it.
2. In the Devices organizer, select Screenshots in the device.
3. Click New Screenshot.

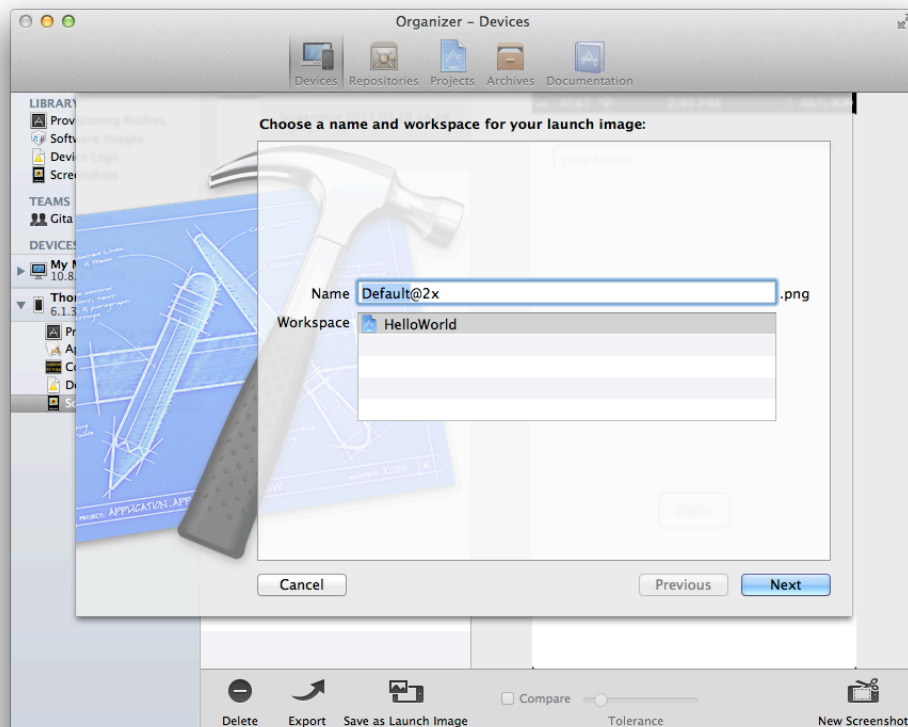


After you capture a screenshot, you can make it your app's launch image.

### To set a screenshot as your iOS app's launch image

1. In the Devices organizer, select Screenshots for a device or for the Library section.

2. Select an image.
3. Click “Save as Launch Image.”
4. Specify the name of the image and the target app, and click Next.



To get a PNG file of the screenshot, you drag it from the Devices organizer to the desktop.

## Capturing Screenshots Directly on Your Device

Alternatively, you can capture screenshots directly on your device and import them into your Mac using the iPhoto application.

To capture a screenshot on your device, you press the Lock and Home buttons simultaneously. Your screenshot is saved in the Saved Photos album in the Photos app.

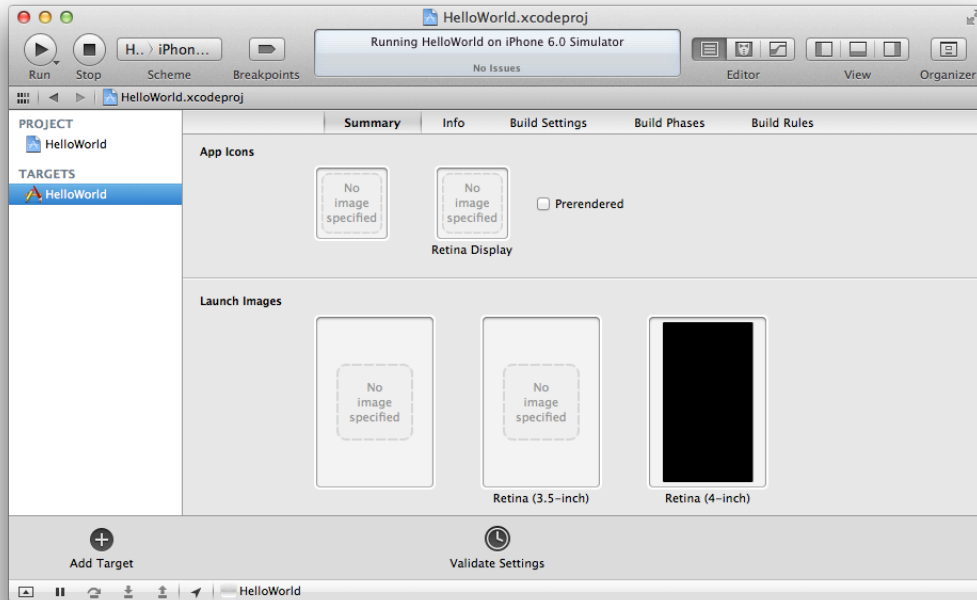
Copy the screenshot from the device to your Mac and follow the steps in “Setting Launch Images in the Project Editor” to set the launch image.

## Setting Launch Images in the Project Editor

Alternatively, you can set the launch images in the project editor.

## To set launch images in the project editor

1. In the project navigator, select the project and your target to display the project editor.
2. Click the Summary tab.
3. Scroll down to the Launch Images section under iPhone/iPad Deployment Info.
4. Drag the launch images to the corresponding image wells.



## Configuring Entitlements

To protect your app from being compromised by a hacker who might damage the user's system, you give permissions, known as *entitlements*, to your app to perform specific functions. An entitlement is a key-value pair whose value you can set to specify a capability or security permission. This chapter describes entitlements you configure in your Xcode project for certain technologies supported by the store. Refer to *Entitlement Key Reference* for a complete list of app entitlements.

You configure entitlements for each target in the Xcode project for your product. For example, if you have a main app and multiple helper apps in one Xcode project, you need to configure entitlements for each target in the project. When you enable entitlements, Xcode adds a file with entitlement key-value pairs to your target.

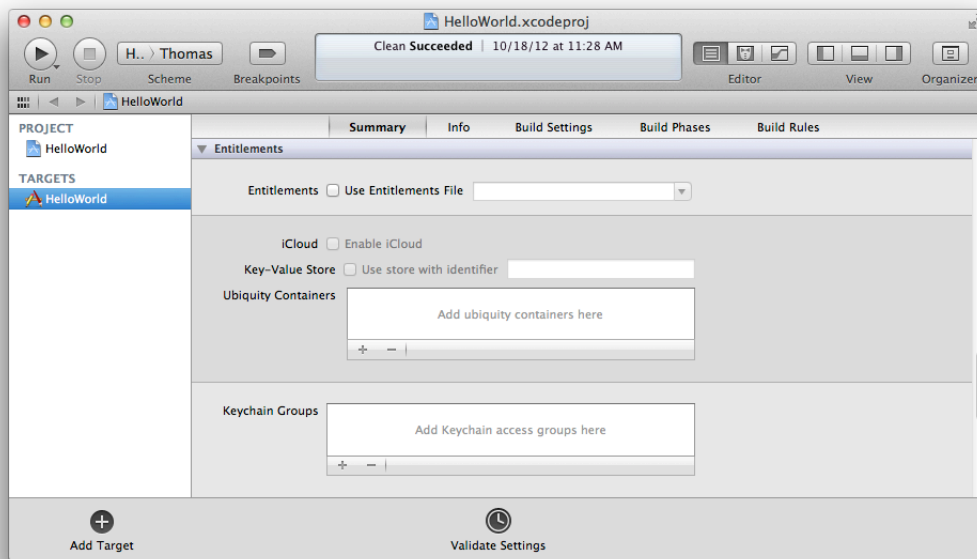
The default name of the file is your project name with the extension `.entitlements`. You can configure entitlements—for example, for iCloud storage or sandboxing—using either the property list editor or the project editor to edit this file.

For entitlements to take effect, you need to code sign your app, as described in [“Creating Your Signing Certificates”](#) (page 24). Some technologies, such as iCloud and Game Center, require further configuration. You need a provisioning profile containing an App ID that enables these technologies. Read [“Configuring Store Technologies in Xcode and iTunes Connect”](#) (page 79) for complete steps on how to configure specific technologies.

Before you can configure entitlements for specific technologies, you need to enable entitlements.

### To enable entitlements

1. In Xcode, select the target in the project editor.
2. Select the Summary tab and scroll down to the Entitlements section.



3. Select the Use Entitlements File option.  
Xcode adds an entitlements file to your project and automatically enters default values for some entitlements.
4. If you want to change the entitlements file name, enter a different suffix in the Use Entitlements File text field.

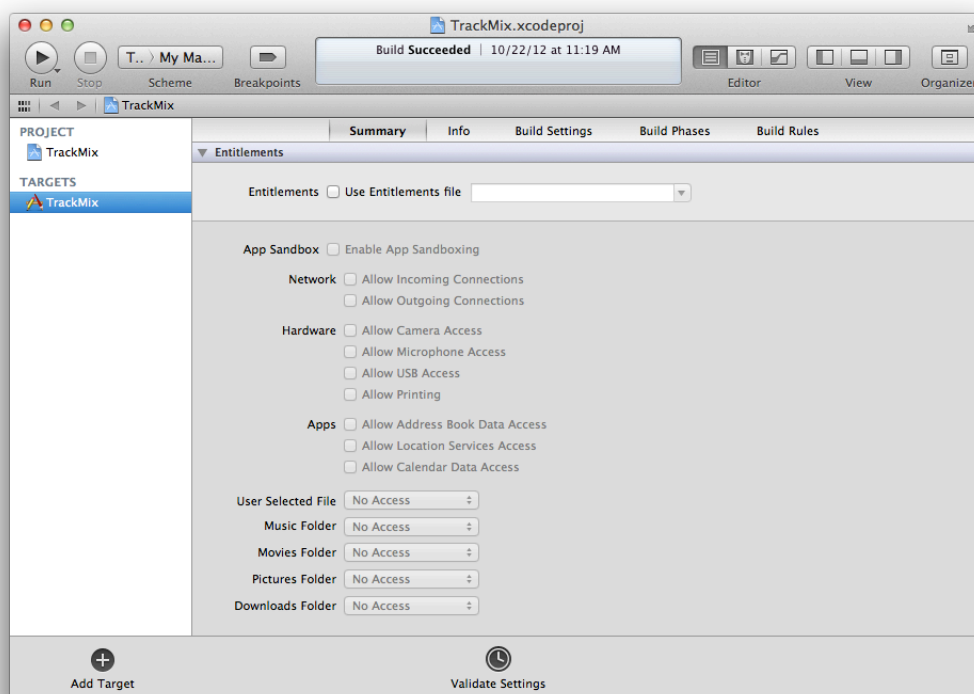
## Configuring App Sandbox for Mac Apps

Sandboxing provides the last line of defense against stolen, corrupted, or deleted user data if malicious code exploits your app. It also minimizes damage from coding errors in your app or in frameworks you link against. Simply enabling sandboxing provides the maximum level of restrictions on how an app can interact with the rest of the system. All apps submitted to the Mac App Store are required to use sandboxing. Therefore, if you plan to submit your app to the Mac App Store, you should enable sandboxing during development.

You configure sandboxing by enabling this feature and then optionally granting permission for specific types of functions.

### To configure App Sandbox

1. In Xcode, select the target in the project editor.
2. Select the Summary tab and scroll down to the Entitlements section.

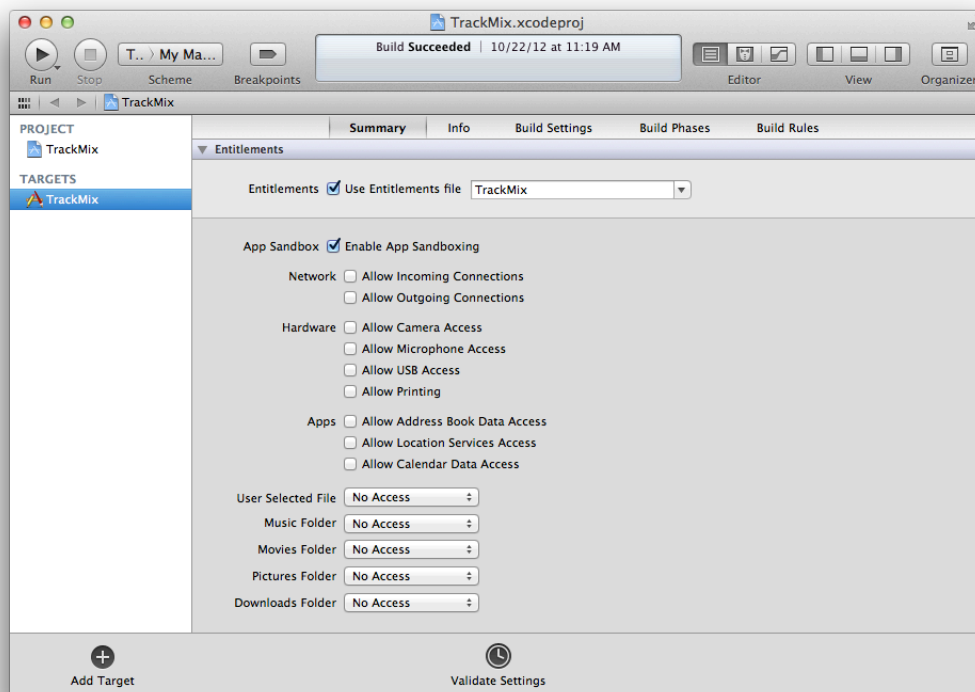


3. Select the Use Entitlements File option.

Xcode adds an entitlements file to your project and automatically enters default values for some entitlements. When you enable entitlements, App Sandbox is automatically enabled.



4. Select Enable App Sandboxing in the App Sandbox section.



5. Use the remaining App Sandbox entitlements to describe the minimum set of capabilities the target needs to do its job.

For a complete description of App Sandbox entitlements, refer to *Entitlement Key Reference*. If you are enabling sandboxing for an existing app, read *App Sandbox Design Guide* to learn the new locations a sandboxed app can access.

## Editing the Information Property List

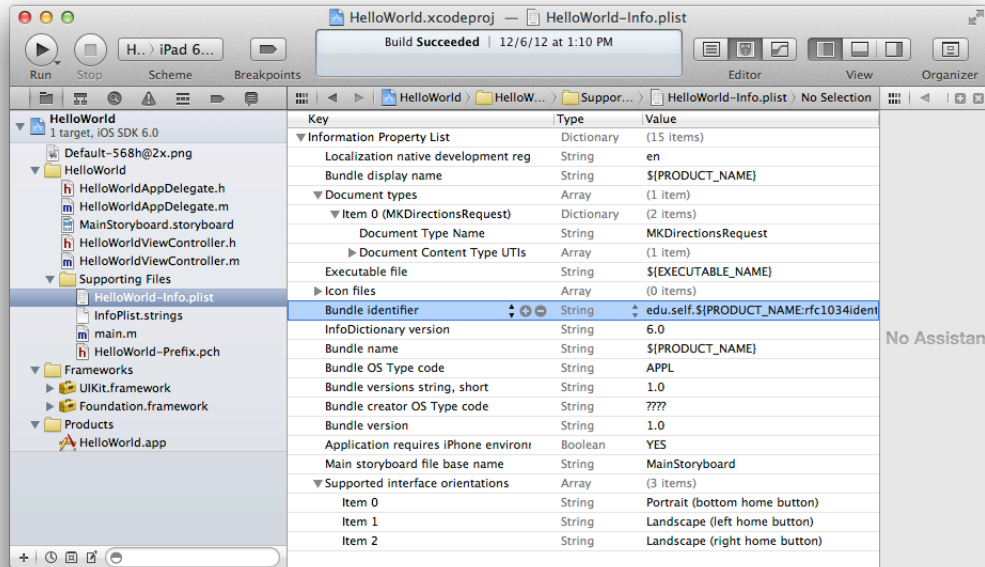
Occasionally, you may need to edit the information property list directly to set keys that don't appear elsewhere in Xcode. The name of this file is typically *ProjectName-Info.plist*.

### To edit the information property list

1. In Xcode, select the project in the project navigator.
2. Click the disclosure triangle next to the *ProjectName* folder to reveal its contents.
3. Click the disclosure triangle next to the Supporting Files subfolder to reveal its contents.

4. Select the `ProjectName-Info.plist` file.

The information property list is displayed to the right in a property list editor.



5. Double-click in the value column and in the row of the key you want to edit.
6. Enter a new value for the key.

Refer to *Property List Editor Help* for how to edit other cells in a property list.

## Setting the Copyright Key for Mac Apps

Make sure that your information property list file contains a valid value for the `Copyright` key. For details on possible values, see “`NSHumanReadableCopyright`” in *Information Property List Key Reference*.

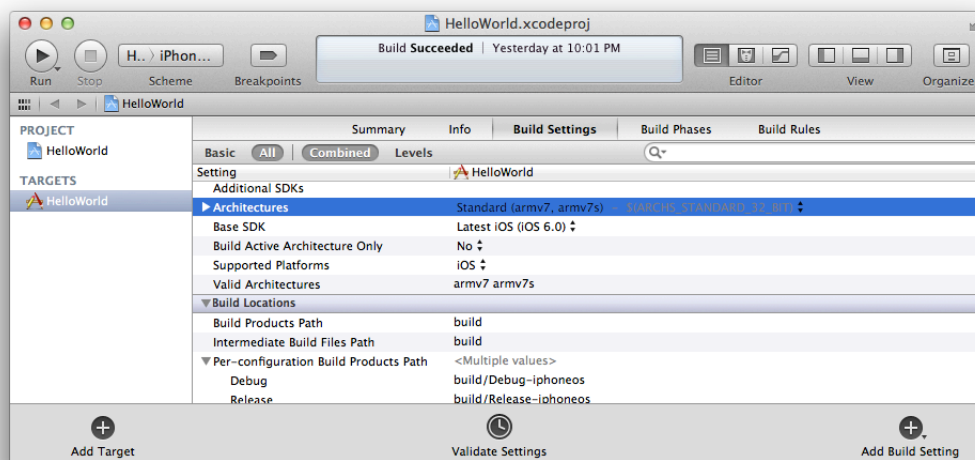
## Specifying Build Settings

There are miscellaneous build settings that you might need to configure for the store as well. You do this in the Build Settings pane of the project editor.

### To edit a build setting

1. In the project editor, select the project or target whose build setting you want to edit.

2. Click Build Settings at the top of the project editor.



3. Locate the build setting in the left column, or type the name of the build setting in the search field in the upper-right corner.
4. Click All if some build settings fail to appear.
5. Set the value for the build setting in the right column.

## Setting Architectures for iOS Apps

The Architectures build setting identifies the architectures for which your app is built. An iOS device uses one of a set of architectures, which include `armv7` and `armv7s`. You have two options for specifying the value of this setting:

- **Standard.** Produces an app binary with a common architecture, compatible with all supported iOS devices. This option generates the smallest app, but it may not be optimized to run at the best possible speed for all devices.
- **Other.** Produces an app binary for a specified set of architectures.

If you choose Other from the Architectures build-setting value list, click the plus button (+) to enter the custom iOS-device architecture names you support.

**Important:** The store rejects a binary that supports only `armv7s`. If `armv7s` is included in the Architectures list, then `armv7` must also be included.

## Setting the Base SDK

The Base SDK version number must be greater than or equal to the software version number on your development device; otherwise, Xcode cannot initiate a debugging session with the device. The Base SDK for your project and targets should be set to the latest operating system, which is the default value. The Base SDK property is located in the Architectures area in the Build Settings pane. For iOS apps, set Build SDK to Latest iOS. For Mac apps, set the Build SDK to Latest OS X. If you choose another value, you need to download and install the latest SDK version that is greater than or equal to your device software version.

## Setting the Debug Information Format for Mac Apps

Set the Debug Information Format build setting to “DWARF with dSYM.”

## Recap

In this chapter, you learned how to create your Xcode project from a template and configure it for distribution to the store.

# Beta Testing Your iOS App

You've created your iOS app in Xcode and added store technologies. You've tested your app on your own devices and on iOS Simulator. It's time for beta testing. In this phase, you distribute the app to a wider audience—to give the app a “real-world” test and, in some cases, to offer testers a preview of your next version.

You should rigorously test your app on a variety of devices and iOS versions because different kinds of devices and iOS releases have different capabilities. It's not sufficient to test your app on a device provisioned for development or the simulator. iOS Simulator doesn't run all threads that run on devices, and launching apps on devices through Xcode disables some of the watchdog timers. At a minimum, test the app on all devices you support and have available. For example, if you have a game that runs only on iPhone and iPod touch, you should test on those devices and not on iPad. In addition, keep prior versions of iOS installed on devices for compatibility testing. If you don't support certain devices or iOS versions, you need to indicate this in the project target settings in Xcode.

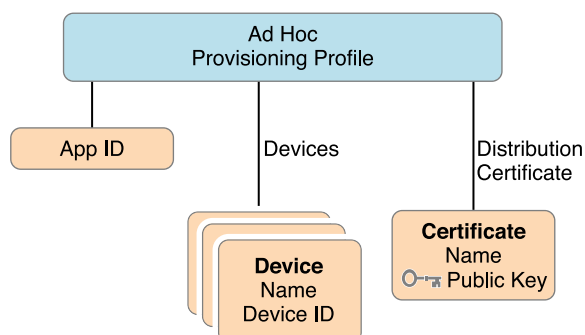
To distribute your app for beta testing:

1. Optionally, create an iTunes Connect app record.
2. Register all test devices.
3. Create a distribution certificate.
4. Create an ad hoc provisioning profile.
5. Archive and validate your app.
6. Create an iOS App Store Package.
7. Install the app on a test devices.
8. Solicit crash reports from testers.

## About Ad Hoc Provisioning Profiles

An **ad hoc provisioning profile** is a distribution provisioning profile for iOS apps that allows your app to be installed on designated devices and use store technologies without the assistance of Xcode. It is one of the two types of distribution provisioning profiles you can create for iOS apps (you use the other type of distribution provisioning profile later to submit your app to the store). An ad hoc provisioning profile ensures that test versions of your app are not copied and distributed without your knowledge.

When you are ready to distribute your app to testers, you create an ad hoc provisioning profile specifying an App ID that matches one or more of your apps, a set of test devices, and a single distribution certificate.



Each iOS device in an ad hoc provisioning profile is identified by its unique **device ID (UDID)**. The devices you register and add to a provisioning profile are stored by Member Center. Each individual or company can register up to 100 devices for development and testing.

You sign the iOS App Store Package containing your app using the distribution certificate specified in the ad hoc provisioning profile and distribute it to testers.

## Creating Your App Record in iTunes Connect

If you are beta testing a final candidate for a release, be sure to validate the app before distributing it to testers. The validation tests are performed by iTunes Connect, which checks whether your Xcode project is configured correctly for the store. For example, it reports a problem if you are missing required app icons. Your iTunes Connect app record needs to be in the “Waiting for Upload” or later state to validate the archive. To create your app record and change it to the “Waiting for Upload” state, read [“Creating an App Record”](#) (page 153) before continuing.

## Registering Test Devices

To register test devices, collect device IDs from testers and add them to Member Center.

Testers can get their device ID using iTunes. (They do not need to install Xcode to do this.) Send the instructions in [“Locating Device IDs”](#) (page 171) using iTunes to testers and ask them to send their device IDs to you, or follow these steps to collect your own device IDs.

In Member Center, register one or more devices, as described in [“Registering Devices Using Member Center”](#) (page 170).

## Creating Distribution Certificates

You need a distribution certificate before you can create an ad hoc provisioning profile. If you did not create a distribution certificate during development, you can create it now by opening the Devices organizer in Xcode and selecting “Refresh from Developer Portal” from the Editor menu. If you are missing your distribution certificate, Xcode offers to request an iOS Distribution certificate on your behalf. Click Submit Request when this dialog appears.

---

**Note:** You must be an individual developer or a company team admin or agent to create a distribution certificate. If you have a company membership, read [“Managing Your Team”](#) (page 184) for a description of team roles and tasks team agents perform on behalf of team members.

---

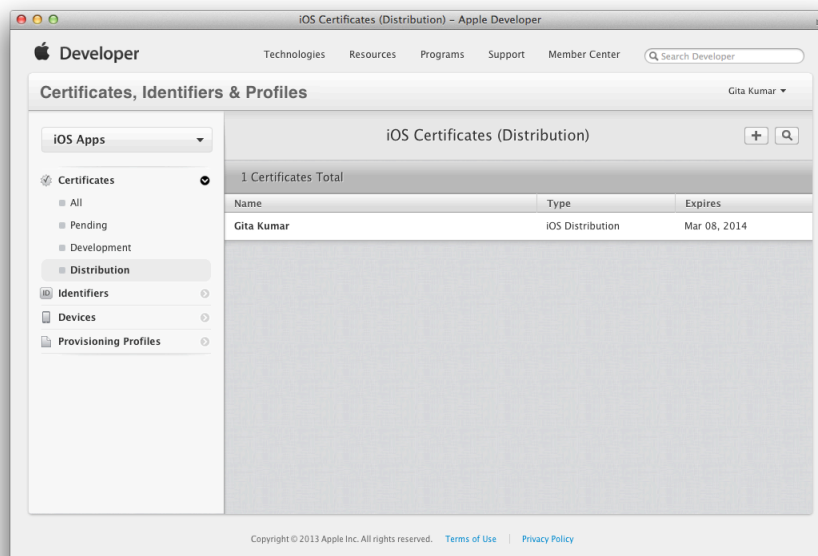
## Verify Your Steps

Before continuing, verify that the distribution certificate in Member Center matches the certificate in Xcode. The certificate must be valid to sign your app.

### To verify distribution certificates using Member Center

1. In [Certificates, Identifiers & Profiles](#), select Certificates.
2. Under the Certificates section, select Distribution.

The names, types, and expiration dates of the development certificate should match the information that you view in Xcode and Keychain Access.

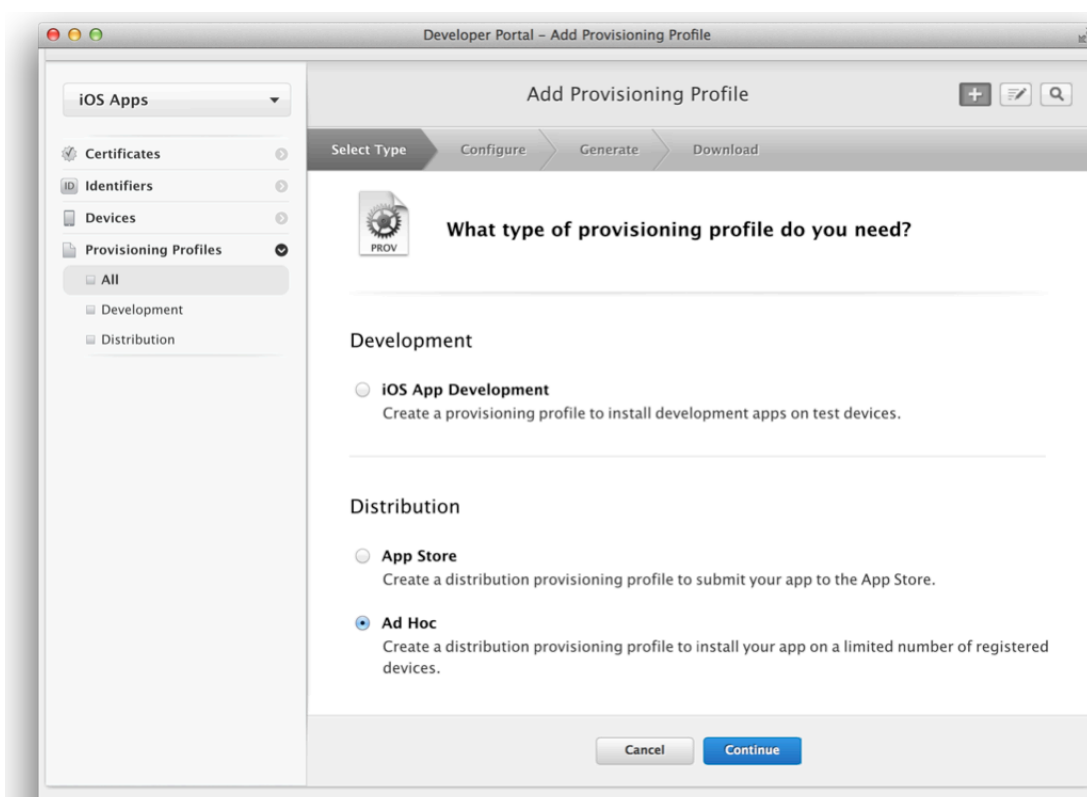


## Creating Ad Hoc Provisioning Profiles

Now you're ready to create an ad hoc provisioning profile, which allows testers to run your app on their device without needing Xcode. To create an ad hoc provisioning profile, you select an App ID, a single distribution certificate, and multiple test devices.

### To create an ad hoc provisioning profile

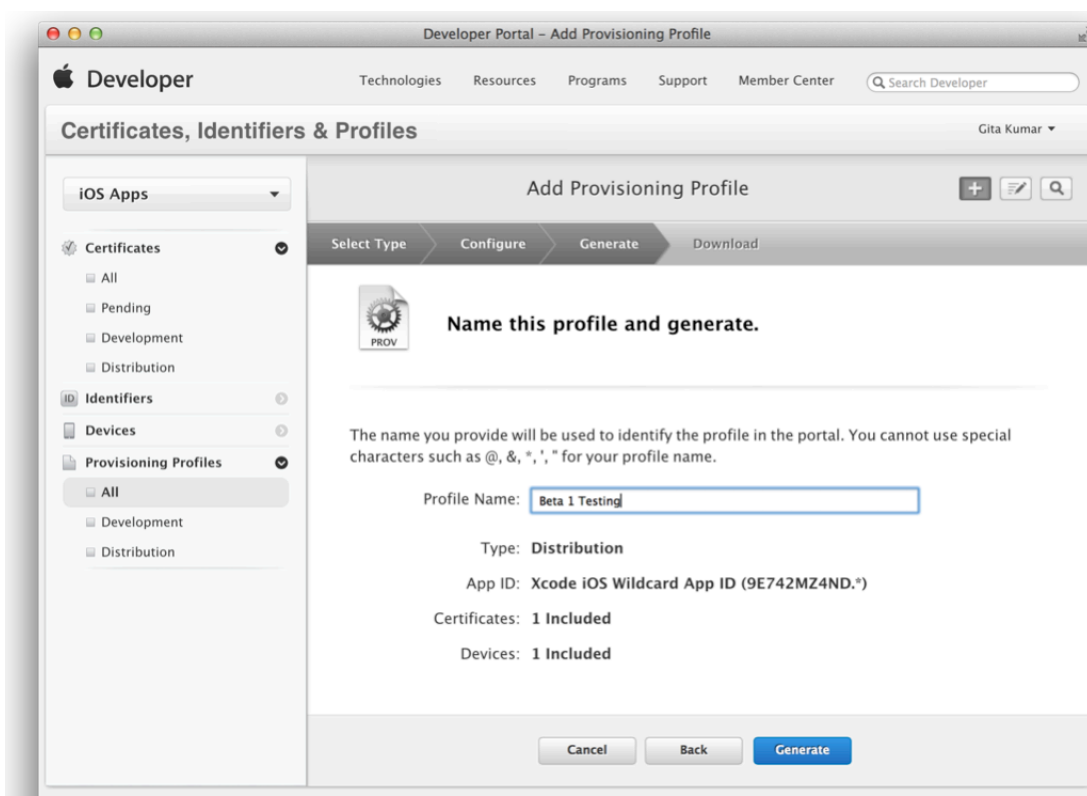
1. In [Certificates, Identifiers & Profiles](#), select Provisioning Profiles.
2. Under Provisioning Profiles, select All.
3. Click the plus button (+) in the upper-right corner.
4. Select Ad Hoc as the distribution method, then click Continue.



5. Select your App ID.  
Select the App ID you used during development. If you did not configure any store technologies, you can use Xcode iOS Wildcard App ID as the App ID.
6. Select the distribution certificate you want to use, and click Continue.



7. Select the devices you want to use for testing, and click Continue.
8. Enter a profile name, and click Generate.



After the profile is generated, you can download and use it.

In Xcode, select "Refresh from Developer Portal" from the Editor menu to download the provisioning profile.

## Archive and Validate Your App

You create an archive of your app regardless of the type of distribution method you choose. Xcode archives allow you to build your app and store it, along with critical debugging information, in a bundle that is managed by Xcode. For example, if you distribute a build of your app to testers, archiving the debugging information makes it easier to interpret crash reports that they send you later. After your app is released, you use the same debugging information to decipher crash reports that you download from iTunes Connect.

**Important:** Save the archive for any version of an app you distribute to users. You need the debugging information stored in the archive to decipher crash reports later.

With archives you can distribute the same build to the store that you distribute for testing. It's important to test the exact build that is a candidate for release. Differences between Xcode build settings can cause bugs that don't appear during testing, because the binary being tested was built differently. By having Xcode make an archive of your app, you can be sure you are testing the *exact* same build of your app that you submit to the store.

Follow these steps to archive and validate your app:

1. Set the code signing identity to your distribution certificate.
2. Review the Archive scheme settings.
3. Create and validate an archive of your app.

## Code Signing Your App

You set the Code Signing Identity build setting to the distribution certificate in the ad hoc provisioning profile. The app is code signed when you create the archive in the following step.

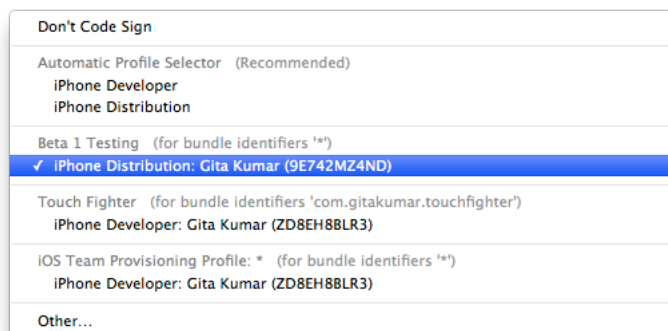
### To set the code signing identity to your distribution certificate

1. In the project editor, select the target.

**Important:** If you want to sign multiple targets with the same code signing identity, select the project, not a target.

2. Select the Build Settings tab.
3. Click All.
4. Type `Code Signing` in the search field in the Build Settings pane of the project editor.
5. From the Code Signing Identity pop-up menu (in the ad hoc provisioning profile section), choose your distribution certificate.

The distribution certificate begins with “iPhone Distribution:” followed by your team name. If you are an individual developer, your team name is the same as your name.



## Troubleshooting

If your distribution certificate or ad hoc provisioning profile doesn't appear in the Code Signing Identity menu when you code sign your app, read [“Your Provisioning Profile Doesn't Appear in the Code Signing Identity Menu”](#) (page 208).

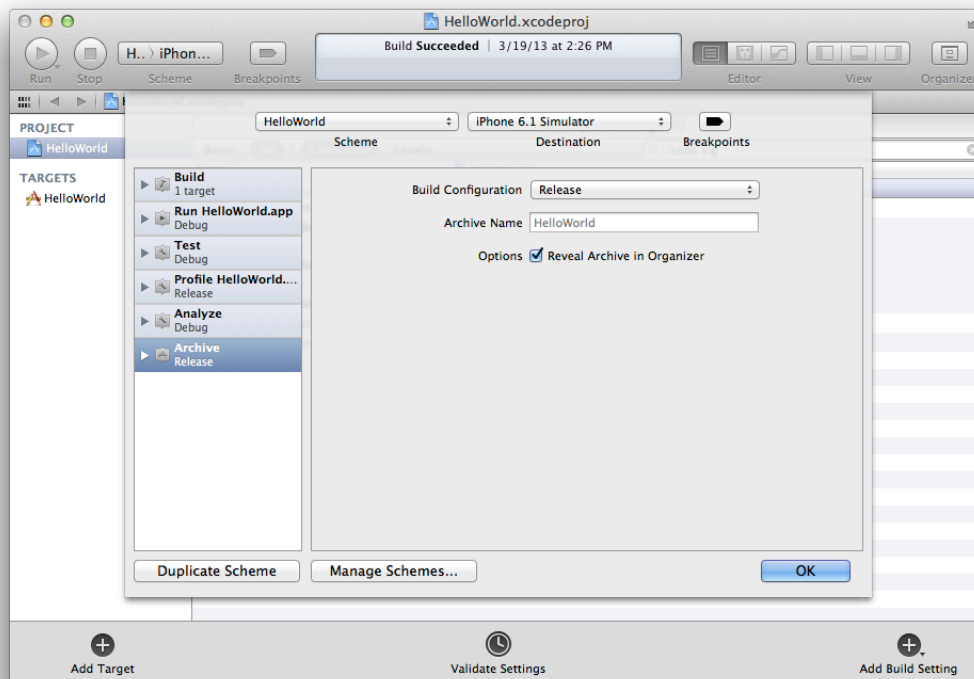
## Review the Archive Scheme Settings

Before you make an archive, review the Archive scheme settings to ensure that you don't archive a debug version of your app.

### To review the Archive scheme

1. In the Xcode project editor, choose Product > Scheme > Edit Scheme to open the scheme editor.

2. Click Archive in the column on the left.



3. Choose Release from the Build Configuration pop-up menu and click OK.

## Creating and Validating an Archive

Next, create an archive of your app. Xcode stores this archive in the Archives organizer. Optionally, validate your archive (run iTunes Connect tests) after creating it.

### To create an archive

1. In the Xcode project editor, choose iOS Device or your device name from the scheme toolbar menu. You cannot create an archive of a simulator build. If an iOS device is connected to your Mac, the device name appears in the scheme toolbar menu. When you disconnect the iOS device, the menu item changes to iOS Device.
2. Choose Product > Archive.

The Archives organizer appears and displays the new archive.



You need to validate the archive before you can submit it to the store so it's best to validate your beta version now to discover issues early. The app record in iTunes Connect must be in the "Waiting for Upload" or later state in order for you to validate your app, as described in ["Creating Your App Record in iTunes Connect"](#) (page 118). If you are not ready to create your app record, you can skip this validation step.

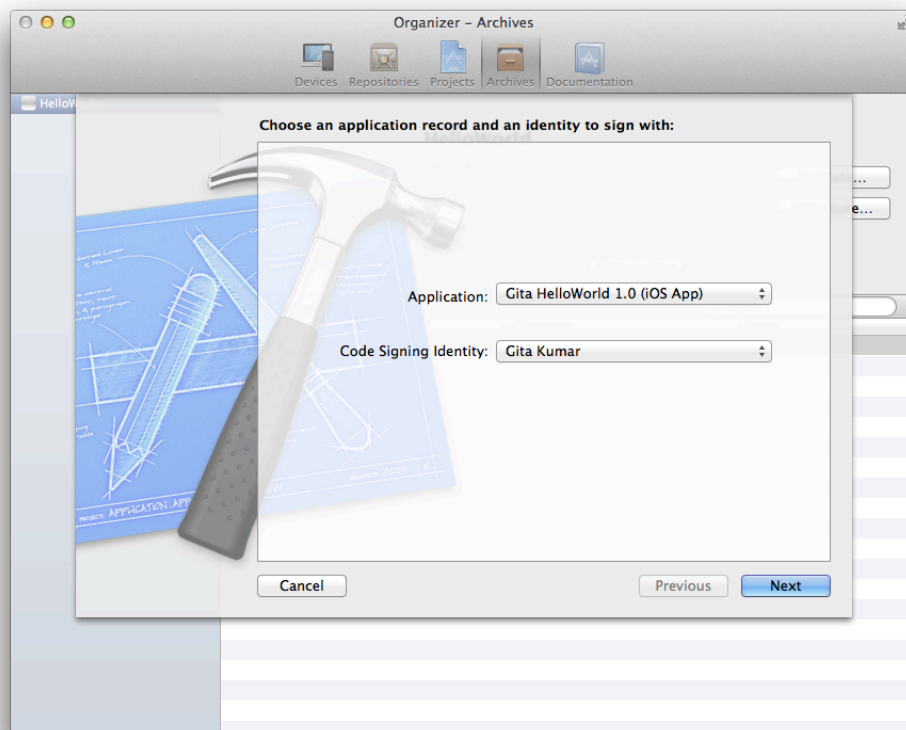
### To validate an archive

1. In the Archives organizer, select the archive.
2. Click the Validate button.
3. Enter your iTunes Connect credentials and click Next.

If a dialog appears stating that no application record can be found, create an app record in iTunes Connect before continuing.

4. Select the app you want to distribute and the appropriate signing identity, and click Next.

Select the code signing identity that appears under the ad hoc provisioning profile you created in a previous step.



iTunes Connect runs validation tests.

5. Review validation issues found, if any, and click Finish.

Optionally, fix any validation issues, create a new archive, and validate it again. Validation errors will not prevent you from distributing your app for beta testing.

## Troubleshooting

After you enter your iTunes Connect credentials, if a dialog appears stating that no application record can be found, create an app record in iTunes Connect before validating your app. Read [“Creating an App Record”](#) (page 153) for how to create an app record.

If your distribution certificate doesn't appear in the Code Signing Identity menu when you validate the archive, read [“Your Provisioning Profile Doesn't Appear in the Code Signing Identity Menu”](#) (page 208).

## Creating an iOS App Store Package

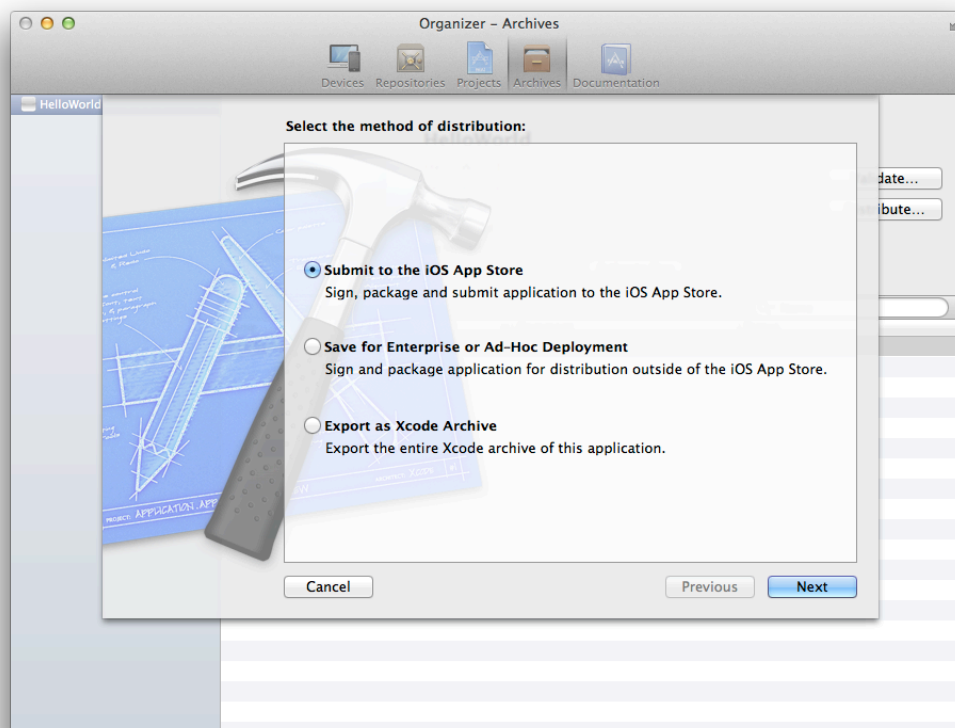
Testers won't have Xcode to run your app, so you want to create an **iOS App Store Package** that they use to install your app on their device. You use Xcode to create an archive and generate an iOS App Store Package (a file with a `.ipa` filename extension) from the archive.

### To create an iOS App Store Package for testing on devices

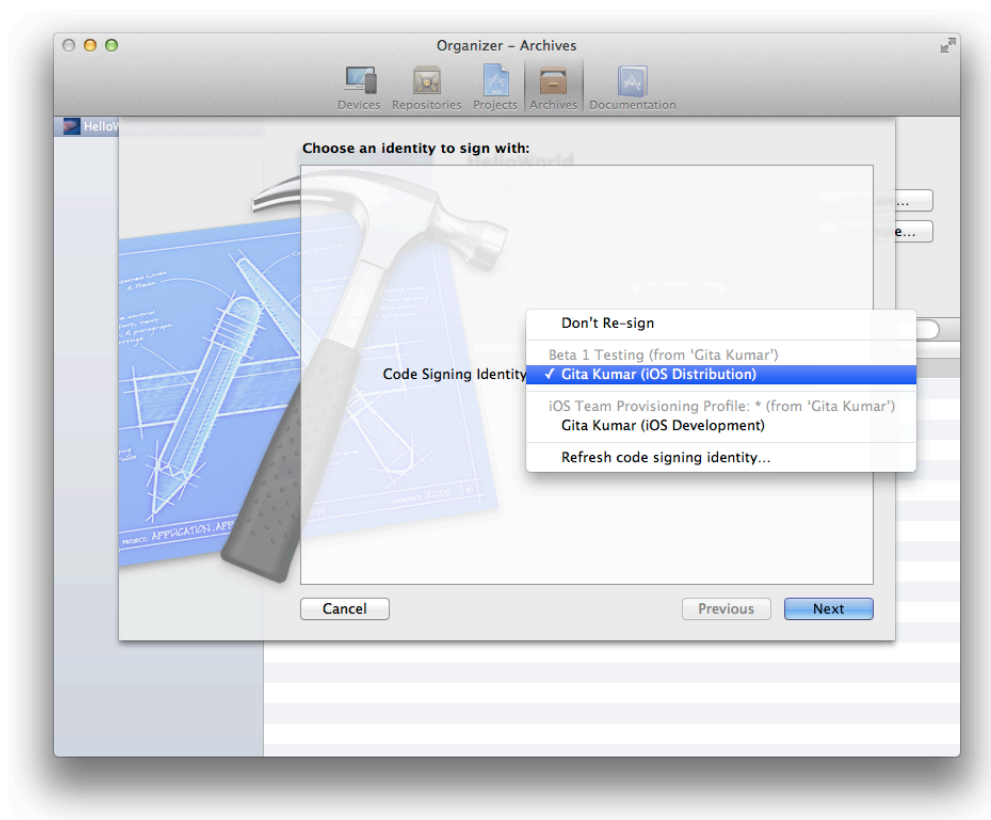
1. In the Archives organizer, select the archive.
2. Optionally, click the Validate button.

If you are close to submitting your app to the store, you should validate it now and fix any problems before distributing it for final testing. For a complete list of issues you should fix before distributing your app, read *App Store Review Guidelines for iOS Apps*.

3. Click the Distribute button.
4. Select "Save for Enterprise or Ad-Hoc Deployment" and click Next.



5. Choose your distribution certificate (the one contained in your ad hoc provisioning profile) from the Code Signing Identity pop-up menu, and click Next.



6. Enter a filename and location for the iOS App Store Package file, and click Save.  
The file will have a .ipa extension

---

**Note:** To send an iOS App Store Package to another team member, choose your development certificate contained in a development provisioning profile as the Code Signing Identity when creating the package. The development provisioning profile you choose—for example, the team provisioning profile—needs to contain the team member’s device.

---

## Troubleshooting

If your distribution certificate or ad hoc provisioning profile doesn’t appear in the Code Signing Identity menu when you create the iOS App Store Package, read [“Your Provisioning Profile Doesn’t Appear in the Code Signing Identity Menu”](#) (page 208).



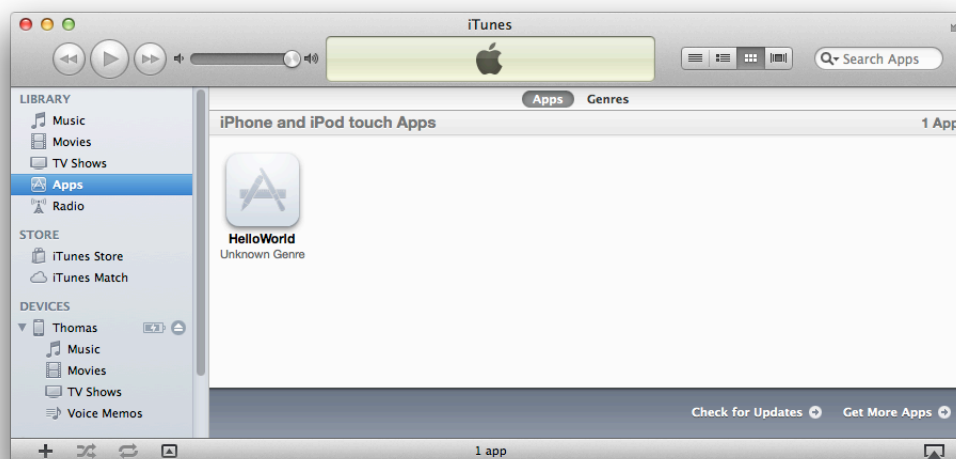
## Installing Your App on Test Devices

Before you distribute your app to testers, follow the steps that testers use to install and run the app on their devices. Use iTunes to install the app on a nondevelopment device. iOS extracts the embedded ad hoc provisioning profile in your app and installs it on the device for you. Then test your app on the device.

Follow these steps to install the app on a testing device.

### To install the app on a device

1. Connect the testing device to a Mac running iTunes.  
Don't use a Mac that you use for development.
2. Double-click the iOS App Store Package file you created earlier (the file with the `.ipa` extension).  
The app appears in the iTunes app list. (You may need to select Apps under Library to see it.)



3. In the Devices section, select the device.
4. Select the Apps tab.
5. Under Apps, choose "Sort by Name" or "Sort by Kind" from the pop-up menu.
6. Click the Install button next to your app.  
The button text changes to "Will Install."
7. Click the Apply button or the Sync button to sync the device.  
This uploads the app to the device so that the user can start testing.

Finally, send the iOS App Store Package file to testers along with these installation and the crash report instructions described next.

## Soliciting Crash Reports from Testers

When an app crashes on a device, iOS creates a record of that event. The next time the tester connects the iOS device to iTunes, iTunes downloads those records (known as crash logs) to the tester's Mac. Testers should send these crash logs to you along with any bug reports. Later, after your app is released, you can also retrieve crash reports of your live app from iTunes Connect.

Tell testers how to retrieve crash reports from their devices and send them to you.

### To send crash reports from a Mac

1. Connect the testing device to a Mac running iTunes.  
iTunes downloads the crash reports to your Mac.
2. In the Finder, choose Go > Go to Folder.
3. Enter `~/Library/Logs/CrashReporter/MobileDevice`.
4. Open the folder identified by your device's name.
5. Select the crash logs named after the app you're testing.
6. Choose Finder > Services > Mail > Send File.
7. In the New Message window, enter the developer's address in the To field and appropriate text in the Subject field.
8. Choose Message > Send.
9. To avoid sending duplicate reports later, delete the crash reports you sent.

### To send crash reports from Windows

1. Enter the crash log directory for your operating system in the Windows search field, replacing `<user_name>` with your Windows user name.
  - For crash log storage on Windows, type:  
`C:\Users\<user_name>\AppData\Roaming\Apple computer\Logs\CrashReporter\MobileDevice`
  - For crash log storage on Windows XP, type:

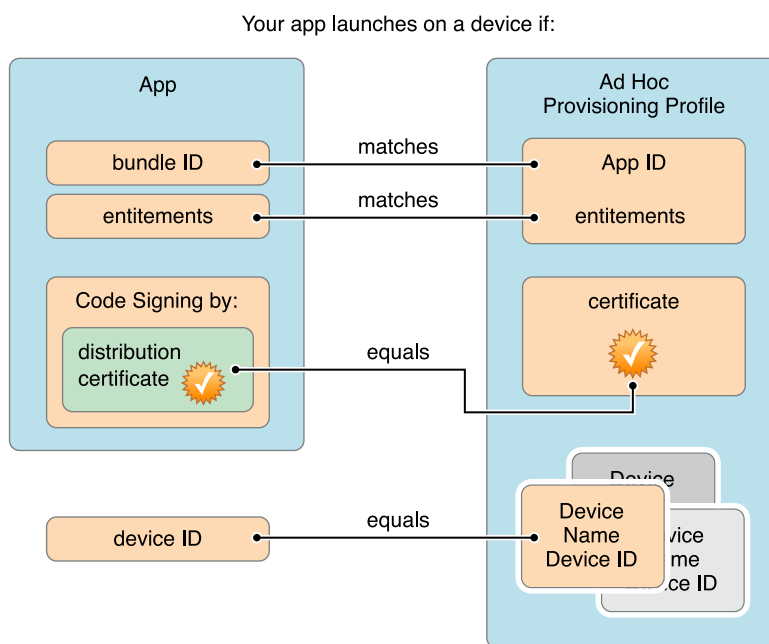
C:\Documents and Settings\\Application Data\Apple  
computer\Logs\CrashReporter

2. Open the folder named after your device's name and send the crash logs for the app you're testing in an email message using the subject-text format <app\_name> crash logs from <your\_name> (for example, MyTestApp crash logs from Anna Haro) to the app's developer.

To learn how to interpret the reports when you receive them from testers, read ["Analyzing Crash Reports"](#) (page 132).

## Ad Hoc Provisioning Profiles in Depth

You signed the iOS App Store Package containing your app using the distribution certificate specified in the ad hoc provisioning profile. Then you installed the provisioning profile and the app on the test device. The app successfully launches if the app's bundle ID matches the App ID, the signature matches the distribution certificate, and the device is in the device list of the ad hoc provisioning profile.



## Recap

In this chapter you learned how to distribute your iOS app for beta testing on designated test devices. You also received instructions to send to testers to install the beta version of your app and send crash reports to you.

# Analyzing Crash Reports

After you distribute your app, you should routinely collect and analyze crash reports. When an app crashes, a **crash report** is created which is very useful for understanding what caused the crash.

You may receive crash reports from multiple sources throughout the lifetime of your app. For iOS apps, you can solicit crash reports from beta testers, as described in [“Soliciting Crash Reports from Testers”](#) (page 130). Similarly, you can collect crash reports for Mac apps during testing. You can also download crash reports from iTunes Connect for an app that is released, as described in [“Viewing Crash Reports”](#) (page 155).

You analyze crash reports using Xcode. To add crash reports to the Devices organizer, drag the crash reports to the Device Logs group in the Library section. You can then view information about the crash such as the stack trace for each execution thread. Xcode automatically **symbolicates** crash logs that you import into the Devices organizer. Xcode replaces memory addresses with human-readable function names and line numbers.

**Important:** For Xcode to symbolicate crash reports (to add to the crash log information about the API used), use Spotlight to index the volume containing your archived apps and their corresponding dSYM files.

For Mac apps, be sure to analyze a crash report using a guest account with a fresh install of the version of OS X that matches the crash report. Do not analyze a crash report using a developer or admin system account, because the problems you want to analyze may not occur.

For how to interpret an iOS crash report, read *Understanding and Analyzing iOS Application Crash Reports*.

Make sure that you are testing the *exact* same build that crashed. You should save all the archives that you distribute for testing and submit to the store. Read *How to Match a Crash Report to a Build* for how to verify if your archive in Xcode matches a crash report. Later, follow these same steps to determine if you are testing the same build that you submitted to the store.

# Submitting Your App

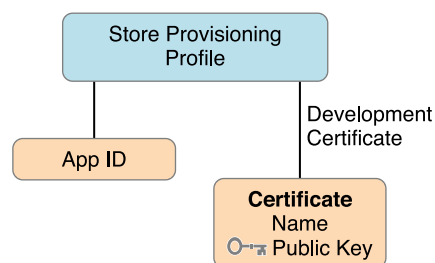
The final step is to code sign and provision your app before you submit it to the store. This is an important step that ensures the submission comes directly from you and only you grant permission for your app to use certain store technologies. Code signing your app or installer package prevents an attacker from submitting a modified version of your app to the store—only someone with the private key for your distribution certificate can submit your app to the store.

Follow these steps to submit your app:

1. Create a distribution certificate.
2. Create a store distribution provisioning profile.
3. Archive and validate your app.
4. For Mac apps, test the Mac Installer Package.
5. Submit your app using Xcode or Application Loader.

## About Store Provisioning Profiles

A **store provisioning profile** is a distribution provisioning profile that authorizes your app to use certain store technologies and ensures that your app is submitted by you. A store distribution provisioning profile contains a single App ID that matches one or more of your apps and a distribution certificate. For iOS apps, you need a store provisioning profile to submit your app. For Mac apps, unless you use store technologies that require provisioning, you need only a distribution certificate.



Configure the App ID before you create the store provisioning profile. You enable and configure technologies for an app by setting entitlements. Some entitlements are enabled for an App ID (a set of apps created by your team) and others are set in the Xcode project. When you submit your app to the store, you select the distribution certificate contained in your store provisioning profile.

## Before You Begin

Before continuing, review the tasks that should be complete before you submit your first binary:

	Task
<input checked="" type="checkbox"/>	Review your Xcode project configuration. Read <a href="#">“Configuring Your Xcode Project for Distribution”</a> (page 98).
<input checked="" type="checkbox"/>	To ensure that your app enables the store technologies you want to use, review your App ID settings. Read <a href="#">“Verify the App ID Settings in Member Center”</a> (page 73).
<input checked="" type="checkbox"/>	Beta test your app on a variety of OS versions and devices. For iOS apps, read <a href="#">“Beta Testing Your iOS App”</a> (page 117).
<input checked="" type="checkbox"/>	Create an app record in iTunes Connect, as described in <a href="#">“Creating an App Record”</a> (page 153). To validate or submit your app, the app record needs to be in the “Waiting for Upload” or later state.

You should use the same App ID you used for development and testing for submitting your app to the store. If you don't use any store technologies that require an explicit App ID, you can use the Xcode wildcard App ID. If you want to create a new App ID, read [“Provisioning Your App for Store Technologies”](#) (page 54). However, if you change your App ID, you need to retest your app before submitting it to the store.

---

**Mac Note:** All apps and their installer packages need to be signed to submit them to the Mac App Store. If you use a helper app, read *Daemons and Services Programming Guide* to learn how to configure it. All Mac apps need to have App Sandbox enabled, too.

---

To streamline the approval process, review the following guidelines and fix any problems before continuing.

- Follow the user interface guidelines in *iOS Human Interface Guidelines* and *OS X Human Interface Guidelines*.
- Review the store guidelines in *App Store Review Guidelines for iOS Apps* and *App Store Review Guidelines for Mac Apps*.

## Creating Distribution Certificates

You need a distribution certificate to submit your app to the store. If you did not create distribution certificates in a previous step during development (see [“Requesting Signing Certificates”](#) (page 26)), you can create them now by opening the Devices organizer in Xcode and selecting “Refresh from Developer Portal” from the Editor menu. If you are missing your distribution certificate, Xcode offers to request one on your behalf. Click Submit Request when this dialog appears. For iOS apps, you’ll need an iOS Distribution certificate and for Mac apps, you’ll need a Mac App Distribution and Mac Installer Distribution certificate. (See [Table 2-1](#) (page 36) for all the types of certificates Xcode requests.)

---

**Note:** You must be an individual developer or a company team admin or agent to create a distribution certificate. If you have a company membership, read [“Managing Your Team”](#) (page 184) for a description of team roles and tasks team agents perform on behalf of team members.

---

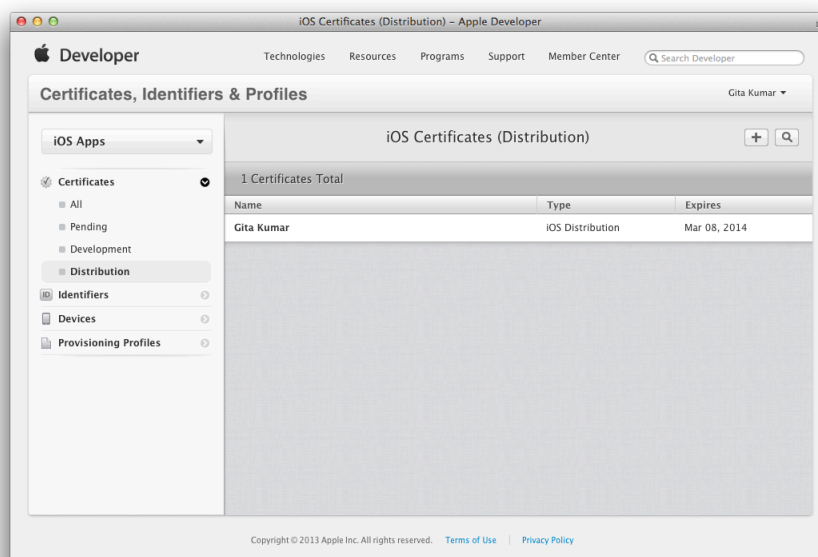
## Verify Your Steps

Before continuing, verify that the distribution certificates in Member Center match the certificates in Xcode. Certificates must be valid in order to sign your app—and for a Mac app, sign your installer package.

### To verify distribution certificates using Member Center

1. In [Certificates, Identifiers & Profiles](#), select Certificates.
2. Under Certificates, select Distribution.

The names and expiration date of the distribution certificate should match the information that you view in Xcode and Keychain Access.



## Creating Store Provisioning Profiles

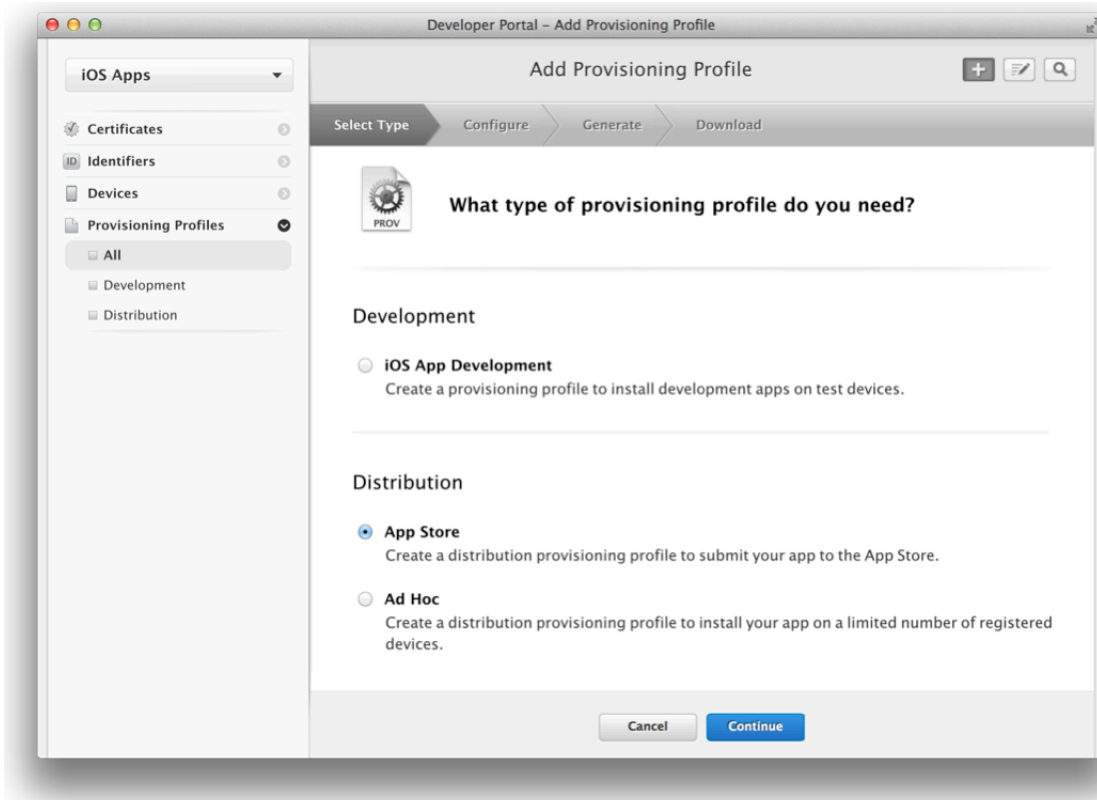
You must provision all iOS apps before submitting them to the store, and you need a distribution provisioning profile to do so. For Mac apps, if you don't use any store technologies that require provisioning (you didn't need a provisioning profile for development), you can skip this step. To create a distribution provisioning profile, you select either the App Store or the Mac App Store as the method of distribution. Then select an App ID and a single distribution certificate. You do not select any devices to create a store provisioning profile.

### To create a store provisioning profile

1. In [Certificates, Identifiers & Profiles](#), select Provisioning Profiles.
2. Click the plus button (+) in the upper-right corner.

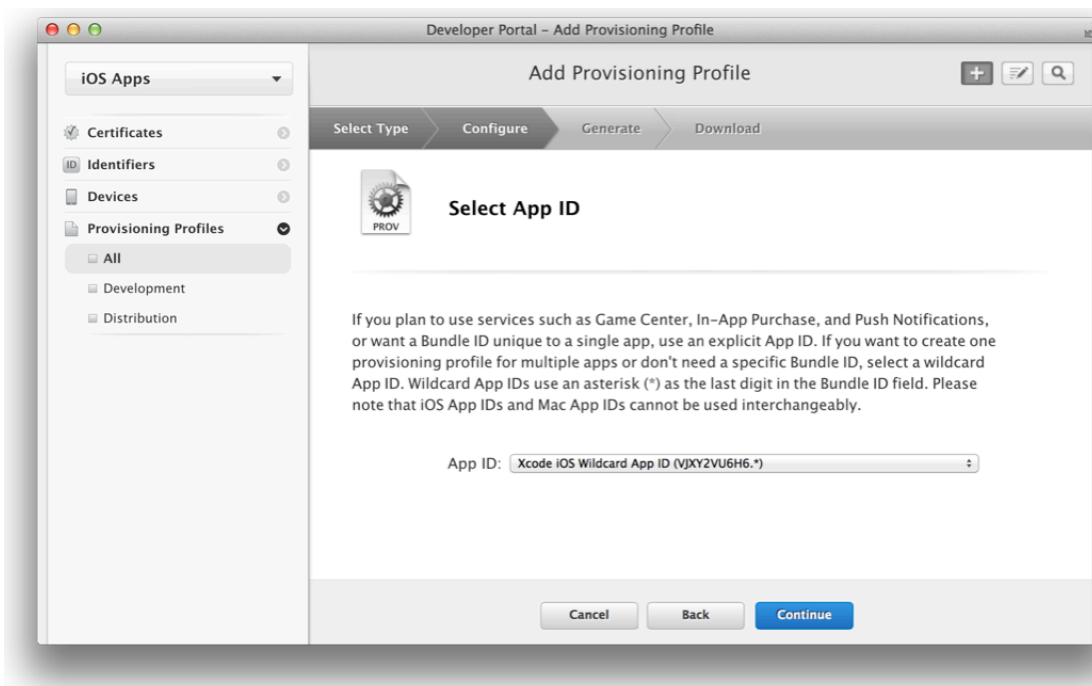


3. Select "App Store" for iOS apps or "Mac App Store" for Mac apps as the distribution method and click Continue.



4. Select the App ID you used for development and click Continue.

If your app doesn't require an explicit App ID, you can select a wildcard App ID.



5. Select your distribution certificate and click Continue.  
A store provisioning profile contains a single distribution certificate.
6. Enter a profile name and click Generate.  
After the profile is generated, you can download and use it.

## Downloading the Distribution Provisioning Profile

Finally, download the provisioning profile in Xcode by choosing “Refresh from Provisioning Portal” from the Editor menu.

## Verify Your Steps

Open the Devices organizer in Xcode and select Provisioning Profiles under Library. Your new distribution provisioning profile should appear. If not, choose “Refresh from Provisioning Portal” from the Editor menu again. Also verify that the provisioning profile appears in the Code Signing Identity build setting menu.

## Archiving and Validating Your App

Just before you submit your app to the store, you create a signed archive of your app and validate it. It is recommended that you submit the last archive you distributed for final testing. Test your final build before submitting it to the store, and if you have not done so, test the archive again for regressions after validating it.

Follow these steps to archive and validate your app:

1. Set the code signing identity to your distribution certificate.
2. Review the Archive scheme settings.
3. Create and validate an archive of your app.
4. If necessary, test your archive before submitting it.

**Important:** Archives allow you to build your app and store it, along with critical debugging information, in a bundle that is managed by Xcode. Save an archive for any version of an app you distribute to users. You need the debugging information stored in the archive to decipher crash reports later.

## Code Signing Your App

For iOS and Mac apps that use provisioning profiles, you sign your app using the distribution certificate in the distribution provisioning profile. For Mac apps that don't need a provisioning profile, you can instead sign the app using the distribution certificate. You set the Code Signing Identity build setting to the distribution certificate in the store provisioning profile. The app is code signed when you create the archive.

---

**Mac Note:** If you import external frameworks, sign the frameworks using the `codesign` command-line tool before Xcode signs the app, as described in *Mac OS X Code Signing In Depth*.

---

### To set the code signing identity to your distribution certificate

1. In the Xcode project editor, select the target.

**Important:** If you want to sign multiple targets with the same code signing identity, select the project, not a target.

2. Select the Build Settings tab.
3. Click All.
4. Type `Code Signing` in the search field in the Build Settings pane of the project editor.

5. From the Code Signing Identity pop-up menu (in the distribution provisioning profile section if you use a provisioning profile), choose the distribution certificate.

For iOS apps, the distribution certificate begins with “iPhone Distribution:” followed by your team name. For Mac apps, the distribution certificate begins with “3rd Party Mac Developer Application:” followed by your team name. If you are an individual developer, your team name is the same as your name.



## Troubleshooting

If your distribution certificate or store provisioning profile doesn't appear in the Code Signing Identity menu when you code sign your app, refresh the provisioning profiles, as described in [“The Code Signing Identity Build Setting Doesn't Match Any Certificates”](#) (page 211).

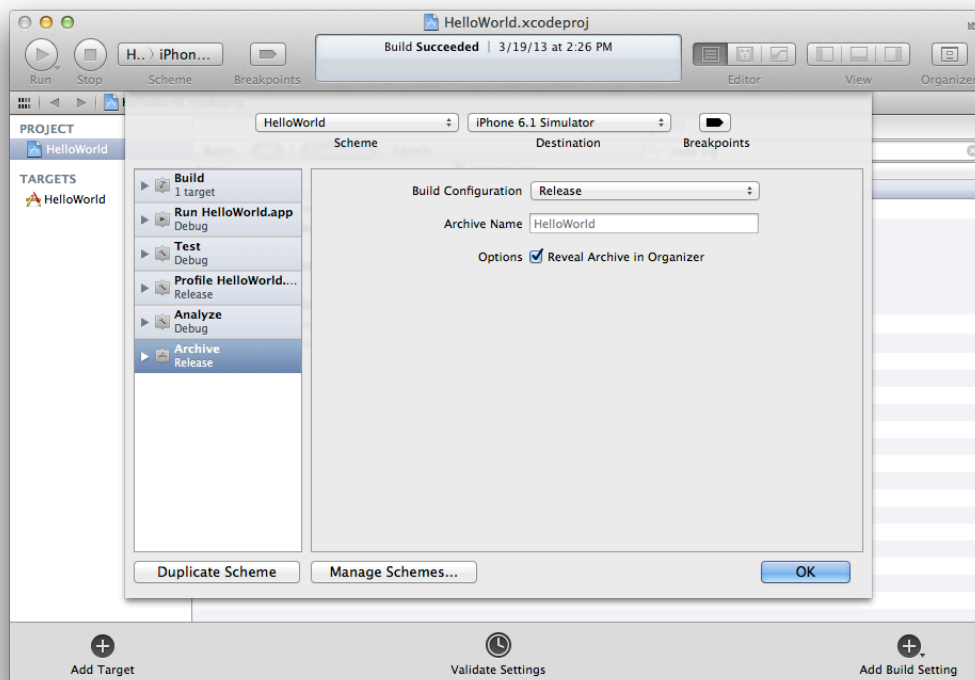
## Review the Archive Scheme Settings

Before you make an archive, review the Archive scheme settings to ensure that you don't archive a debug version of your app.

### To review the Archive scheme

1. In the Xcode project editor, choose Product > Scheme > Edit Scheme to open the scheme editor.

2. Click Archive in the column on the left.



3. Choose Release from the Build Configuration pop-up menu, and click OK.

## Creating and Validating an Archive

No matter what method you choose to distribute your app, you want to create an archive first. Before creating the archive, build and run your app one more time to ensure that it is the version you want to distribute. Immediately after creating the archive, validate it and fix any validation errors before continuing.

### To create an archive

1. In the Xcode project editor, select the project.
2. From the Scheme toolbar menu, choose a scheme.

---

**iOS Note:** Choose iOS Device or the device name from the scheme toolbar menu. You cannot create an archive of a simulator build. If an iOS device is connected to your Mac, the device name appears in the scheme toolbar menu. When you disconnect the iOS device, the menu item changes to iOS Device.

---

3. Choose Product > Archive.

The Archives organizer appears and displays the new archive.

**Important:** You cannot validate your app unless the app record in iTunes Connect is in the “Waiting for Upload” or later state, as described in [“Creating an App Record”](#) (page 153).

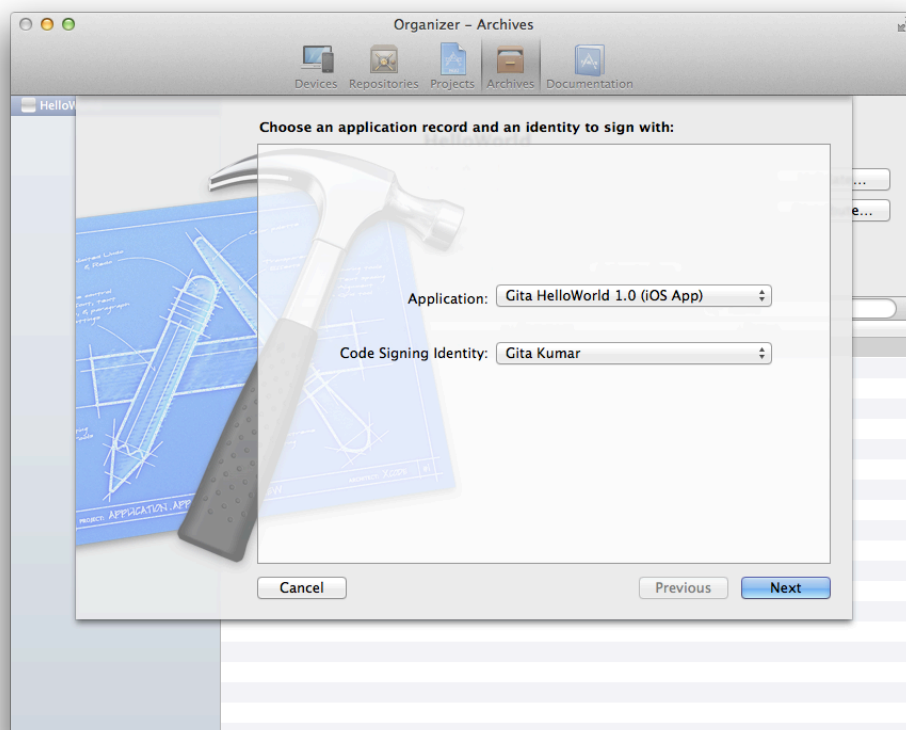
### To validate an archive

1. In the Archives organizer, select the archive.
2. Click the Validate button.
3. For Mac apps, select Mac App Store as the validation method and click Next.
4. Enter your iTunes Connect credentials and click Next.

If a dialog appears stating that no application record can be found, you need to create an app record in iTunes Connect before continuing.

5. Select the app you want to distribute and the appropriate signing identity, and click Next.

If you used a provisioning profile for development, select the code signing identity that appears under the store provisioning profile you created in a previous step.



iTunes Connect runs validation tests.

6. Review validation issues found, if any, and click Finish.

Fix any validation issues you find, create a new archive, and repeat these steps until there are no further issues. You cannot proceed until the archive passes all the validation tests.

## Troubleshooting

After you enter your iTunes Connect credentials, if a dialog appears stating that no application record can be found, create an app record in iTunes Connect before validating your app. To learn how to create an app record, read [“Creating an App Record”](#) (page 153).

If your distribution certificate doesn't appear in the Code Signing Identity menu when you validate the archive, refresh the provisioning profiles, as described in [“The Code Signing Identity Build Setting Doesn't Match Any Certificates”](#) (page 211).

## Test the Mac Installer Package

Before you submit to the Mac App Store, test the installation process to verify that your app installs correctly. Do this by saving the installer package to your disk and running a test using the `installer` command before submitting it.

You save an installer package to your disk by following the same steps for distributing your Mac app. When doing so, select Export as the distribution method, Mac Installer Package as the file format, and the Mac Installer Distribution certificate as the signing certificate. The name of the Mac Installer Distribution certificate is your team name, and it appears under “Identities without profiles” in the Code Signing Identity menu.

Do not test the installation process by opening the package with the Installer app. Only the `installer` command verifies that your app will be installed correctly when it is purchased from the Mac App Store.

To test your installer package, execute the following command in a Terminal window:

```
sudo installer -store -pkg path-to-package -target /
```

If the installer finds a bundle with the same bundle ID as the one it is installing, it upgrades the existing app. Users can then install upgrades even if they have moved your app. If you have a copy of your app installed (for example, in your build products directory), you may want to remove it so that the installer installs your app in `/Applications`. Other options include archiving the existing version in a ZIP file or moving it to another volume and unmounting that volume.

## Submitting Your App Using Xcode

Submitting your app to the App Store or Mac App Store is just one of several types of distribution methods you can choose. To submit your app using Xcode, select an archive and click the Distribute button in the Archives organizer. In the first dialog that appears, you select the store distribution method. The following dialogs and choices are slightly different for iOS and Mac apps.

If you prefer to upload your binary using Application Loader, read *Using Application Loader* for how to use Application Loader for this step.

**Important:** You cannot submit your app unless the app record in iTunes Connect is in the “Waiting for Upload” state or later, as described in [“Creating an App Record”](#) (page 153).

## Submitting Your iOS App

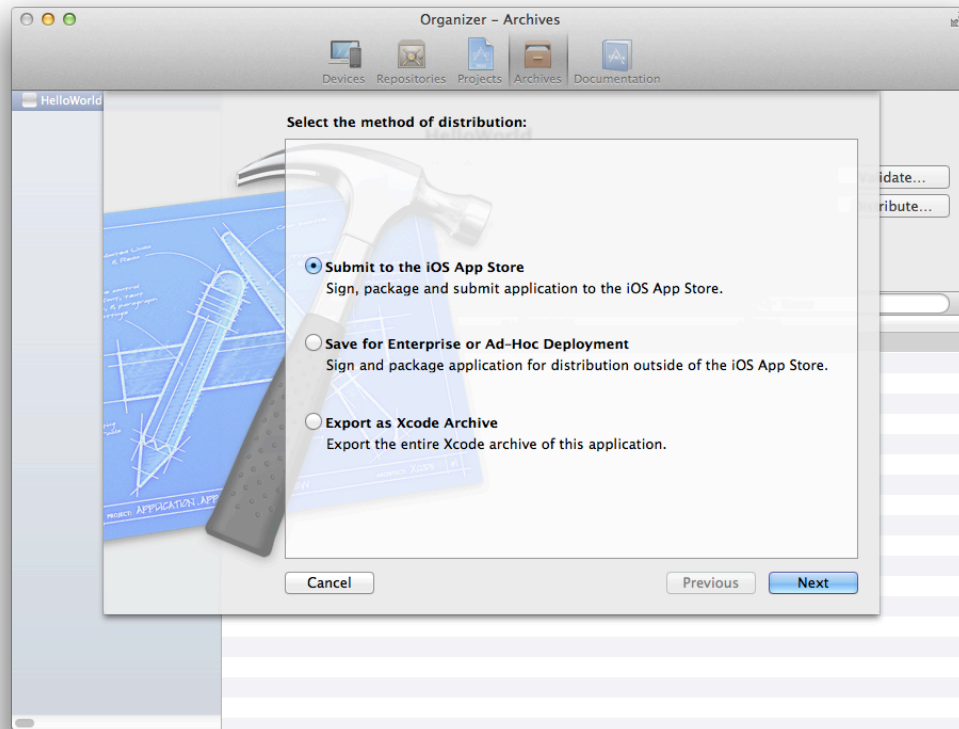
For iOS apps, select the “Submit to the iOS App Store” option when distributing your app.

### To submit an archive to the App Store

1. In the Archives organizer, select the archive.
2. Click the Distribute button.

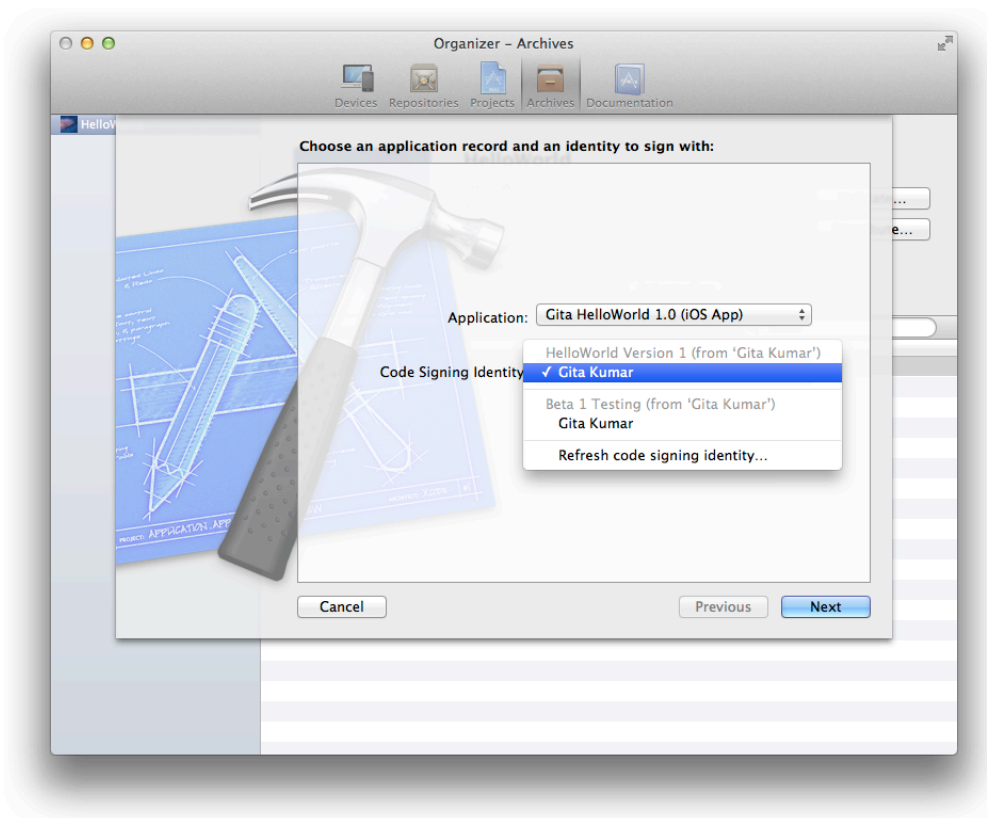


3. Select "Submit to the iOS App Store" and click Next.



4. Enter your iTunes Connect credentials and click Next.
5. Select the app you want to submit from the Application pop-up menu.

6. Select the distribution certificate in the store provisioning profile from the Code Signing Identity pop-up menu, and click Next.



iTunes Connect runs validation tests.

7. If issues are found, click Cancel and fix them before continuing.
8. If no issues are found, click Finish to submit your app.

Xcode transmits the archive to Apple, where the binary is examined to determine whether it conforms to the app guidelines. If the binary is rejected, correct the problems that were brought up during app approval and resubmit it.

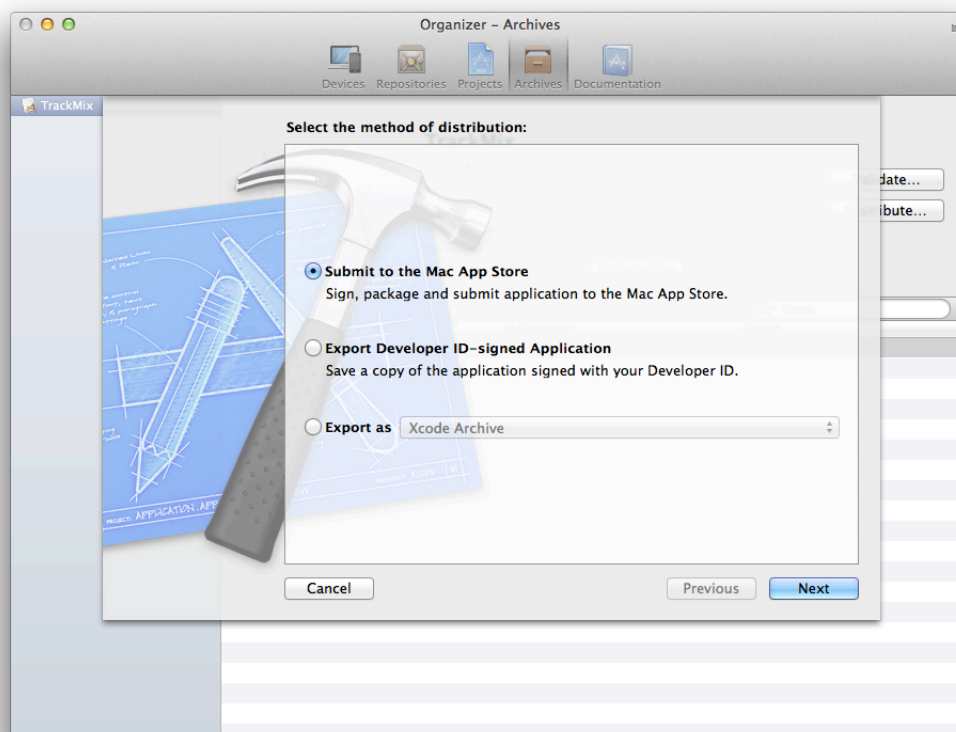
If you successfully submit your app, view the status of your app in iTunes Connect, as described in [“Viewing the Status of Your App”](#) (page 153). After submitting your binary, Apple may discover other problems with your app’s metadata in iTunes Connect that needs to be fixed before Apple can review your app. If no problems are found, the status of the app should change to “Waiting For Review.” To resolve any problems with your app record, refer to *iTunes Connect Developer Guide*.

## Submitting Your Mac App

For Mac apps, select the “Submit to the Mac App Store” option, and then later select an installer distribution certificate.

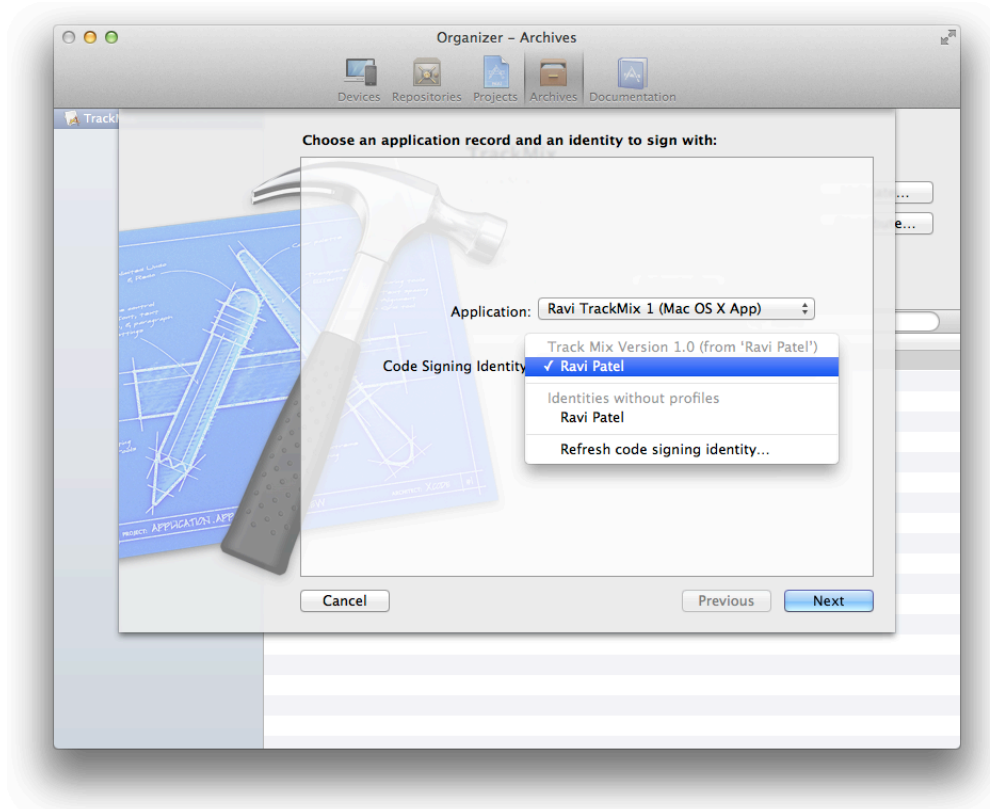
### To submit an archive to the Mac App Store

1. In the Archives organizer, select the archive.
2. Click the Distribute button.
3. Select “Submit to the Mac App Store” and click Next.



4. Enter your iTunes Connect credentials and click Next.
5. Select the app you want to submit from the Application pop-up menu.
6. Select the appropriate distribution certificate from the Code Signing Identity pop-up menu, and click Next.

Select the distribution certificate that contains your team name. If you are an individual developer, your team name is the same as your name. If you use a store provisioning profile, select the distribution certificate in your store provisioning profile.



iTunes Connect runs validation tests.

7. If issues are found, click Cancel and fix them before continuing.
8. If no issues are found, click Finish to submit your app.

Xcode transmits the archive to Apple, where it is examined to determine whether it conforms to the app guidelines. If the app is rejected, correct the problems that were brought up during app approval and resubmit it.

If you successfully submit your app, view the status of your app in iTunes Connect, as described in [“Viewing the Status of Your App”](#) (page 153). After submitting your binary, Apple may discover other problems with your app’s metadata in iTunes Connect that needs to be fixed before Apple can review your app. If no problems are found, the status of the app should change to “Waiting For Review.” To resolve any problems with your app record, refer to *iTunes Connect Developer Guide*.

## Troubleshooting

If your app doesn't appear after you enter your iTunes Connect credentials, read [“Before You Begin”](#) (page 134) to complete the necessary iTunes Connect configuration steps.

If your distribution certificate doesn't appear in the Code Signing Identity menu when you submit your app, refresh the provisioning profiles, as described in [“The Code Signing Identity Build Setting Doesn't Match Any Certificates”](#) (page 211).

## Recap

In this chapter you learned how to submit your app to the store using Xcode. This chapter doesn't cover the approval process. To learn how to view the status of your app, read *iTunes Connect Developer Guide*.

# Releasing and Updating Your App

After your app is approved, you can set the date when it is available to customers. After doing so, be prepared to maintain your app on the store. You need to respond to crash reports and customer reviews. You also need to fix problems promptly. You'll use iTunes Connect to manage and update versions of your app. This chapter provides an overview of a few common tasks you'll perform.

You'll use iTunes Connect to manage and update versions of your app.

**Check the status of your app.** Use iTunes Connect to check the status of your app after you submit your app and are waiting for approval, as described in [“Viewing the Status of Your App”](#) (page 153).

**View customer reviews.** Customer ratings and reviews on the store can have a big effect on the success of your app; if users run into problems, determine the bug and submit a new version of the app through the approval process. To view customer reviews, read [“Viewing Customer Reviews”](#) (page 156).

**View crash reports.** Use iTunes Connect to download crash reports submitted to Apple by users. Crash reports represent significant problems that users are finding in the app. To access and analyze these crash reports, read [“Viewing Crash Reports”](#) (page 155) and [“Analyzing Crash Reports”](#) (page 132).

**Update your app.** You follow the same distribution process to submit updates to your app. In iTunes Connect, you use the same app record but create a new version of your app. To update your app, read [“Creating New Versions of Your App”](#) (page 157).

iTunes Connect provides data to help you determine how successful the app is, including sales and financial reports, customer reviews, and crash logs submitted to Apple by users. Crash logs are particularly important, because they represent significant problems that users are seeing in the app. Make investigating these reports a high priority.

## Recap

This chapter summarized a few common tasks you perform to ship and manage your app after you submit it to the store. Before you release your app, you also need to enter sales and marketing information in iTunes Connect. To learn all the tasks you perform to manage your app on the store, read *iTunes Connect Developer Guide*.

# Managing Your App in iTunes Connect

iTunes Connect is a marketing and business web tool that iOS and Mac developers use to sign contracts, set up tax and banking information, submit versions of their app, and obtain sales and finance reports. During development, you enter metadata about your app, technologies that it uses, and any versions in to iTunes Connect.

Initially, only the team agent, who joined the developer program, has access to iTunes Connect. Otherwise, your developer program team and iTunes Connect users are completely separate. Because iTunes Connect is primarily used to manage the business aspects of your app and people performing those types of tasks are typically not developers, the team agent tightly controls access to iTunes Connect. Individual developers are the team agent for their one man team but can add non-developer iTunes Connect users too. Similar to team members, iTunes Connect users have roles and privileges.

iTunes Connect users with the admin and technical roles perform a number of tasks, explained in this chapter, in support of the development team and before the team can submit their app to the store:

1. Add iTunes Connect users to give other team members access to iTunes Connect.
2. Create your app record so you can configure store technologies and submit your app.
3. View the status of your app when you are ready to submit it or waiting for approval.
4. Change the availability date of an app to release it.
5. View crash reports and customer reviews after your app is available.
6. Create a new version of your app.

## About iTunes Connect User Roles and Privileges

The team agent manages access privileges to iTunes Connect. For example, changing the price of an app is a task you likely want to limit to a small number of people in your organization. Access to the iTunes Connect tool is configured separately and is designed to be more fine-grained than the access you set for team members. In iTunes Connect, each user can be assigned one or more roles; each role has different privileges. Table 11-1 describes the roles at a high level.

**Table 11-1** iTunes Connect roles and responsibilities

Role	Responsibilities
Legal	The legal role is automatically assigned to the team agent, and only the team agent is permitted to have this access. The legal role allows the team agent to sign legal contracts and other agreements.
Admin	The admin role grants access to all tasks in iTunes Connect except for those assigned to the legal role. A team agent is always assigned the admin role, and this access cannot be revoked without changing which person on the team acts as the team agent. An admin can assign iTunes Connect roles to other people on the team.
Technical	The technical role grants the ability to edit the app information stored in iTunes Connect and to view test accounts for store technologies.
Finance	The finance role grants access to financial reports and sales information. The finance role also authorizes the person to view contract, tax, and banking information.
Sales	The sales role grants access only to sales data.

Table 11-2 lists the most common modules (areas of iTunes Connect) you need to access, along with the roles that have access to each module. The legal role is not shown, because only the team agent has those rights. All participants can edit their own personal details stored in their accounts in iTunes Connect.

**Table 11-2** Abbreviated list of iTunes Connect modules, including availability by role

Responsibility	Legal	Admin	Technical	Finance	Sales
Manage Users	✓	✓	✗	✗	✗
Manage Test Users	✓	✓	✓	✗	✗
Manage Your Applications	✓	✓	✓	✗	✗
Sales and Trends	✓	✓	✗	✓	✓
Tax and Banking	✓	✓	✗	✓	✗
Contracts	✓	✗	✗	✗	✗
Payments and Financial Reports	✓	✓	✗	✓	✗
Catalog Reports	✓	✓	✓	✓	✓



## Adding iTunes Connect Users

To add a an iTunes Connect user, read “Setting Up an iTunes Connect User” in *iTunes Connect Developer Guide*.

## Creating an App Record

Certain store technologies require you to create an app record and enter the bundle ID using iTunes Connect during development. Later, you also need to create an app record in iTunes Connect to submit your app to the store. When you are ready to create your app record, read “Adding New Apps” in *iTunes Connect Developer Guide*.

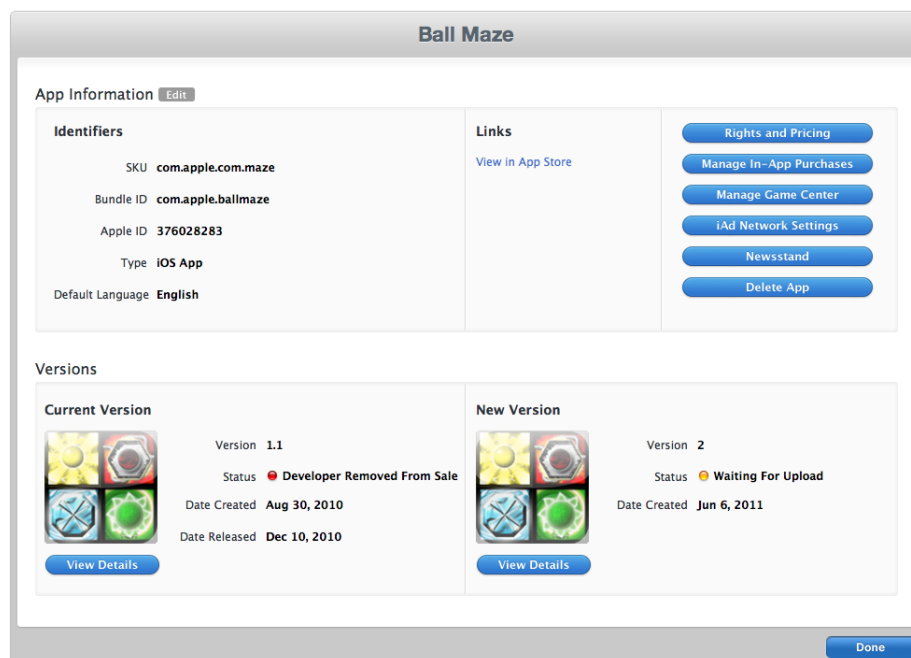
## Viewing the Status of Your App

To submit your app to the store, the status of the app record needs to be “Waiting for Upload” or later. You can view the status of your app in iTunes Connect.

### **To view the status of your app**

1. Sign in to [iTunes Connect](#).
2. On the iTunes Connect homepage, click Manage Your Applications.
3. Locate the app you want to edit, and click the large icon or app name.

The status of each version of your app appears below in the Versions section below the version number.



Refer to “Checking the Status of an App” in *iTunes Connect Developer Guide* for details on each status. Follow the steps in “Ready to Upload Your Binary” in “Adding New Apps” to change your app record from the “Prepare for Upload” to “Waiting for Upload” state.

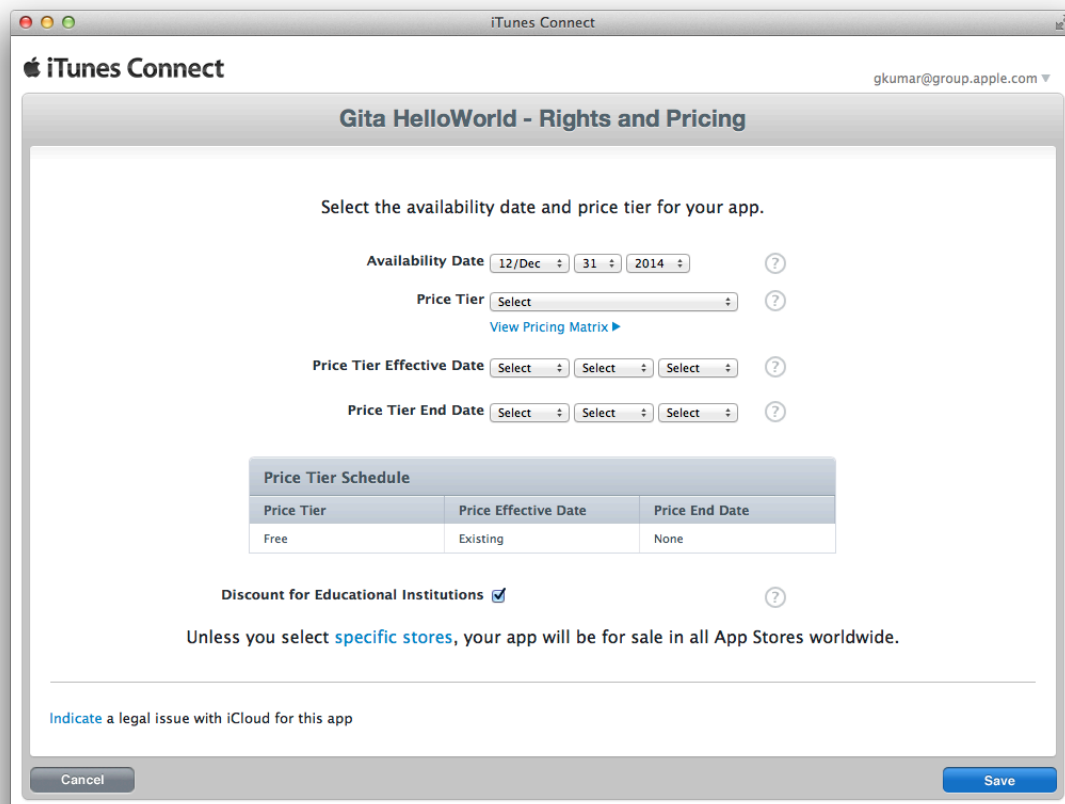
## Changing the Availability Date of Your App

Use iTunes Connect to set a date when the app is available on the store. For example, you can choose a date that immediately releases the app to the store after it is approved, or you can set a later date. Using a later availability date allows you to arrange other marketing activities around the launch of your app.

### To set the availability date

1. Sign in to [iTunes Connect](#).
2. Select Manage Your Applications.
3. Select your app in the Recent Activity section.
4. Click Rights and Pricing.

5. Choose a date from the Availability Date pop-up menus.



6. Optionally, edit the other fields on this form.
7. Click Save.

Changes you make to Rights and Pricing go live immediately (expect 24 hours for a full refresh of the changes on the store).

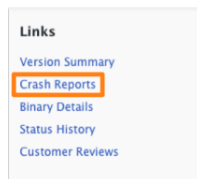
## Viewing Crash Reports

All crash logs contain stack traces for each thread at the time of termination. To view a crash log, open it from the Xcode Organizer window. As long as your Mac computer has the archive corresponding to the version of the app that generated the crash log, Xcode automatically resolves any addresses in the crash log with the actual classes and functions in the app.

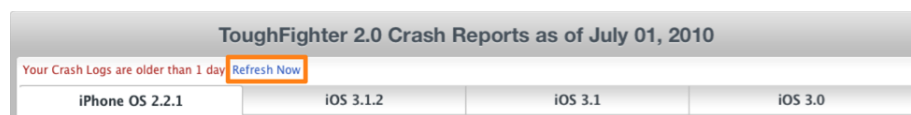
You view and save crash reports in iTunes Connect from the version details page.

## To view crash reports

1. Sign in to [iTunes Connect](#).
2. On the iTunes Connect homepage, click Manage Your Applications.
3. Locate the app you want to edit, and click the large icon or app name.
4. Click View Details for the version of your app.
5. Select Crash Reports in the upper-right corner.



6. Click Refresh Now to retrieve any new available crash reports.



7. Select the crash report you want to view, and save the crash report you want to retain.

To view the crash reports in Xcode, follow the steps in [“Analyzing Crash Reports”](#) (page 132).

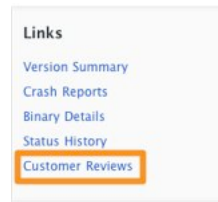
## Viewing Customer Reviews

You view customer reviews in iTunes Connect in the same way that you view crash reports, described in [“Viewing Crash Reports”](#) (page 155), except that you select Customer Reviews in the upper-right corner.

## To view crash reports

1. Sign in to [iTunes Connect](#).
2. On the iTunes Connect homepage, click Manage Your Applications.
3. Locate the app you want to edit, and click the large icon or app name.
4. Click View Details for the version of your app.
5. Select Customer Reviews under Links.

If you do not see this link, customer reviews are not available for this version of your app.



## Creating New Versions of Your App

To create new versions of your app, read “Updating Your App to a New Version” in *iTunes Connect Developer Guide*.

## Recap

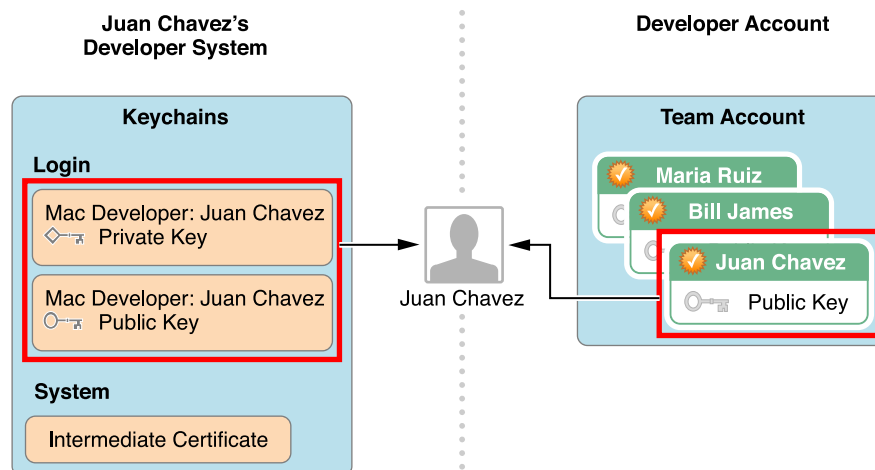
In this chapter you learned how to grant access to iTunes Connect and perform common development tasks. For complete documentation on using iTunes Connect, refer to *iTunes Connect Developer Guide*.

# Best Practices for Maintaining Certificates and Provisioning Profiles

During the lifetime of your developer program membership, you occasionally need to refresh your team certificates and provisioning profiles, especially when projects and team members change. You may also need to re-create expired certificates and repair invalid provisioning profiles. This chapter covers miscellaneous tasks you perform to maintain your certificates and provisioning profiles.

## About Protecting Your Code Signing Identities

A code signing identity—which consists of a public-private key pair that is issued by Apple—represents your credentials. The private key is stored locally on your Mac so you should protect it as you would an account password. If you move to another development Mac, you need to copy your code signing identity to the other Mac.



For a company, other team members have their own code signing identities installed on their Macs. Member Center contains a repository for all of the combined team assets but doesn't store any of the private keys. The private keys for other team members are stored locally on their respective Macs.

You should keep a secure backup of your private key. If the key is lost, you won't be able to sign code without creating an entirely new identity. Worse, if someone else has your private key, they may be able to impersonate you. In the wrong hands, someone might attempt to distribute an app that contains malicious code. Not only could that cause the app to be rejected, it could also mean your developer credentials could be revoked by Apple. Private keys are stored only in the keychain and cannot be retrieved if lost.

If you want to code sign your app using another Mac, you must export your developer profile on the Mac you used to create your certificates and import it on the other Mac.

## Exporting and Importing Certificates and Provisioning Profiles

After you create signing certificates and install them in your keychain—or later when you create provisioning profiles—you should export them to create a backup of your assets. You do this to move them to another Mac that you use for development, or repair a certificate if the private key is missing. Because the private key is stored in your login keychain, you can't refresh your provisioning profiles and certificates to replace a missing private key. Instead, import your certificates and provisioning profiles from a backup.

The **developer profile** contains the following team assets:

- Development certificates
- Distribution certificates
- Provisioning profiles

Because the developer profile represents your credentials to sign and submit apps to the store, Xcode encrypts and password protects the exported file.

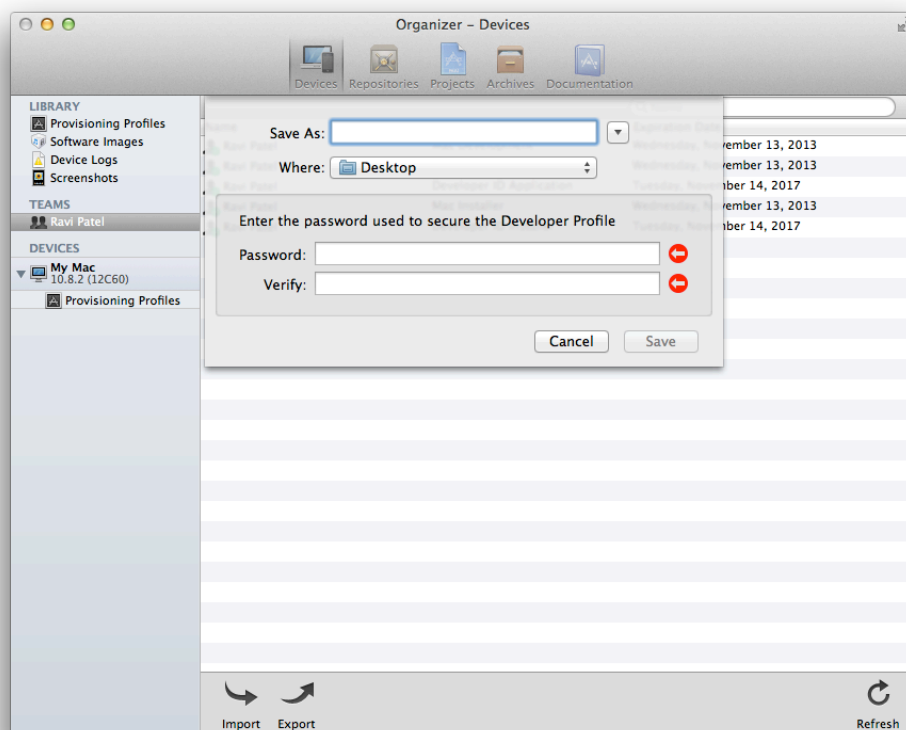
### Exporting Your Developer Profile

You should always maintain a backup copy of your developer profile. Do this when you first create certificates in Xcode and any time you make changes to your provisioning profile.

#### To export your developer profile

1. In the Devices organizer, select your team in the Teams section.
2. Click Export at the bottom of the window.
3. Enter a filename in the Save As field.
4. Enter a password in the Password and Verify fields.

The file is encrypted and password protected.



5. Click Save.

The file is saved to the location you specified with a `.developerprofile` extension.

## Importing Your Developer Profile

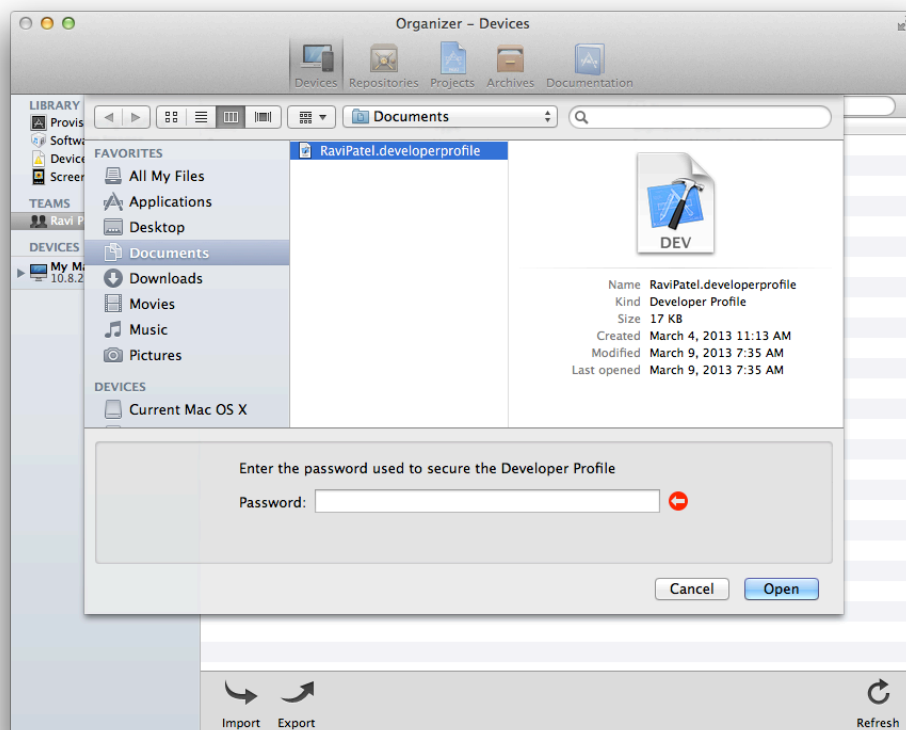
You import your developer profile to restore missing private keys or when you need to switch to another Mac.

### To import your developer profile

1. In the Devices organizer, select your team in the Teams section.
2. Click Import at the bottom of the window.
3. Locate and select the file containing your developer profile.



The file should have a `.developerprofile` extension.



4. Enter the password you used to encrypt the file.
5. Click Open.

If you use provisioning profiles, refresh the provisioning profiles after importing the signing certificates by opening the Devices organizer and selecting “Refresh from Developer Portal” from the Editor menu. If you imported your developer profile to repair a missing private key, verify that your certificates and provisioning profiles are now valid.

## Removing Certificates from Your Keychain

You remove certificates from your keychain if they are invalid, no longer used (perhaps they belong to a previous team you were a member of), or are missing the private key and consequently, are not usable. If you remove certificates from your keychain that are contained in provisioning profiles, those provisioning profiles also become invalid and need to be removed from Xcode and your devices.

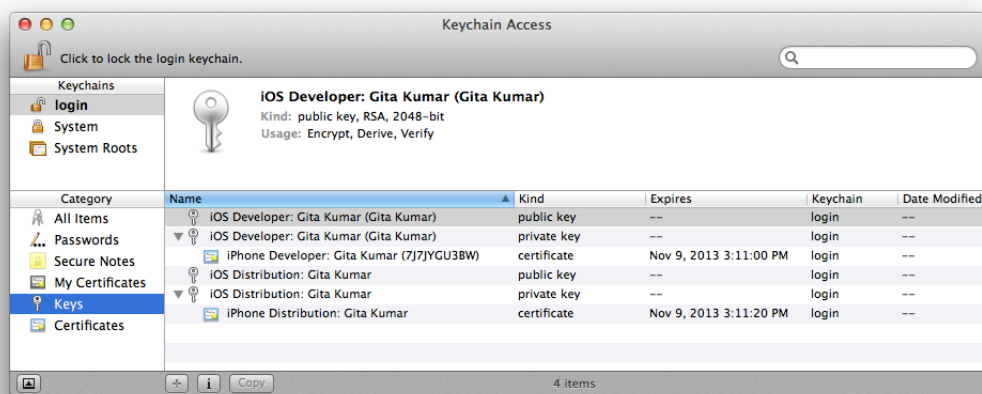
If you want to remove certificates because of missing private keys, you should not perform this operation without following all of the steps in “[Re-Creating Certificates and Updating Related Provisioning Profiles](#)” (page 181) to re-create your certificates.

If you have a backup of your developer profile (containing your code signing identities), don’t revoke your certificates after removing them from your keychain. Instead, import your certificates, as described in “[Importing Your Developer Profile](#)” (page 160).

**Warning:** You cannot re-create a private key once you remove it from your keychain unless you import it from a developer profile file. If you are intentionally re-creating your certificates, you must revoke all of your certificates immediately after removing them from your keychain. If you don’t, Xcode downloads and installs them in your keychain the next time you refresh your provisioning profiles. Without the private keys, you cannot sign apps using the certificates.

### To remove signing certificates from your keychain

1. Launch Keychain Access (located in /Applications/Utilities).
2. In the Category section, select Keys.
3. Click the disclosure triangles for all the private keys to reveal the associated certificates.



4. Select all of the private keys associated with the certificates that you want to remove.  
Refer to [Table 2-1](#) (page 36) for how to recognize the type of certificate by the name as it appears in Keychain Access.
5. Select the corresponding public key for each private key.
6. Press Delete (on the keyboard), and when a dialog appears, click Delete.

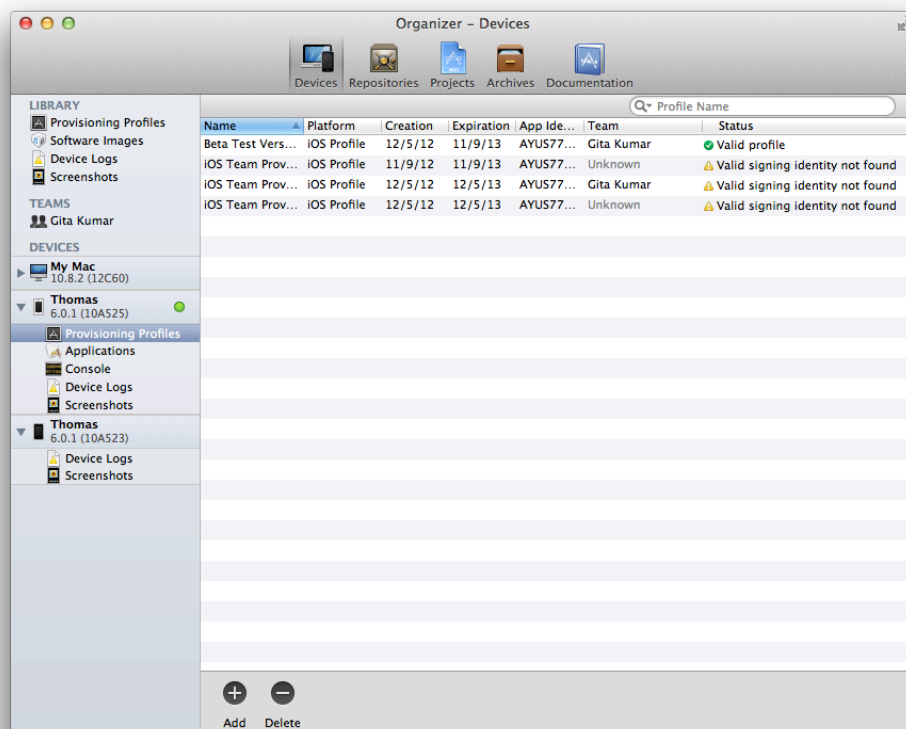
7. In the Category section, select Certificates.
8. Select all of the certificates that you want to remove.  
The certificates won't have private keys.
9. Press Delete (on the keyboard), and when a dialog appears, click Delete.

### To remove invalid provisioning profiles from Xcode

1. In Xcode, select Provisioning Profiles in the Library section of the Devices organizer.
2. Select the invalid provisioning profiles that appear in the list.  
The status of an invalid provisioning profile is "Valid signing identity not found."
3. Press Delete (on the keyboard), and when a dialog appears, click Delete.

### To remove invalid provisioning profiles from a device

1. Connect the device to your Mac.
2. In Xcode, click the disclosure triangle next to your device in the Devices organizer.
3. Select Provisioning Profiles under your device.



4. Select the invalid provisioning profiles.
5. Press Delete (on the keyboard), and when a dialog appears, click Delete.

## Revoking Certificates

You revoke certificates when you no longer need them or when you want to re-create them because of another code signing issue (refer to “[Certificate Issues](#)” (page 208) for the types of problems that can occur). You also revoke certificates if you suspect that they have been compromised. If you are a team admin for a company, you may want to revoke development certificates of team members who no longer work on your project.

Table 12-1 lists the types of certificates each team member can revoke. Individual developers are the team agent for their one-person team, which means they have permission to revoke all types of development and distribution certificates except as indicated in Table 12-1. For a company, any team member can revoke their own development certificate, but they can only revoke distribution certificates if they are a team agent or admin.

Table 12-1 Team certificate revoking privileges

Type of Certificate	Team Agent	Team Admin	Team Member
Your development certificates: iOS Development Mac Development	✓	✓	✓
Other team admin and member certificates: iOS Development Mac Development	✓	✓	✗
The team agent’s certificate: iOS Development Mac Development	✓	✗	✗
Store distribution certificates: iOS Distribution Mac App Distribution Mac Installer Distribution	✓	✓	✗

Type of Certificate	Team Agent	Team Admin	Team Member
Developer ID certificates: Developer ID Application Developer ID Installer	X	X	X
Push notification certificates: APNs Development iOS APNs Production iOS APNs Development Mac APNs Production Mac	✓	✓	X
Pass certificate: Pass Type ID	X	X	X

You cannot revoke Developer ID or Passbook certificates using Member Center. Instead, send a request to Apple at [product-security@apple.com](mailto:product-security@apple.com) to revoke these types of certificates.

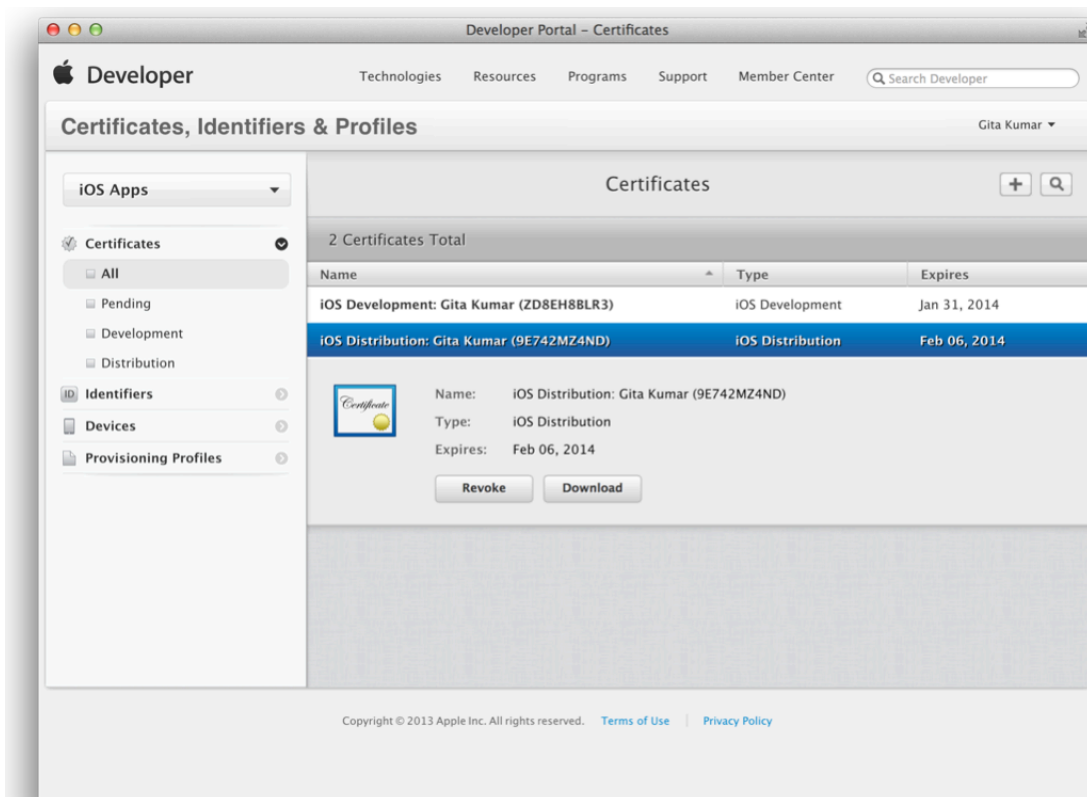
If Apple revokes your Developer ID certificate, users will no longer be able to install applications that have been signed with that certificate. Instead of revoking a Developer ID certificate, you can create additional Developer ID certificates using Member Center as described in [“Requesting Additional Developer ID Certificates”](#) (page 168).

If you want to revoke certificates because of missing private keys, do not perform this operation without performing all of the steps in [“Re-Creating Certificates and Updating Related Provisioning Profiles”](#) (page 181).

### To revoke a certificate

1. In [Certificates, Identifiers & Profiles](#), select Certificates.
2. Under Certificates, select All.

3. Select the certificate you want to revoke and then click Revoke.



4. Click Revoke when a dialog appears.

## Replacing Expired Certificates

When your development or distribution certificate expires, remove it and request a new certificate in Xcode. Follow the same steps to re-create certificates, as described in [“Re-Creating Certificates and Updating Related Provisioning Profiles”](#) (page 181).

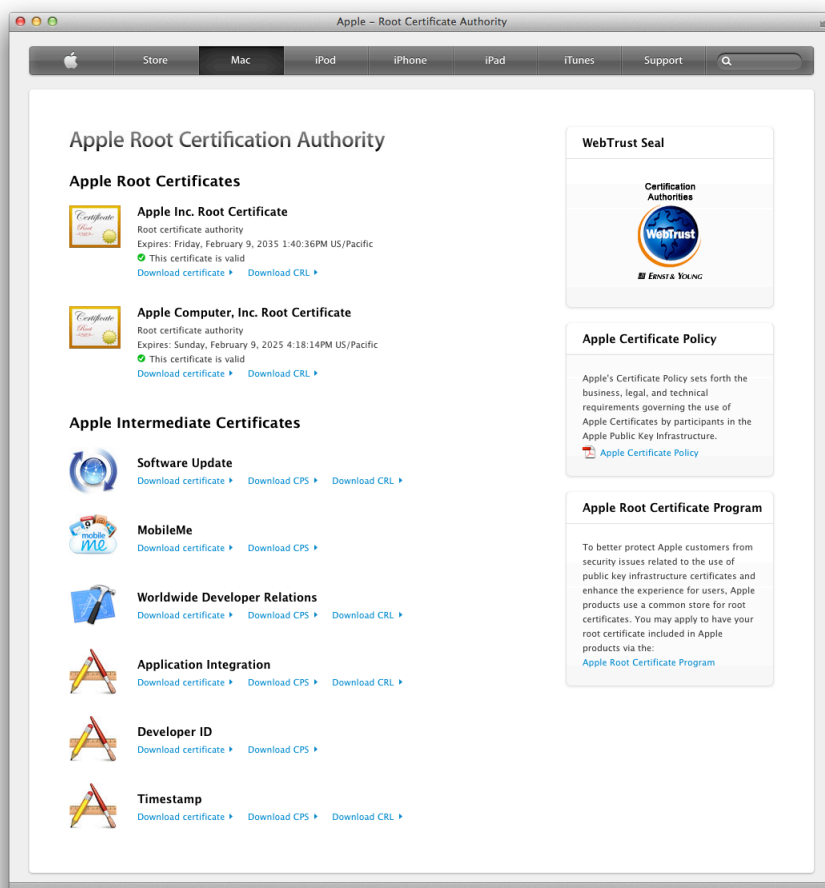
## Installing Missing Intermediate Certificate Authorities

To use your certificates, you need to have the correct intermediate certificate in your keychain. An intermediate certificate ensures that your certificates were issued by a trusted source. The intermediate certificate, named Apple Worldwide Developer Relations Certification Authority, is installed in your system keychain when you install Xcode. The intermediate certificate for Developer ID certificates is called Developer ID Certification Authority. If you accidentally remove an intermediate certificate, you can install it again.

### To install a missing intermediate certificate

1. Go to <http://www.apple.com/certificateauthority>.
2. Click "Download certificate" under Apple Intermediate Certificates for the intermediate certificate you are missing.

A certificate file, with a .cer extension, appears in your Downloads folder.



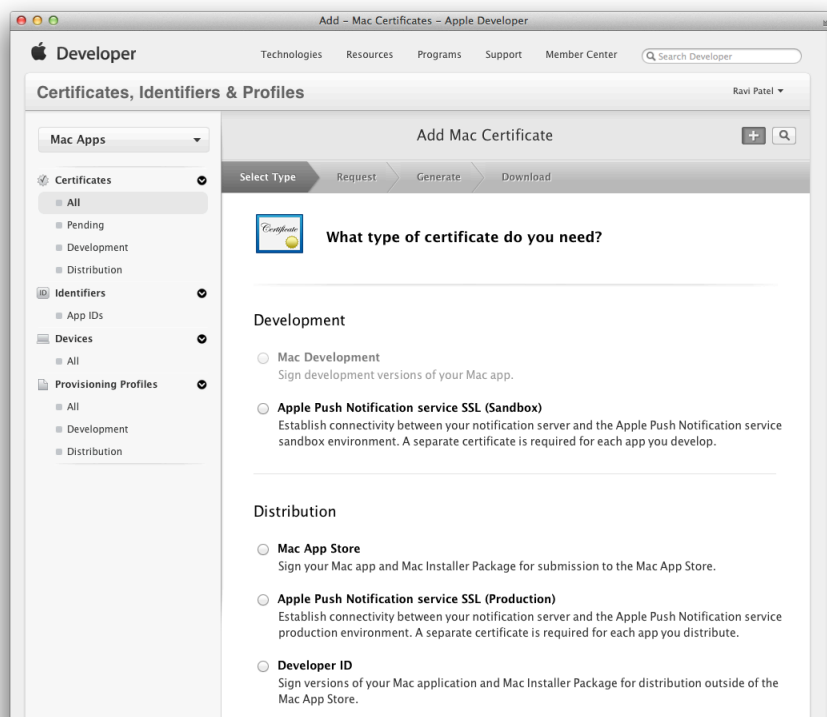
3. Double-click the certificate file to install it in your system keychain.

## Requesting Additional Developer ID Certificates

Developer ID certificates are used to distribute your application outside of the Mac App Store. Create your Developer ID certificates, along with other types of certificates, using Xcode as described in “[Requesting Signing Certificates](#)” (page 26). If you want more Developer ID certificates, you can create up to five of each type using Member Center.

### To create a Developer ID certificate

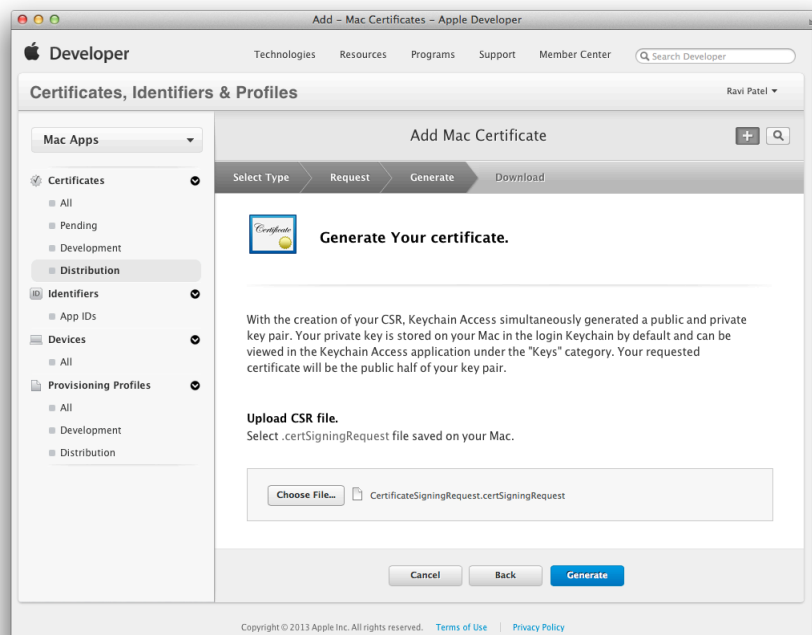
1. In [Certificates, Identifiers & Profiles](#), select Certificates.
2. Under Certificates, select All.
3. Click the plus button (+) in the upper-right corner.
4. Select Developer ID under Distribution and click Continue.



5. Select the certificate type—Developer ID Application or Developer ID Installer—and click Continue.
6. Follow the instructions to create a certificate signing request (CSR) using Keychain Access and click Continue.



7. Click Choose File.



8. Select a CSR file (with a `.certSigningRequest` extension) and click Choose.
9. Click Generate.
10. Click Download.

The certificate file appears in your Downloads folder.

To install the Developer ID certificate in your keychain, double-click the downloaded certificate file (with a `.cer` extension). The Developer ID certificate should appear in the My Certificates category in Keychain Access.

## Registering App IDs

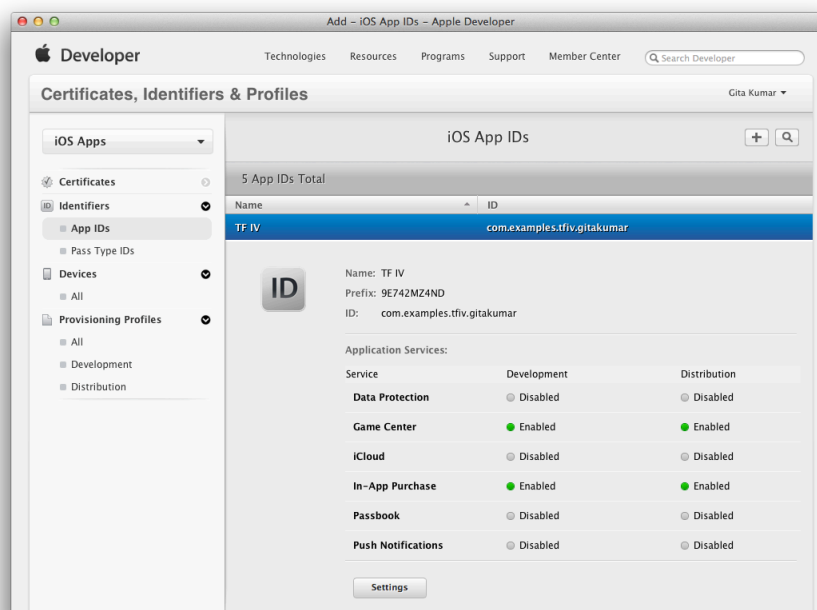
You register App IDs in the process of adding store technologies to your app. This is the first step to creating specialized development and distribution provisioning profiles, as described in [“Provisioning Your App for Store Technologies”](#) (page 54). Read [“Creating App IDs”](#) (page 57) if you only need to create an App ID.

## Deleting App IDs

You can also remove App IDs when you no longer need them.

## To remove an App ID

1. In [Certificates, Identifiers & Profiles](#), select Identifiers.
2. Under Identifiers, select App IDs.
3. Select the App ID you want to delete and then click Settings.



4. Scroll to the bottom of the page and click Delete.
5. Read the dialog that appears and click Delete.

Provisioning profiles that contain a deleted App ID become invalid. You can change the App ID in the provisioning profiles, as described in ["Editing Provisioning Profiles"](#) (page 175), or delete them.

## Registering Devices Using Member Center

In Member Center, you can register individual devices as needed or multiple devices by uploading a file that contains information about each device. Each year, you are allowed to register a fixed number of devices. The maximum number of devices you can register is 100. If you later delete a device, it won't decrease the current count of registered devices.

## Locating Device IDs

You need the name and device ID for each device you want to register. There are multiple methods to obtain device IDs depending on whether you have Xcode installed.

In Xcode, a team member can select a device in the Devices organizer to display the device ID.

### To locate your device ID using Xcode

1. Choose Window > Organizer.
2. Select your Mac in the Devices section.



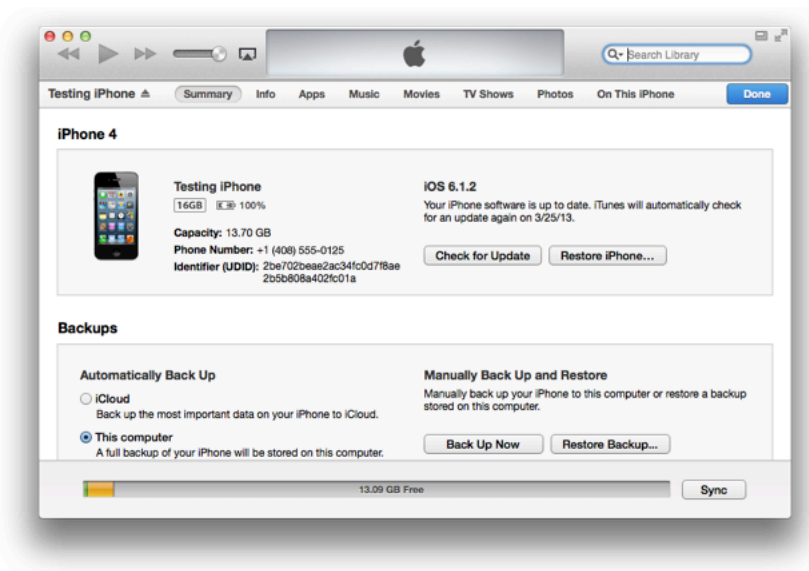
3. Select and copy the text in the Identifier field.

For iOS apps, you can also obtain a device ID using iTunes. For example, testers follow these steps to get their device ID using iTunes when they don't have Xcode installed.

### To get a device ID using iTunes

1. Launch iTunes on your Mac.
2. Connect your device to your Mac.

3. In the upper-right corner, select the device.
4. In the Summary pane, click the Serial Number label under Phone Number.  
The label Serial Number changes to Identifier and displays the device ID.



5. To copy the device ID, select Edit > Copy Identifier (UDID).  
If the Edit menu contains a Copy Serial Number menu item, click the Serial Number label under Phone Number to change the menu item to Copy Identifier (UDID).
6. Paste the device ID in a document or email message.

For Mac apps, you can get a device ID using the System Information app. For example, use this method if you want to register a Mac for testing that is not used for development.

### To locate your Mac device ID using System Information

1. Open the System Information app located in the /Applications/Utilities folder.
2. Select Hardware in the left column.

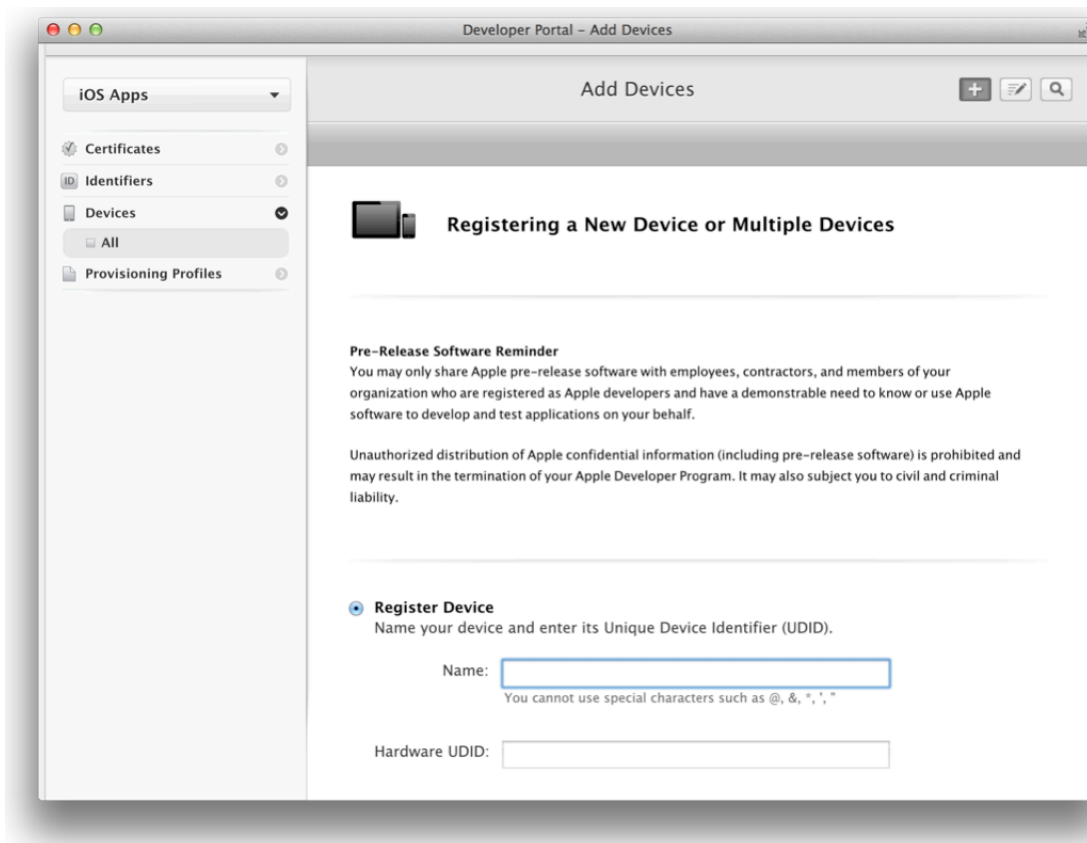
The device ID, or hardware UUID, appears near the bottom of the Hardware Overview pane and is of the form XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX.

## Registering Individual Devices

To register a device using Member Center, you need to have the device name and device ID.

## To register a single device

1. In [Certificates, Identifiers & Profiles](#), select Devices.
2. Under Devices, select All.
3. Click the plus button (+) in the upper-right corner.
4. Select Register Device.
5. Enter a device name and the device ID.



6. Scroll to the bottom of the page, and click Continue.
7. Review the registration information, then click Submit.

## Registering Multiple Devices

If you have many test devices, you can create a file containing the device names and device IDs, and upload the entire file to Member Center. Member Center supports these two file formats: a property list file with a `.deviceids` file extension, and a plain text file. The file format you choose depends on whether you have access to the devices you want to register.

## Creating a Property List Devices File

If you have access to the testing devices, you can use the iPhone Configuration Utility to create a property list file that contains the device names and device IDs of the devices you connect to your Mac.

### To download iPhone Configuration Utility

1. Go to <http://www.apple.com/support/iphone/enterprise/>.
2. Scroll down to the iPhone Configuration Utility section.
3. Click the appropriate download link and follow the online instructions.

### To create the devices file using iPhone Configuration Utility

1. Launch iPhone Configuration Utility.
2. Connect each device to your Mac in turn.  
iPhone Configuration Utility adds the device information to the Devices section under Library.
3. Select Devices under Library and then select the devices you want to add to the file.
4. Click Export in the toolbar.
5. Enter a filename in the Save As text field.
6. Choose Device UDIDs from the “Export type” pop-up menu.
7. Click Save.

## Creating a Plain Text Devices File

If you don't have access to the testing devices, you can create a .txt file containing the device names and device IDs you collect using another method. In this case, create a tab-delimited file with one device ID and one device name in each row. You can use the first row for your headers, because that row is ignored when parsed.

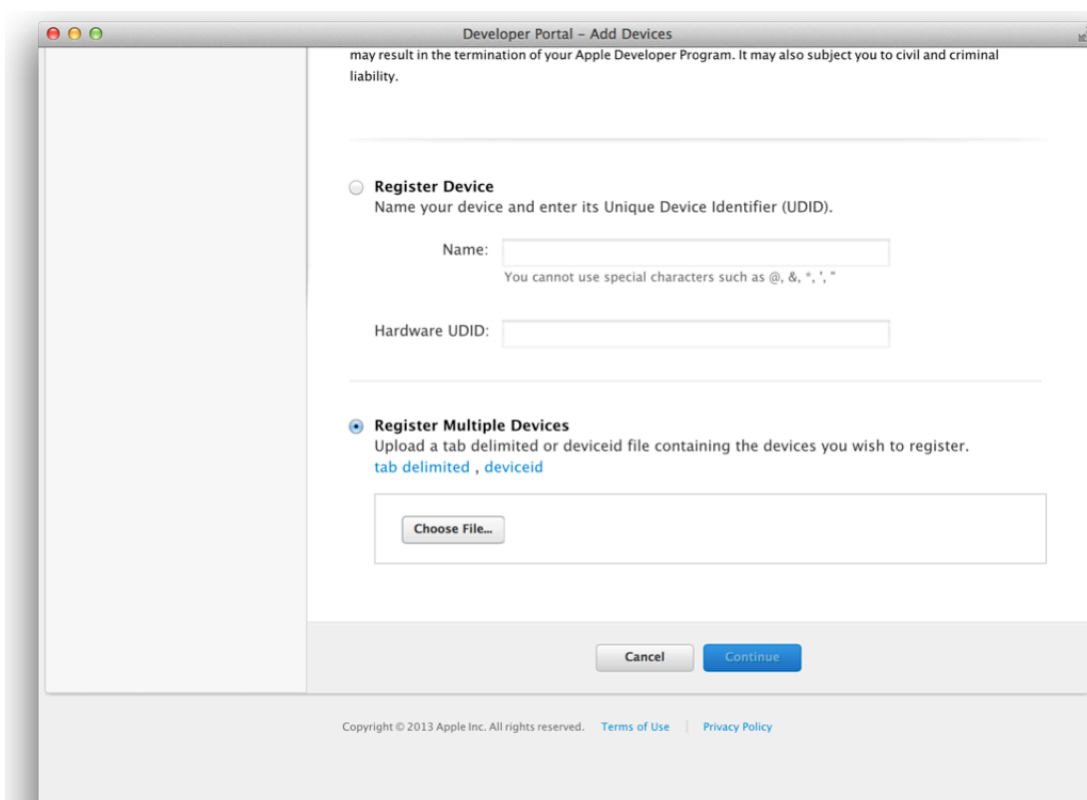
## Uploading the Devices File

In [Member Center](#), the steps to upload the devices file are the same for both formats.

### To register multiple devices

1. In [Certificates, Identifiers & Profiles](#), select Devices.
2. Under Devices, select All.
3. Click the plus button (+) in the upper-right corner.

4. Select Register Multiple Devices.
5. Click Choose File.



6. Select the file you want to upload and click Choose.  
Select either the `.deviceids` or `.txt` file you created earlier.
7. Click Continue.
8. Review the registration information, then click Submit.

## Editing Provisioning Profiles

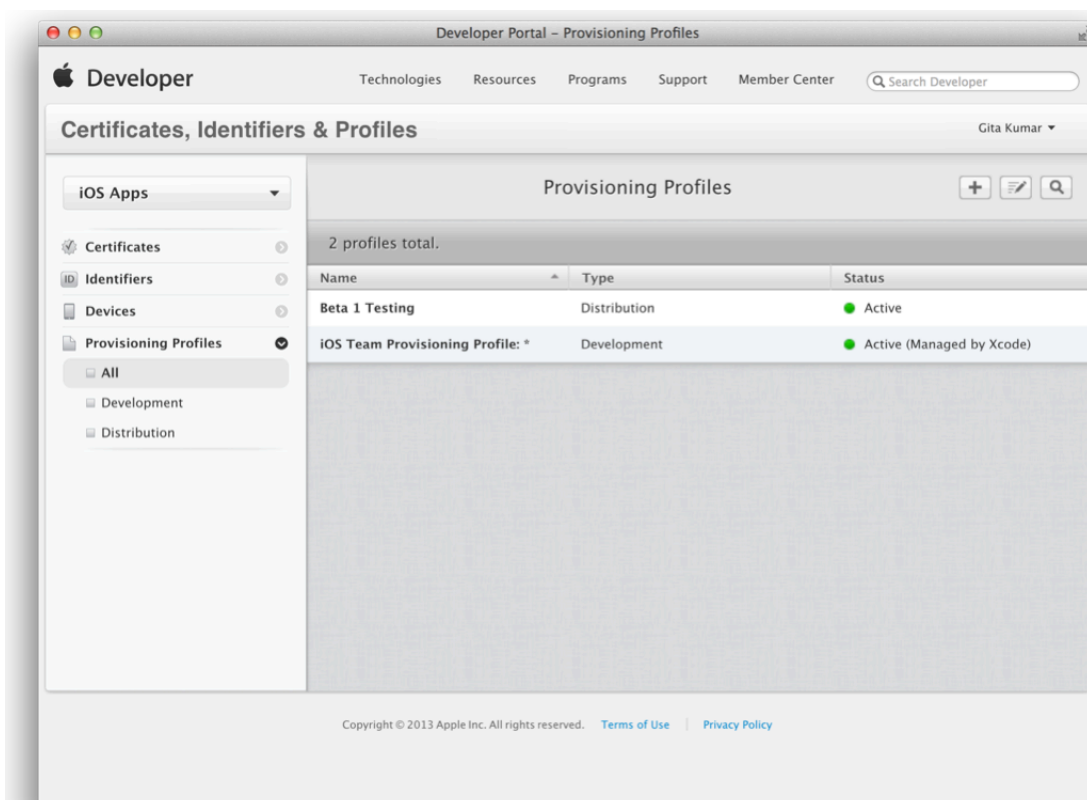
You don't need to re-create provisioning profiles to edit them. You can change the name of a provisioning profile and other properties depending on the type of provisioning profile. For all types of provisioning profiles, you can change the App ID. For an iOS app, you can add devices to an ad hoc provisioning profile. If you change a provisioning profile, remember to replace instances of the provisioning profile that you installed on devices.

If you want to repair a provisioning profile because you re-created a certificate, be sure to follow all the steps in “Re-Creating Certificates and Updating Related Provisioning Profiles” (page 181).

**Note:** You cannot modify the team provisioning profile created by Xcode.

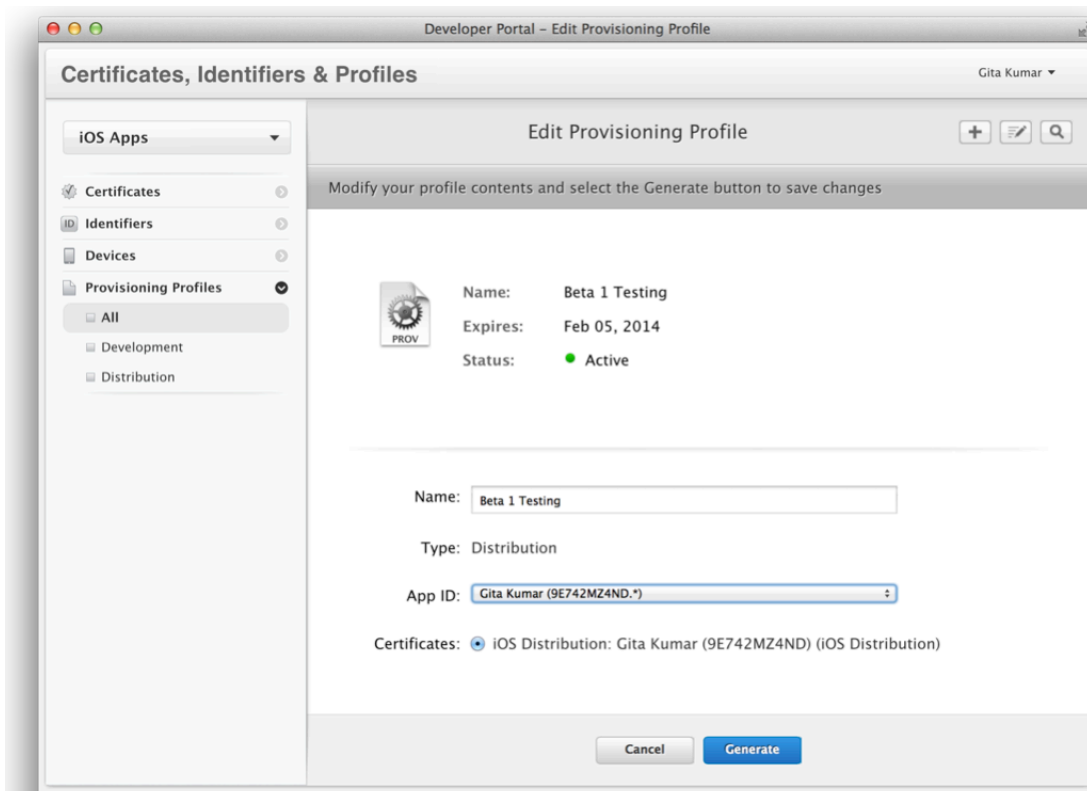
## To edit a provisioning profile

1. In [Certificates, Identifiers & Profiles](#), select Provisioning Profiles.
2. Under Provisioning Profiles, select All.
3. Select the provisioning profile you want to modify, and then click Edit.





4. Make your changes to the provisioning profile, such as changing its name, adding certificates, or selecting a different set of devices.



5. Click Generate.

After the profile is generated, you can download the profile or refresh provisioning profiles in Xcode.

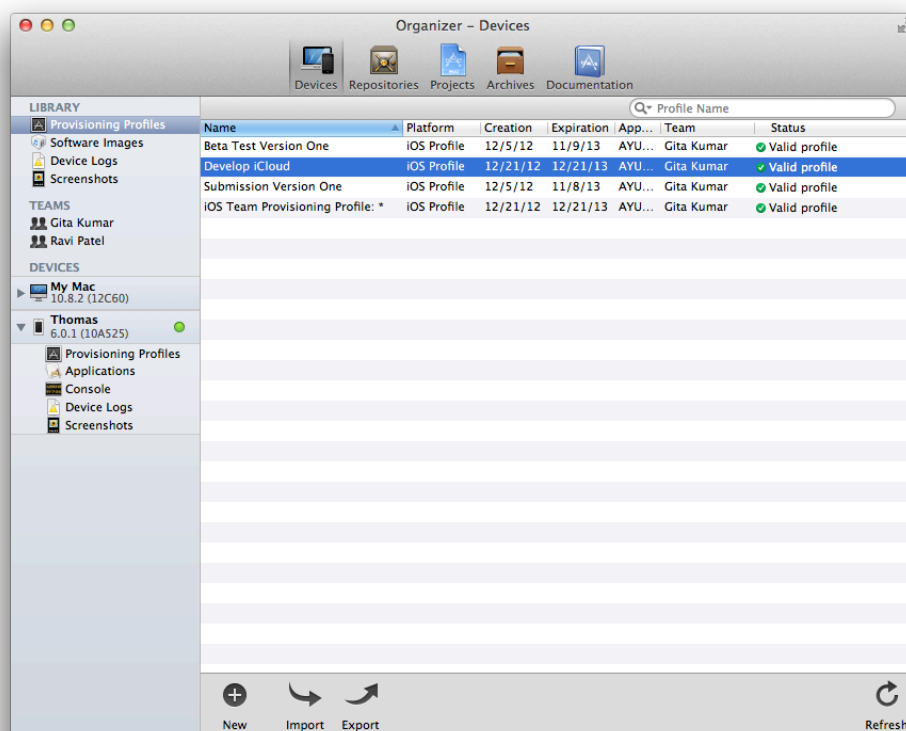
## Installing and Removing Provisioning Profiles from Devices

If you want to launch an app on a device that requires a development provisioning profile, install the provisioning profile on the device. You can later remove a provisioning profile that you no longer need or that is invalid. You can install and remove provisioning profiles from devices using Xcode.

**iOS Note:** iOS automatically installs an embedded ad hoc provisioning profile on a device.

## To install a provisioning profile on a device using Xcode

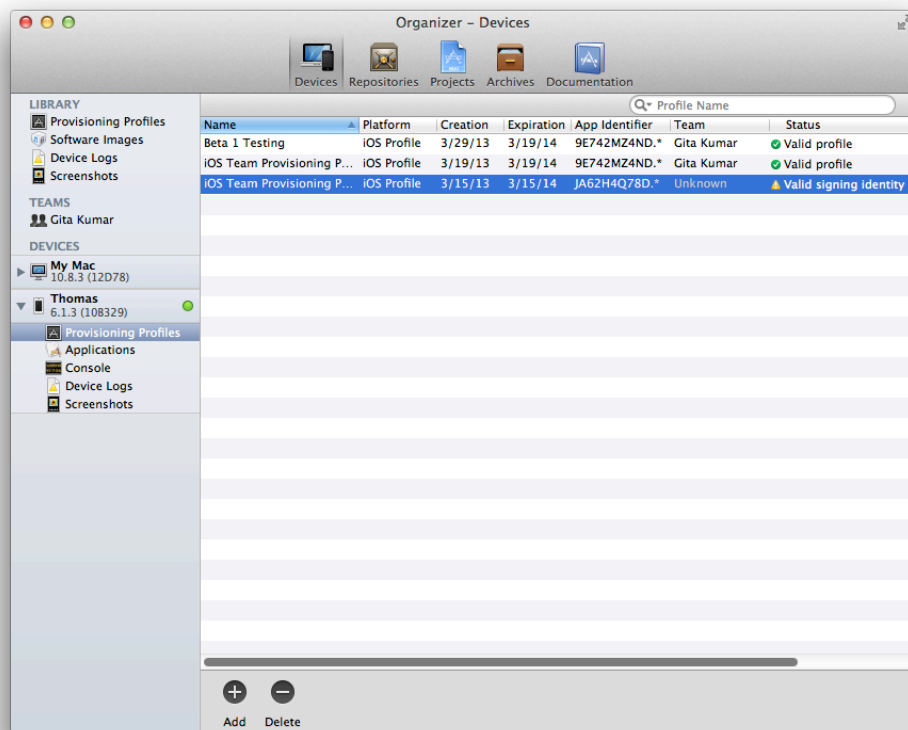
1. For iOS apps, connect the device to your Mac.
2. Select Provisioning Profiles in the Library section of the Devices organizer.
3. Drag the provisioning profile from the Library section to your device in the Devices section.



## To remove a provisioning profile from a device using Xcode

1. For iOS apps, connect the device to your Mac.
2. Select the disclosure triangle next to your device under Devices to reveal the contents.
3. Select Provisioning Profiles in the left column under your device.

4. Select the provisioning profile you want to delete on the right.



5. Click Delete.

## Removing Provisioning Profiles from Your Team

Occasionally, you may need to remove a provisioning profile from your team. After doing so, you should also remove the provisioning profile from your devices, as described in [“Installing and Removing Provisioning Profiles from Devices”](#) (page 177).

### To remove a provisioning profile from your team

1. In [Certificates, Identifiers & Profiles](#), select Provisioning Profiles.
2. Under Provisioning Profiles, select All.
3. Select the provisioning profile you want to remove.
4. Click Delete.
5. Read the dialog that appears and click Delete.

## Renewing Expired Provisioning Profiles

If a provisioning profile expires, the provisioning profile appears as such in both Xcode and Member Center. Using Xcode, a team agent or admin can renew an expiring or expired provisioning profile.

### To renew an expired provisioning profile

1. In the Devices organizer, select Provisioning Profiles in the Library section.
2. In the provisioning profiles list, select the provisioning profile you want to renew.
3. Click Renew.
4. Enter your Apple ID user name and password, and click “Log in!”

If you installed the provisioning profile on your device, replace the expired provisioning profile with the renewed provisioning profile.

### To replace a provisioning profile

1. In the Library section in the Devices organizer, select Provisioning Profiles.
2. From the provisioning profiles list, drag the new provisioning profile to your device.
3. Delete the old provisioning profile from your device by selecting it and clicking Delete.

If the provisioning profile is an ad hoc provisioning profile, then re-sign and distribute your app using the provisioning profile, as described in [“Beta Testing Your iOS App”](#) (page 117).

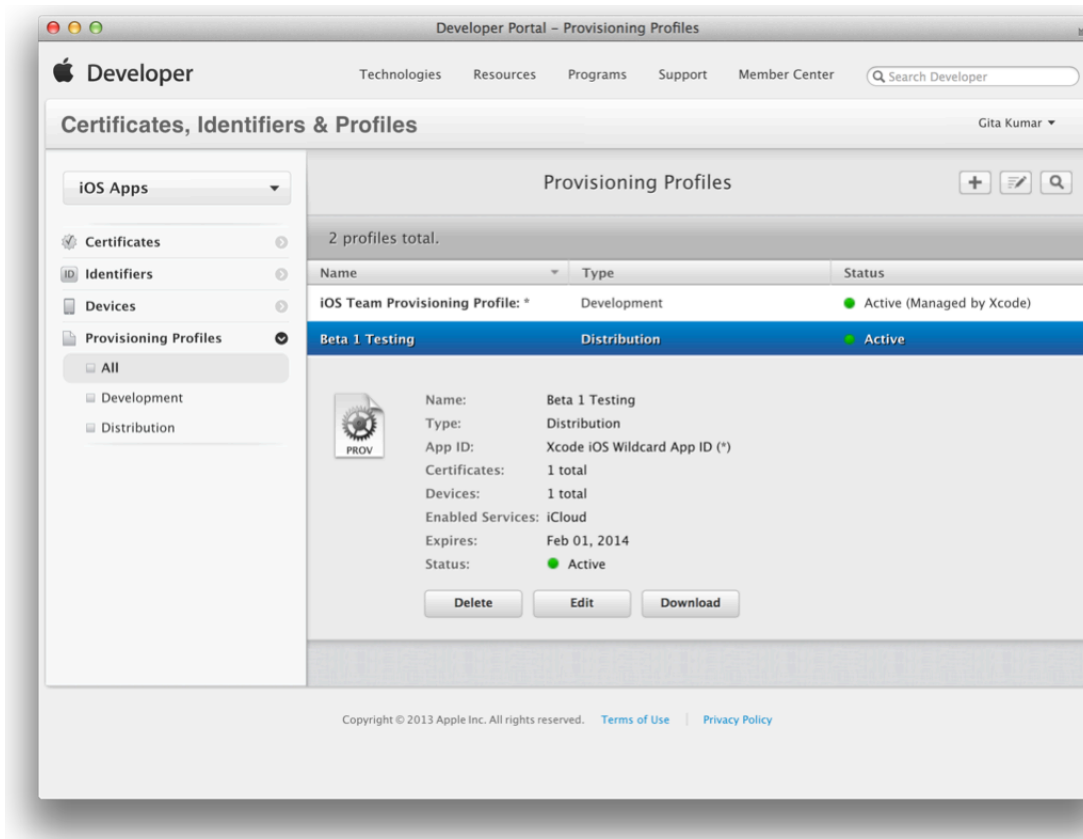
## Downloading Provisioning Profiles from Member Center

If necessary, you can download specific provisioning profiles directly from Member Center.

### To download a provisioning profile

1. In [Certificates, Identifiers & Profiles](#), select Provisioning Profiles.
2. Under Provisioning Profiles, select All.
3. Select the ad hoc provisioning profile.

4. Click Download.



A file with the provisioning profile name and the `.mobileprovision` extension appears in your Downloads folder.

## Re-Creating Certificates and Updating Related Provisioning Profiles

Re-creating certificates and updating related provisioning profiles is not a simple task because these assets are related and reside on both your Mac and in Member Center. For example, if you remove a certificate from Member Center, any provisioning profile that contains that certificate becomes invalid. Similarly, if you remove a certificate from your Mac, Xcode attempts to download it again from Member Center the next time you refresh your developer profile.

There are several reasons you might want to re-create your certificates and update related provisioning profiles. For example, you do this if:

- You accidentally removed one or more private keys from your keychain and don't have a backup to restore from.

- You moved to another Mac for development and refreshed your certificates without first importing your developer profile. In this case, the certificates are installed in your keychain but without the private keys.
- You want to remove revoked or expired certificates and their related provisioning profiles.
- You want to remove all the certificates and provisioning profiles from a previous team before you join another team.
- If you use multiple Macs for development or belong to multiple teams, you might want to set up your Mac for a new project.

However, if you are experiencing certificate, provisioning, or build issues, you should review [“Troubleshooting”](#) (page 208) first before performing these steps because removing your certificates is irreversible.

Choose the certificates you want to re-create. For example, if you are experiencing problems running your app on a device, you only need to re-create your development certificate. Keep in mind that re-creating a distribution certificate doesn't affect your development certificates or development provisioning profiles. Similarly, re-creating a development certificate doesn't affect your distribution certificate or distribution provisioning profiles.

**Important:** Re-creating your development or distribution certificates does not affect apps that you've submitted to the store nor does it affect your ability to update them.

Follow these steps to re-create your certificates and update related provisioning profiles:

1. Remove the certificates and related provisioning profiles locally, as described in [“Removing Certificates from Your Keychain”](#) (page 161).
2. Revoke the certificates using Member Center, as described in [“Revoking Certificates”](#) (page 164).
3. Request new certificates using Xcode, as described in [“To verify that your device is registered”](#) (page 43).
4. Modify custom provisioning profiles that contain the revoked certificates, as described in [“Editing Provisioning Profiles”](#) (page 175).

You don't need to modify team provisioning profiles after revoking development certificates, because Xcode maintains them for you.

5. In Xcode, select “Refresh from Developer Portal” from the Editor menu, to refresh your provisioning profiles.
6. If necessary, install the modified provisioning profiles on your devices, as described in [“Installing and Removing Provisioning Profiles from Devices”](#) (page 177).

If you are repairing multiple Macs, follow all these steps on the primary Mac. Then remove the certificates locally on the other Mac and import the certificates on that Mac. Once the certificate has been repaired on the primary Mac, export your developer profile. Then remove the certificates locally on all of the other Macs before importing the certificates to those Macs.

If you have a backup (a file with a `.developerprofile` extension) and did not revoke the certificates, import your certificates instead, as described in [“Importing Your Developer Profile”](#) (page 160).

## Recap

In this chapter you learned how to maintain your certificates and provisioning profiles in a valid state and remove assets that you no longer need. To resolve specific certificate and provisioning profile issues, read [“Troubleshooting”](#) (page 208).

# Managing Your Team

If you have a company membership in an Apple Developer Program, you can add people to your team and assign them roles, thereby granting them levels of access to team assets. Team members have roles and privileges that pertain to the development process. These roles define who is allowed to sign apps, who is allowed to create signing certificates, and so on. After adding team members, you may be responsible for performing other tasks on their behalf. For example, you may need to approve signing certificates and create provisioning profiles for team members. If you are an individual, you are the team agent for your one-person team and don't need to perform any of the tasks described in this chapter.

Team members are not the same as iTunes Connect users. Only the person who joins the Apple Developer Program initially has access to iTunes Connect. To learn how to add additional iTunes Connect users, read [“Managing Your App in iTunes Connect”](#) (page 151).

## About Apple Developer Program Team Roles and Privileges

A person's role on the team defines the level of access he or she has to the team's assets and types of tasks he or she can perform using developer tools. This privilege level extends to the kinds of tasks that a developer is allowed to perform on behalf of the team. For example, only certain members of the team are allowed to submit apps to the store. By giving you control over team roles, Apple makes it easier for you to maintain good security practices for the team.

If your team belongs to multiple developer programs, you can set different team roles for each program. You can also choose not to give someone access to a program.

### Team Roles

Table 13-1 lists the roles a person can play and describes each. Each level of access includes all the capabilities of the levels below it.

Table 13-1 Team roles

Role	Description
Team agent	A <b>team agent</b> is legally responsible for the team and acts as the primary contact with Apple. The team agent can invite team members and change the access level of any other team member. There is only one team agent.



Role	Description
Team admin	A <b>team admin</b> can set the privilege levels of other team members, except the team agent. Team admins manage all assets used to sign your apps, either during development or when your team is ready to distribute an app. Team admins are the only people on a team who can sign apps for distribution on nondevelopment devices. Team admins also approve signing certificate requests made by team members.
Team member	A <b>team member</b> can gain access to prerelease content delivered by Apple in Member Center. A team member can also sign apps during development, but only after he or she makes a request for a development signing certificate and has that request approved by a team admin.

## Team Privileges

Each team role defines a set of privileges or tasks that a person can perform. Table 13-2 drills deeper into this list of privileges granted to members of the team. The privileges are listed in chronological order to help guide you through the process. Refer to [Table 12-1](#) (page 164) for the types of certificates that each team member can revoke.

**Table 13-2** Privileges assigned to each membership level

Privilege	Team agent	Team admin	Team member
Have legal responsibility for the team	✓	✗	✗
Be the primary contact with Apple	✓	✗	✗
View prerelease Apple content	✓	✓	✓
Enroll in additional developer programs and renew them	✓	✗	✗
Invite team admins and team members	✓	✓	✗
Request development certificates	✓	✓	✓
Approve team member requests for development certificates	✓	✓	✗
Request distribution certificates	✓	✓	✗
For Mac apps, request Developer ID certificates	✓	✗	✗

Privilege	Team agent	Team admin	Team member
Add devices for development and testing	✓	✓	✗
Create App IDs and enable store technologies	✓	✓	✗
Create development and distribution provisioning profiles	✓	✓	✗
Create SSL certificates for Apple Push Notification service	✓	✓	✗
Download development provisioning profiles	✓	✓	✓
Submit apps to the App Store or Mac App Store	✓	✗	✗

## Team Agent

To start, one person must enroll in either the iOS or the Mac Developer Program; this person becomes the team agent for the team. The team agent may enroll in both programs if your team intends to develop apps for both operating systems. During this step, the team agent signs the legal agreements required to become an Apple developer and enters financial information so that the team can be paid for purchases of their app from the store.

The team agent is special; he or she has unrestricted access to the team and is legally responsible for the team. Initially, the team agent also performs most of the tasks to organize the team. After others have joined the team, the team agent may decide to delegate some of this authority to other members of the team, allowing those others to perform these tasks instead.

**Important:** Because the team agent has sole legal responsibility for the team, the team may not demote the team agent using Member Center or iTunes Connect, nor can the team agent's privileges be restricted. To change the person acting as the team agent, you must contact Apple directly.

The team agent might need to sign updated or new licensing agreements, particularly when the team wants to incorporate specific technologies into an app. For example, an app that uses the iAd service requires that the team agent sign a separate agreement.

## Before You Begin

You'll use Member Center to perform most of the team management tasks. Before starting, sign in to [Member Center](#), as described in “[Accessing Member Center](#)” (page 16).

## Inviting Team Members and Assigning Roles

If you enroll as a company, you are the de facto team agent who has permission to add other developers, called *team members*, to your account. In general, team members have read access to view and download information managed by Member Center—but they do not have write access. However, you can assign an admin role to a team member, which allows that person to have some of the privileges of a team agent—for example, a team admin can create signing certificates and provisioning profiles but can't sign agreements. Assigning roles helps team agents delegate some of their responsibilities.

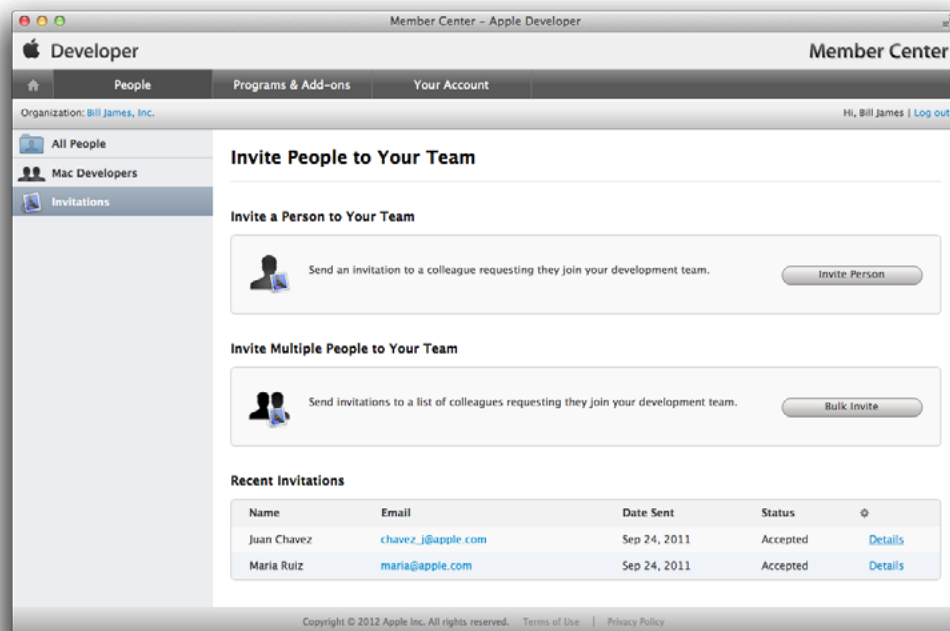
### Inviting Team Members

When you invite people to join your team, you enter information about them and set their role on the team.

#### **To invite team members**

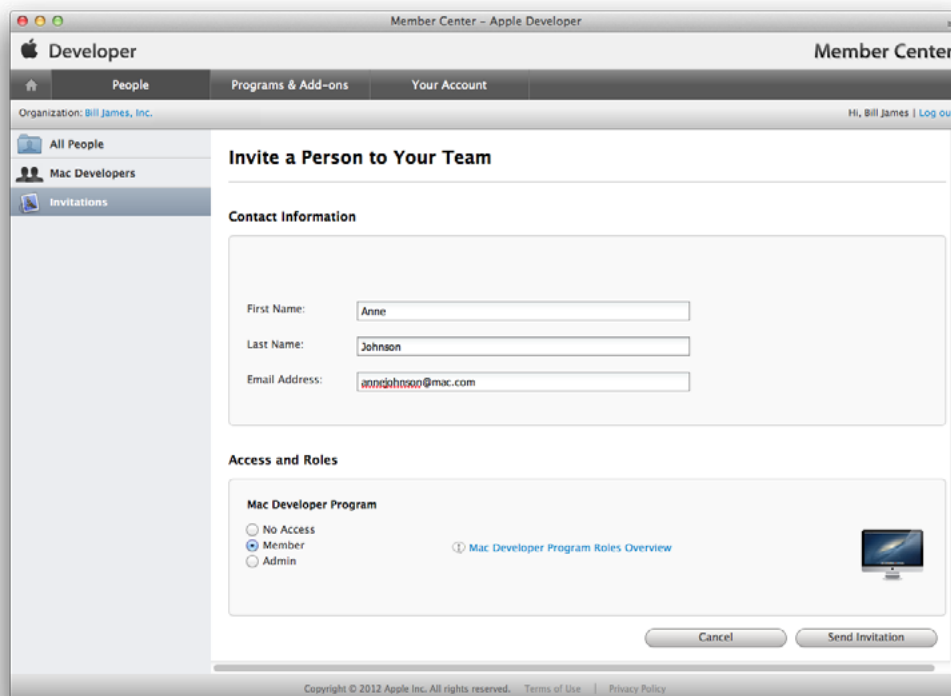
1. In [Member Center](#), click People in the tab bar at the top.
2. Click Invitations in the sidebar.

3. Click Invite Person.



4. Enter the first name, last name, and email address of the person you want to invite.

5. Specify the person's access and role for each program.



6. Click Send Invitation.

The person you specified receives an email invitation, which he or she must verify by clicking the invitation code in it. If the person does not have an Apple ID, he or she is asked to create one before accepting the invitation.

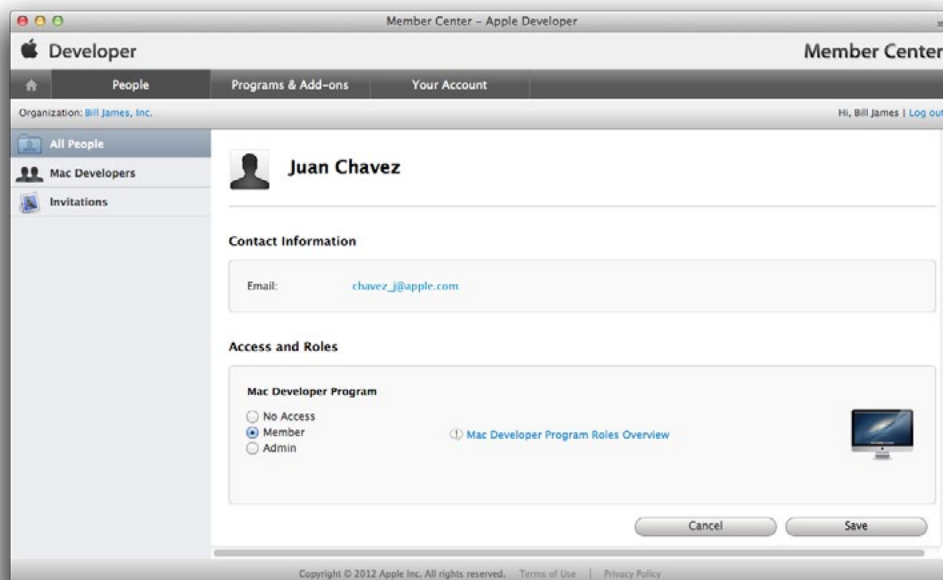
## Changing Team Roles

After the team member accepts the invitation, the team agent receives a confirmation email and the team member has access to Member Center. Later, the team agent can change the role of a team member.

### To change a team member's role

1. In [Member Center](#), click People in the tab bar at the top.
2. Click All People in the sidebar.
3. Click Details in the last column in the row of the person whose role you want to change.

- Specify the person's access and role for each program, and click Save.



**Important:** Team members should belong to only one iOS or Mac Developer Program; otherwise, Xcode displays information for multiple teams, which can be confusing.

## Approving Development Certificates

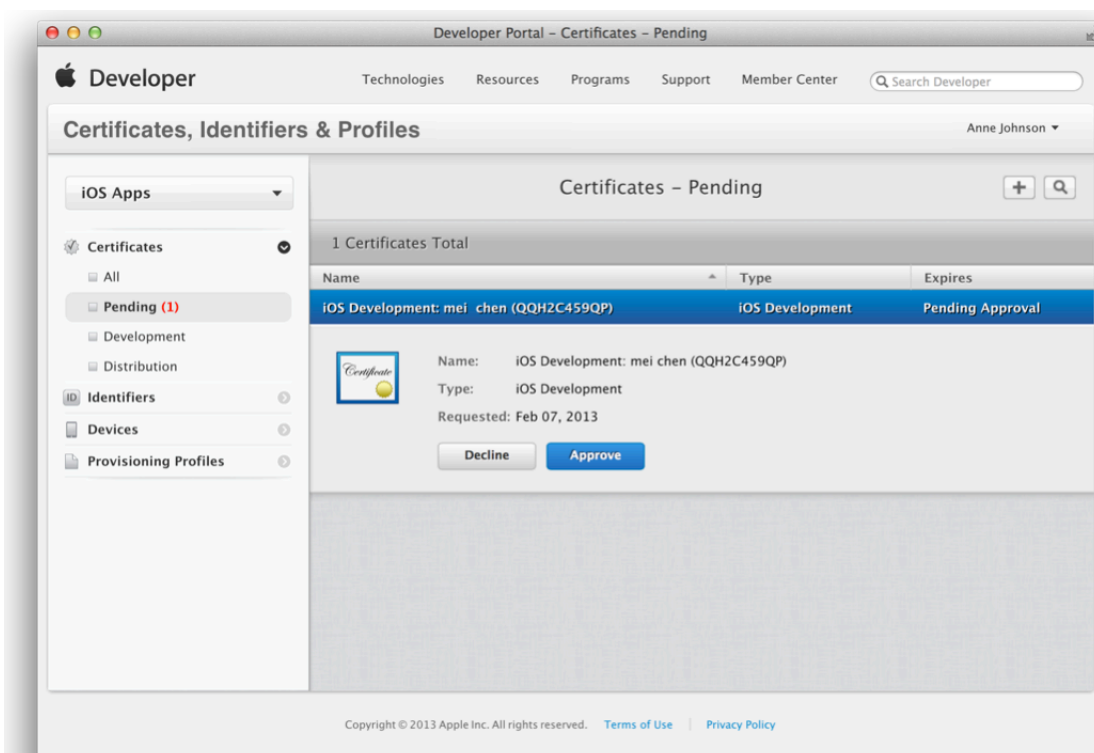
If you are a team admin for a company, it is your responsibility to approve team member requests for development certificates. Team members need a development certificate in order to sign apps, to use the team provisioning profile, or to be added to other provisioning profiles. Team admins are notified via email when a team member requests a development certificate. The email contains a link to Member Center to approve the request.

To learn how to request development certificates using Xcode, read [“Requesting Signing Certificates”](#) (page 26). Team admins also use Xcode to request their own signing certificates, which are automatically approved.

**Important:** All developers on a team should keep a secure backup of their private key. If the private key is lost, that team member can no longer sign code without creating a new code signing identity. After approving a development certificate, the team admin or agent should remind the team member to export his or her developer profile, as described in “Exporting Your Developer Profile” (page 159).

### To approve a development certificate request

1. In [Certificates, Identifiers & Profiles](#), select Certificates.
2. Under Certificates, select Pending.
3. Select the certificate.
4. Click either Decline or Approve.



If you use the team provisioning profile, you need to regenerate it after approving the certificate. Xcode regenerates the team provisioning profile whenever a team member refreshes provisioning profiles in Xcode. Afterward, all other team members need to refresh their provisioning profiles to download the latest team provisioning profile.

## Registering Team Member Devices

Before creating development provisioning profiles, team members need to register their devices. Team admins can register their own devices using Xcode, as described in [“Adding Devices to Your Team Provisioning Profile”](#) (page 40). A team member needs to send a request to a team admin to register his or her device. The team member provides the device name and device ID to their team admin.

In Xcode, a team member can select the device in the Devices organizer to display the device ID. If you’re a Mac developer, you can also get the device ID using the System Information app.

### To locate your device ID using Xcode

1. Choose Window > Organizer.
2. Select your Mac in the Devices section.



3. Select and copy the text in the Identifier field.

### To locate your Mac device ID using System Information

1. Open the System Information app located in the /Applications/Utilities folder.



2. Select Hardware in the left column.

The device ID, or hardware UUID, appears near the bottom of the Hardware Overview pane and is of the form `XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX`.

In Member Center, you can register one or more devices, as described in [“Registering Devices Using Member Center”](#) (page 170).

## Recap

In this chapter you learned how to perform some tasks on behalf of team members who don't have privileges to create development certificates or register their devices.

# Distributing Applications Outside the Mac App Store

In some cases, you may want to distribute an application outside the Mac App Store. As that app won't be distributed through the Mac App Store, use a Developer ID certificate to assure your users assurance that you are an Apple-identified developer.

Mac users have the option of turning on Gatekeeper, a security feature that gives users the ability to install software only from the Mac App Store and identified developers. If your application is not signed with a Developer ID certificate issued by Apple, it will not launch on Macs that have this Gatekeeper option selected. To avoid this situation, sign your applications and installer packages using a Developer ID certificate and thoroughly test the end-user experience using a Gatekeeper-enabled Mac before distributing your application outside of the Mac App Store.

This chapter describes the Xcode workflow to create and test Developer ID-signed applications for distribution and provides links to more information for developers who use the command line for signing their applications or installer packages.

## Creating Developer ID-Signed Applications or Installer Packages

Creating a Developer ID-signed application or installer package is a multistep process. For most developers, the entire Developer ID workflow takes place within Xcode. First you request Developer ID certificates. There are two types of Developer ID certificates: a Developer ID Application is used to sign applications, and a Developer ID Installer is used to sign installer packages. Using Xcode, you export and sign an archive of your application using the Developer ID Application certificate. You can also use command-line utilities to sign an installer package using the Developer ID Installer certificate.

**Important:** But before begin, enroll in the Mac Developer Program, as described in [“Enrolling in an Apple Developer Program”](#) (page 15). Only Mac Developer Program members are eligible to request Developer ID certificates and sign applications or installer packages using them.

## Requesting Developer ID Certificates

Use the Xcode Organizer window to obtain the Developer ID Application and Developer ID Installer certificates. You also need the Developer ID Certification Authority intermediate certificate, that Xcode installs in your keychain for you, to use these certificates.

When you refresh your certificates and provisioning profile assets for the first time, Xcode asks whether it should create signing certificates on your behalf. Signing certificates that begin with the text “Developer ID” are used to distribute your application outside the Mac App Store.

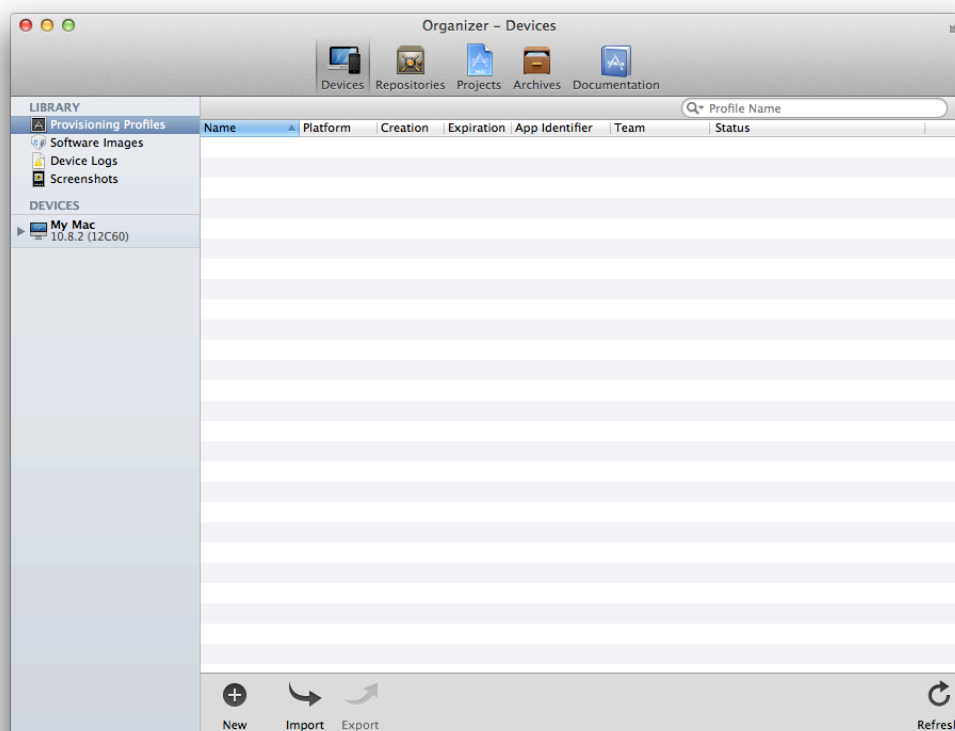
---

**Note:** Only a team agent can request Developer ID certificates. If you are an individual developer, you are the team agent and can request these certificates. You need to contact [product-security@apple.com](mailto:product-security@apple.com) to revoke Developer ID certificates.

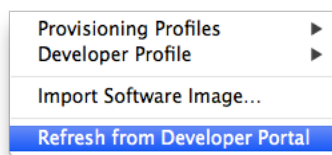
---

### To request your Developer ID certificates

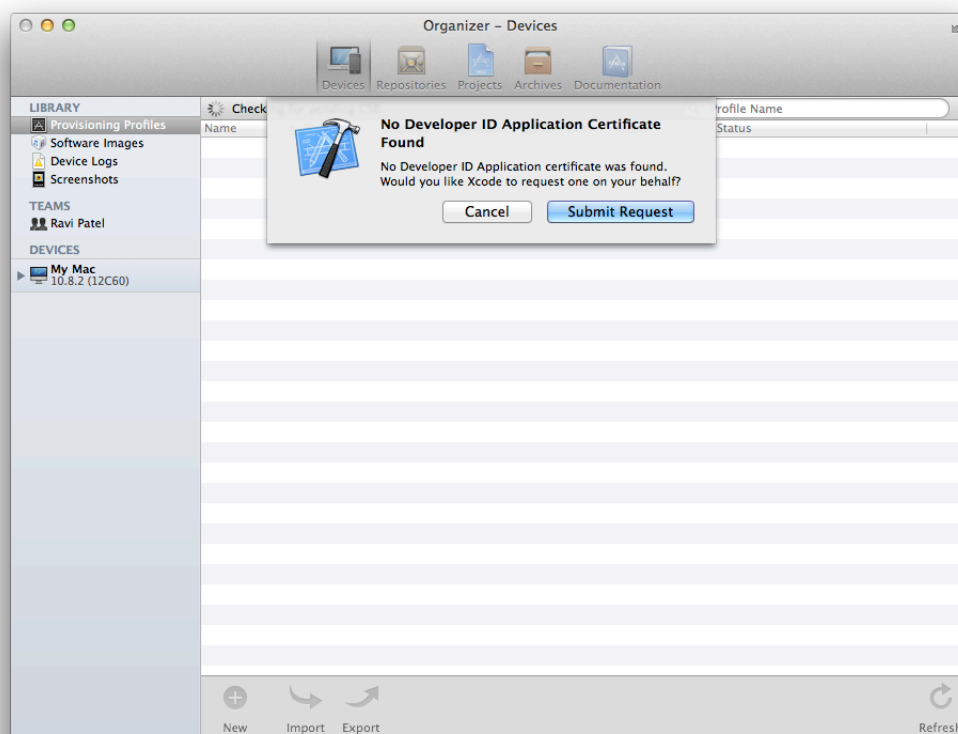
1. In Xcode, choose Window > Organizer to open the Organizer window.
2. Click Devices to display the Devices organizer.



3. Select “Refresh from Developer Portal” from the Editor menu.

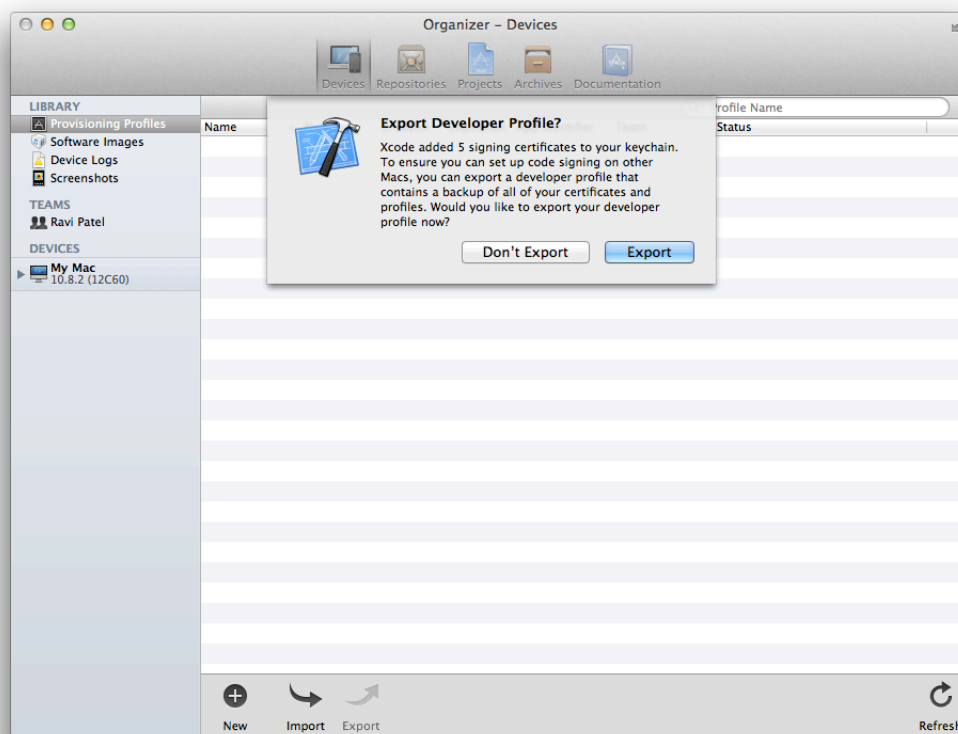


4. In the dialog that appears, enter your Apple ID user name and password, and click “Log in.”  
After you log in to your account, multiple dialogs appear, asking whether Xcode should request certain types of signing certificates on your behalf. If you just joined the Mac Developer Program, the first dialog asks whether Xcode should request your Mac Development certificate. The last two dialogs ask whether Xcode should request your Developer ID certificates.
5. Click the Submit Request button each time a certificate request dialog appears.  
After you submit the last certificate request, allow the refresh process to complete. Your Developer ID Application and Developer ID Installer certificates are added to your keychain.



6. In the dialog that appears at the end of the refresh process, asking whether you want to export your developer profile, click Export.

You should always back up your certificates after you create them. The private keys for your certificates are stored in your keychain, and the public keys are stored in Member Center. As a result, you can't refresh your provisioning profiles and certificates in Xcode to replace a missing private key in your keychain.



7. Enter a filename and password, and click Save.

Because the file contains your developer profile, which can be used to sign applications in your name, it is encrypted and password protected. (You will need the password later to import your developer profile to another Mac.)

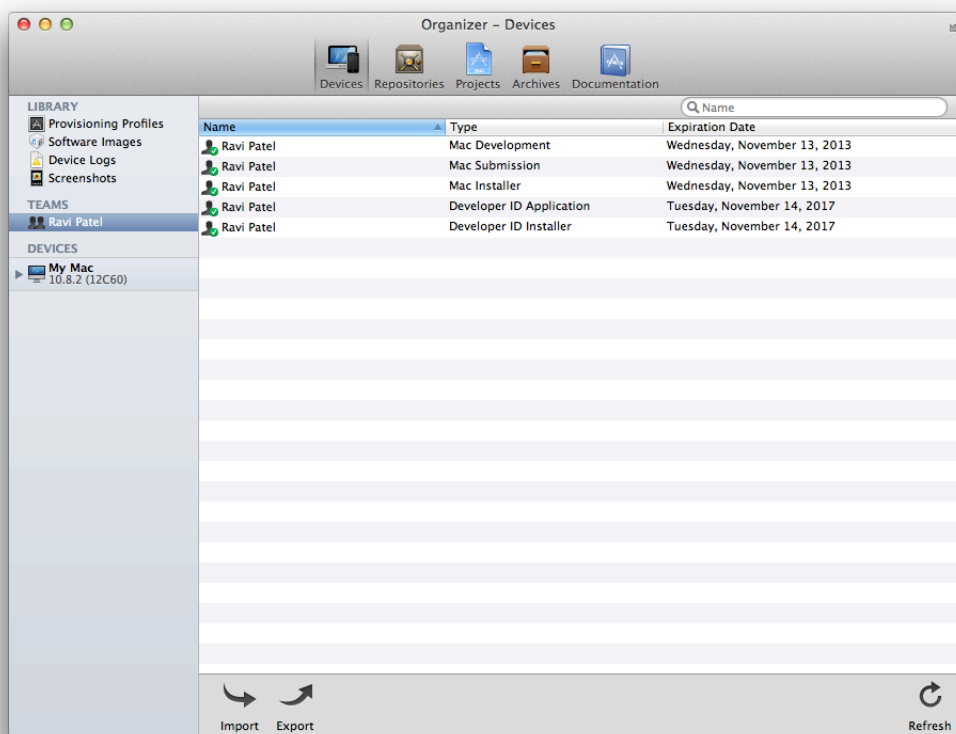
**Important:** Your Developer ID private keys are valuable, and you should back them up. Exporting your developer profile lets you create a password-protected backup. Save that backup as you would any essential backup; for example, save it to a different disk. Later, if you need to replace a private key, import it from your backup.

Your Developer ID Certification Authority intermediate certificate, which is required for Developer ID code signing, is not exported. If you need to obtain another copy, retrieve it from Apple at <https://developer.apple.com/certificationauthority/DeveloperIDCA.cer>.

If you need to export your developer profile later, follow the steps in “Exporting Your Developer Profile” (page 159).

## Verify Your Steps

To verify your steps, view your Developer ID certificates in your Teams folder in the Devices organizer.



## Code Signing Your Application

Optionally, code sign your application during development and testing using the Developer ID Application certificate. Later, you re-sign the application with this certificate when you archive and export it from Xcode.

### To code sign an application with your Developer ID Application certificate

1. In the Xcode project editor, select the target.

**Important:** If you want to sign multiple targets with the same code signing identity, select the project, not a target.

2. Select the Build Settings tab.

3. Click All.
4. Type Code Signing in the search field in the Build Settings pane of the project editor.  
The list of build settings now shows only the Code Signing settings.
5. From the Code Signing Identity pop-up menu, choose your Developer ID Application certificate.
6. Click Run.

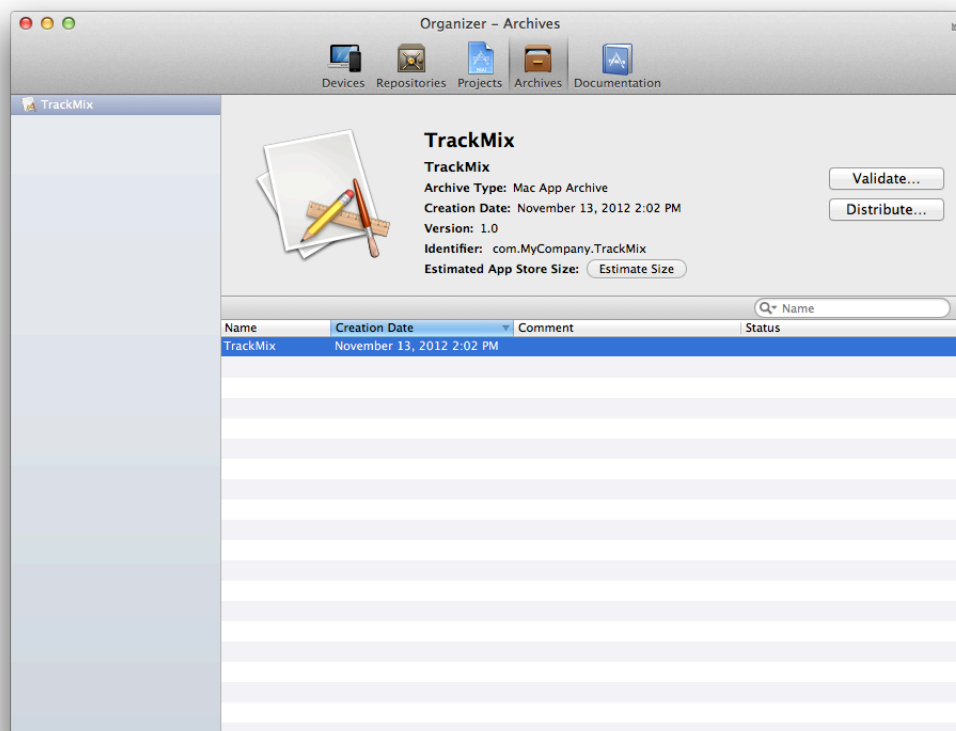
## Exporting a Developer ID-Signed Application

To export your application for distribution outside of the Mac App Store, use the Archives organizer.

### To create a Developer ID-signed application

1. In Xcode, choose Product > Archive.

Xcode constructs an archive containing your code-signed application and opens the Organizer window, showing the archive.

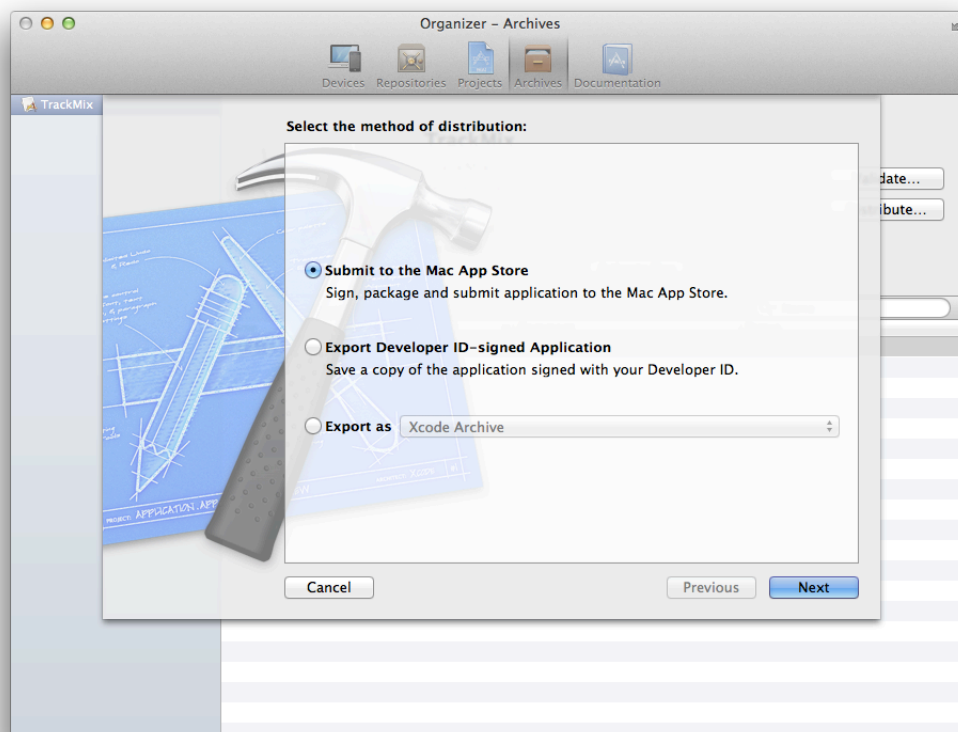


---

**Note:** You can set the Code Signing Identity build setting to any valid signing certificate during this step because the archive is re-signed with the Developer ID certificate in a later step.

---

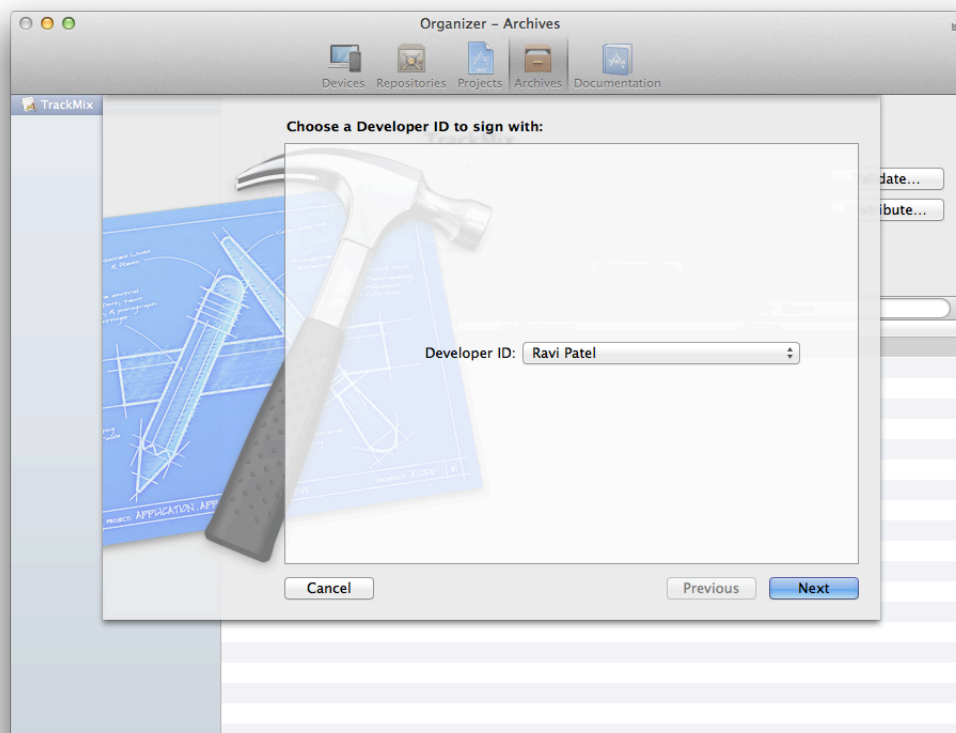
2. Select the newly created archive in the Organizer window, and click Distribute.
3. In the dialog that appears, offering a choice of distribution methods, select “Export Developer ID-signed Application” and click Next.



4. Choose your Developer ID certificate name from the Developer ID pop-up menu and click Next.



If you are an individual developer, your name appears in the pop-up menu; otherwise, your company name appears in the pop-up menu.



5. Enter a filename and location to save the signed application and click Save.

## Signing an Installer Package

If you want to distribute your application outside the Mac App Store as part of an installer package, create the package as you normally do. One way to create the installer package is to use the `packagemaker` (1) command-line utility. Code sign the package with your Developer ID Installer certificate with the `productsign` command. To test your installer package use the following command and replace `MyPackageName.pkg` the filename of your package:

```
spctl -a -v --type install MyPackageName.pkg
```



**Warning:** Make sure you sign the installer package using your Developer ID Installer certificate. The `productsign` command-line utility allows you to sign an installer package using your Developer ID Application certificate. Although this approach may appear to work, the resulting installer archive will fail on the destination Mac.

If your development workflow includes code signing from the command line, read *Code Signing Guide*.

## Verify Your Steps

Before you distribute your application, test the end-user experience by launching your application with Gatekeeper enabled and disabled. You can enable and disable Gatekeeper using System Preferences or the `spctl(8)` command-line utility. This command-line utility is also useful for testing. To simulate the end-user experience, you need to quarantine your application and test it again with Gatekeeper enabled.

## Enabling and Disabling Gatekeeper

You can turn on Gatekeeper by using the Security & Privacy system preferences or the `spctl(8)` command-line utility for system policy control. Gatekeeper system preferences are hidden by default, but you can show them using the `spctl(8)` command-line utility.

### To enable or disable Gatekeeper using the Security & Privacy system preferences

1. In the Finder, launch System Preferences and select Security & Privacy.
2. Click the lock button if it appears locked, and enter the administrator password.

3. To enable Gatekeeper, select “Mac App Store and identified developers.”



4. To disable Gatekeeper, select Anywhere.

### To enable Gatekeeper using the `spctl` command

1. In Terminal, enter the following command:

```
$ spctl --master-enable
```

When prompted, enter your administrator password.

2. To confirm that Gatekeeper is enabled, enter the following command:

```
$ spctl --status
```

With Gatekeeper enabled, the previous command prints the following text in Terminal:

```
assessments enabled
```

## To disable Gatekeeper using the `spctl` command

1. In Terminal, enter the following command:

```
$ spctl --master-disable
```

When prompted, enter your administrator password.

2. To confirm that Gatekeeper is disabled, enter the following command:

```
$ spctl --status
```

With Gatekeeper disabled, the previous command prints the following text in Terminal:

```
assessments disabled
```

---

**OS X v10.7 Note:** Gatekeeper is available in OS X v10.7 and later. However, in OS X v10.7, the Gatekeeper system preferences are hidden. To show Gatekeeper system preferences, enter the following command in Terminal:

```
defaults write com.apple.systempreferences  
ShowGatekeeperOptionsInSecurityPreferences -bool YES
```

To hide Gatekeeper system preference, enter this command in Terminal:

```
defaults write com.apple.systempreferences  
ShowGatekeeperOptionsInSecurityPreferences -bool NO
```

---

## Testing Gatekeeper Behavior

After signing your application with a Developer ID certificate, you can test whether it was signed correctly and simulate the launch behavior of your application when Gatekeeper is enabled. On a Mac with Gatekeeper enabled, a quarantined copy of your application launches only if it is Developer ID signed. (Learn about quarantine in this [Knowledge Base article](#).) You can also test the behavior of Gatekeeper for an application that is not Developer ID signed.

## Testing a Developer ID-Signed Application

You can use the `spctl` command-line utility to test whether your application is signed correctly using a Developer ID certificate.

### To test your Developer ID-signed application using `spctl`

1. Enable Gatekeeper on your test machine by entering the following command in Terminal:

```
$ spctl --master-enable
```

2. Enter the following command in Terminal by replacing `TrackMix.app` with the path to your application.

```
$ spctl -a -v TrackMix.app
```

If the application is correctly signed, text similar to the following appears in Terminal:

```
./TrackMix.app: accepted  
source=Developer ID  
override=security disabled
```

## Testing the Launch Behavior

To thoroughly test your Developer ID-signed application, simulate launching the application on a Mac not used for development.

### To prepare for testing Gatekeeper behavior

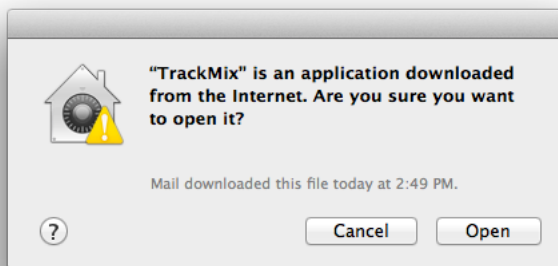
1. Enable Gatekeeper on your test machine (as described in [“Enabling and Disabling Gatekeeper”](#) (page 202)).
2. Quarantine a copy of your Developer ID-signed application. You can do this in either of the following ways:
  - Email your Developer ID-signed application to yourself and use the copy that Mail downloads.
  - Host your Developer ID-signed application on your own local or remote server and use the copy that Safari downloads.

You are ready to test Gatekeeper behavior.

## To test Gatekeeper behavior for your Developer ID-signed application

- In the Finder, locate the quarantined copy of your Developer ID-signed application and double-click its icon.

The Mac displays an alert asking whether you are sure you want to open the application.



This alert, which allows you to open the quarantined application with Gatekeeper enabled, confirms that your Developer ID workflow is correct.



**Tip:** If you do not see an alert at this point, it is likely that you have opened a nonquarantined copy of your application. Review the steps in [“To prepare for testing Gatekeeper behavior”](#) (page 205).

## To test Gatekeeper behavior for blocking applications that are not Developer ID signed

1. Enable Gatekeeper on your test machine (as described in [“Enabling and Disabling Gatekeeper”](#) (page 202)).
2. Quarantine a copy of your application that is not Developer ID signed.

As before, you can invoke quarantine on this copy of your application in either of the following ways:

- Email your application to yourself and use the copy that Mail.app downloads.
- Host your Developer ID-signed application on your own local or remote server and use the copy that Safari downloads.

3. In the Finder, locate the quarantined copy of your non-Developer ID-signed application and double-click its icon.

The Mac displays an alert that blocks you from opening the application. By way of this alert, Gatekeeper protects a Mac by preventing first-time opening of applications from unidentified developers. Applications previously opened by a user are no longer quarantined, and Gatekeeper does not prevent them from launching.

## Recap

This chapter showed you how to distribute your Mac application outside of the Mac App Store so that users won't block your app from launching.

# Troubleshooting

Just as troubleshooting is a part of every complex technical process, it is a necessary part of developing, testing, submitting, and releasing an app. The potential problems you may encounter are organized here in four general areas—certificates, provisioning, building, and debugging—with each problem followed by specific advice. Use this chapter as a reference to find solutions to the problems you might encounter.

## Certificate Issues

Before you re-create a certificate that has become invalid or unusable, see whether it has one or more of the following common problems.

### Your Provisioning Profile Doesn't Appear in the Code Signing Identity Menu

If your provisioning profile doesn't appear in the Code Signing Identity menu (in the Build Settings pane of the project editor in Xcode), first refresh provisioning profiles in Xcode. If your provisioning profile still doesn't appear, select Don't Code Sign or a certificate under Automatic Profile Selector from the Code Signing Identity menu. The next time you select the Code Signing Identity menu, your provisioning profile should appear in the menu. If it still doesn't appear, quit and relaunch Xcode.

### Duplicate Provisioning Profile Appear in the Devices Organizer

If duplicate valid provisioning profiles appear in the Devices organizer in Xcode, delete all the provisioning profiles in the Devices organizer and refresh provisioning profiles in Xcode. If duplicate provisioning profiles still appear, sort the provisioning profiles by expiration date and delete the older versions of the duplicates.

### Your Certificates Are Invalid Because You're Missing Private Keys

Certificates might be invalid because the corresponding private key is not in your keychain. Try to restore your missing private keys from a developer profile backup, as described in [“Exporting and Importing Certificates and Provisioning Profiles”](#) (page 159). If you cannot retrieve your private keys from another Mac, refer to [“Re-Creating Certificates and Updating Related Provisioning Profiles”](#) (page 181) to re-create all your certificates. You can perform these steps for one or more invalid certificates.



## Your Developer ID Certificates Are Invalid Because You're Missing Private Keys

Follow the same steps in ["Your Certificates Are Invalid Because You're Missing Private Keys"](#) (page 208) to repair Developer ID certificates, except contact Apple at [product-security@apple.com](mailto:product-security@apple.com) if you need to revoke Developer ID certificates. Alternatively, you can continue to develop and distribute applications by requesting additional Developer ID certificates as described in ["Requesting Additional Developer ID Certificates"](#) (page 168).

## Your Certificates Are Invalid Because You're Missing an Intermediate Certificate

If your certificates are invalid, you could be missing the intermediate certificate used to authenticate your certificate. If you verify your certificate in Keychain Access (see ["To verify that your device is registered"](#) (page 43)) and instead of a green circle with a checkmark, a red circle with a white X appears with the status "This certificate was signed by an unknown authority," you are missing the intermediate certificate. If you do not have a certificate called *Apple Worldwide Developer Relations Certification Authority* in your system keychain, read ["Installing Missing Intermediate Certificate Authorities"](#) (page 167) to learn how to reinstall it. The intermediate certificate for Developer ID certificates is called the *Developer ID Certification Authority*.

## Your Certificates Have Trust Issues

If you view your certificate in Keychain Access, and a blue circle and white plus sign appear in the detail area instead of a green circle with a checkmark, your certificate has trust issues. To learn how to fix this problem, read ["Xcode Doesn't Trust Your Certificate"](#) (page 211).

## Your Certificates Have Expired

You cannot renew expired certificates. Read ["Replacing Expired Certificates"](#) (page 166) for how to remove the expired certificates and request new ones.

---

**Mac Note:** If your Developer ID certificates expire, users can still download, install, and run versions of your Mac applications that were signed with these certificates. However, you will need new Developer ID certificates to sign updates and create new applications.

---

## You're Missing Signing Certificates

Your signing certificates may be missing from your keychain because you never requested them or because you moved to a Mac on which you haven't developed apps before.

If you never requested your certificates (there are none in your keychain), refresh the provisioning profiles in Xcode to request them, as described in ["To verify that your device is registered"](#) (page 43).

If you moved to a new Mac, export your certificates as a developer profile file on the Mac you first requested the certificates from, and then import them on your new Mac, as described in [“Exporting and Importing Certificates and Provisioning Profiles”](#) (page 159).

If you no longer have access to the other Mac or user account and you did not keep a backup of your certificates, refer to [“Re-Creating Certificates and Updating Related Provisioning Profiles”](#) (page 181) to re-create all of your certificates.

## You Have Duplicate Certificates

If you have other certificates in your keychain from other accounts, you should remove them, as described in [“Your Developer ID Certificates Are Invalid Because You’re Missing Private Keys”](#) (page 209). You should have only one of each type of certificate. Duplicates may cause problems when you attempt to sign your app. For a list of the certificate types, refer to [Table 2-1](#) (page 36).

## Provisioning Issues

Common provisioning issues result from using the wrong provisioning profile with your app or using an invalid or expired provisioning profile.

### Xcode Cannot Install Your App on Your Development Device

If Xcode cannot install your app on your development device because of a problem with the provisioning profile you’re using with the app, ensure that the provisioning profile is properly configured in your development team’s signing assets. To configure provisioning profiles, see [“Configuring Store Technologies in Xcode and iTunes Connect”](#) (page 79).

### Your Provisioning Profile Has Expired

If the provisioning profile stored on the development device for your app has expired, Xcode won’t be able to install the app on that device. Replace the expired provisioning profile as described in [“Renewing Expired Provisioning Profiles”](#) (page 180).

## Build and Code Signing Issues

Common build errors tend to involve incorrect code signing identities. For iOS apps, your device might not appear in the run destination menu.

## Xcode Cannot Find Your Provisioning Profile

You will receive the following error message after replacing a provisioning profile with a modified version, such as when a provisioning profile's App ID changes:

```
Code Sign error: Provisioning Profile 'xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxx' can't be found
```

To address this error, ensure that the correct provisioning profile and code signing identity are selected for the value of the Code Signing Identity build setting. This error may also occur if the project's and target's Code Signing Identity build settings are different. See [“Signing Your App Using Your Development Provisioning Profile”](#) (page 71).

## Xcode Doesn't Trust Your Certificate

You get this error message when Xcode cannot verify the authenticity of your development or distribution certificate:

```
Code Sign error: CSSMERR_TP_NOT_TRUSTED
```

If the trust setting is not Use System Defaults, you will receive a `CSSMERR_TP_TRUSTED` error message from the `codesign` command-line utility when you build and run your app. Do not change the trust settings of your certificates from the default Use System Defaults. Follow these steps to repair your development, distribution, and intermediate certificates.

### To set the trust level of a certificate to the system defaults

1. Launch Keychain Access.
2. In the Category section, select My Certificates.
3. Double-click the certificate.
4. In the certificate window, display the Trust section by clicking the corresponding disclosure triangle.
5. For the option “When using this certificate,” select Use System Defaults.
6. Close the certificate window.
7. Ensure that the certificate information shows the certificate is valid.

## The Code Signing Identity Build Setting Doesn't Match Any Certificates

You will receive the following error message when your certificate has expired or is otherwise invalid:

Code Signing Identity 'iPhone Developer' doesn't match any valid, non-expired, certificate/private key pair in your keychain.

For multiple targets that use the same code signing identity, you should set this build setting at the project level, not the target. However, if it is set on both the project and the target, the target setting overrides the project setting. If the target's Code Signing Identity build setting is set, delete it by first selecting it and then choosing Edit > Delete. Finally, set the project's Code Signing Identity build setting to your certificate.

If your development certificate or team provisioning profile doesn't appear in the Code Signing Identity menu, try refreshing the provisioning profiles and choose a valid code signing identity as described in ["Code Signing Your App Using the Team Provisioning Profile"](#) (page 49) or ["Signing Your App Using Your Development Provisioning Profile"](#) (page 71).

### To refresh provisioning profiles

1. In Xcode, open the Devices organizer.
2. Select Provisioning Profiles under Library.
3. Click Refresh at the bottom of the window.

## Your Keychain Contains Duplicate Code Signing Identities

You get one of these error messages when there are duplicate code signing identities in your keychain, such as two development identities or two distribution identities (your keychain must contain at most one code signing identity of each type):

```
Build error "iPhone Developer: <your_name> (XYZ123ABC): ambiguous (matches "iPhone Developer: <your_name> (XYZ123ABC)" in /Library/Keychains/System.keychain and "iPhone Developer: <your_name> (XYZ123ABC)" in /Users/./Library/Keychains/login.keychain)"
```

```
[BER0R]CodeSign error: Certificate identity 'iPhone Distribution: <your_name>' appears more than once in the keychain. The codesign tool requires there only be one.
```

To address these errors, first try deleting the duplicate code signing identities from your keychain. If that approach doesn't work, follow the steps in ["Importing Your Developer Profile"](#) (page 160).

## The App ID of Your Provisioning Profile Doesn't Match Your App's Bundle Identifier

When there's a conflict between the App ID in the provisioning profile selected in the Code Signing Identity build setting and your app's bundle identifier, you get error messages similar to the following:

```
Code Sign error: Provisioning profile 'MyApp Profile' specifies the Application Identifier 'com.mycompany.MyApp.*' which doesn't match the current setting 'com.mycompany.MyApp'
```

To address these errors, ensure that your bundle identifier is set correctly in your Xcode project, that the certificate and provisioning profile specified in the Code Signing Identity build setting is correct, and that the provisioning profile uses the correct App ID.

## Device Is Not Listed as a Run Destination

If you have a project or workspace open and your connected device is not listed as a run destination in the Scheme toolbar menu, verify that:

1. The app's targeted iOS version is equal to or greater than the iOS version installed on your device. See [“Setting the Target iOS Devices”](#) (page 105) for details.
2. Your device contains a valid provisioning profile.
3. The version number of the iOS SDK your project uses is equal to or greater than the version number of the iOS version on your device.

For example, if Xcode shows iOS SDK 4.3 but your device has iOS 5.0 installed, you need to install on your Mac an Xcode version that includes iOS SDK 5.0.

## Debugging Information Issue

The most common potential problem with debugging is that Xcode has not yet collected the information from your device.

## Xcode Displays the Unknown iOS Detected Dialog When You Connect a Device

This message appears when Xcode hasn't seen a particular version of iOS before and needs to download (from the device) the debugging symbols for that version from the device. To successfully debug apps on the device, leave the device connected until Xcode finishes downloading the debugging symbols from the device.

# Document Revision History

This table describes the changes to *App Distribution Guide*.

Date	Notes
2013-04-23	Applied minor edits throughout.
2013-04-05	New document that describes the common workflows to develop, test, and distribute your app.

# Glossary

**ad hoc provisioning profile** A type of distribution provisioning profile used for distributing an iOS app for testing.

**App ID** A string that identifies one or more apps from a single team. An App ID consists of a [bundle ID search string](#) preceded by the [Team ID](#), a 10-character string generated by Apple to uniquely identify a team.

**Apple Developer Program** Subscription services that offer Apple developers access to technical resources and support to develop apps for the App Store and Mac App Store. Developers can join one or more of the separate programs for iOS, Mac, and Safari development.

**Apple ID** An Apple-issued developer account with a name and password. Developers use their Apple ID credentials to sign in to any of the developer program tools. A developer or Apple ID can belong to multiple teams, and teams can belong to multiple types of developer programs.

**Apple Push Notification service (APNs)** The service (servers and other infrastructure) that Apple provides to allow developers to push notifications to apps. A message sent by the service is called a [push notification](#).)

**Apple Worldwide Developer Relations Certification Authority** The certificate authority that validates development and distribution certificates for apps submitted to the App Store and the Mac App Store.

**App Store** A service for purchasing and downloading iOS apps. The App Store is available on iOS devices, and in the iTunes Store on Mac and Windows computers.

**bundle ID** A reverse DNS string that precisely identifies a single app.

**bundle ID search string** The second part of an [App ID](#) that is supplied by developers to match a set of bundle IDs, where each bundle ID identifies a single app. For example, if the bundle ID search string is `com.mycompany.MyApp` or a wildcard such as `com.mycompany.*`, then it will match the bundle ID `com.mycompany.MyApp`.

**certificate authority** An organization that authorizes a certificate.

**certificate signing request (CSR)** A file that contains personal information used to generate a signing certificate. This file also contains the public key to be included in the certificate, along with identifying information.

**Certificates, Identifiers & Profiles** An area of the Member Center available to iOS, Mac, and Safari Developer Program members that provides resources needed to develop iOS and Mac apps and Safari Extensions.

**client SSL certificate** A certificate that allows a developer's server to connect to an Apple service. For example, developers use a client SSL certificate to communicate with the Apple Push Notification service.

**code signing certificate** A signing certificate used to sign an app or installer.

**code signing identity** A digital identity used for code signing, including archive signing. A code signing identity includes the certificate with its private and public keys.

**company** A type of Apple Developer Program account that has one or more team members.

**crash report** A report generated by the operating system when an app crashes.

**data protection** A digital safeguard that adds a level of security to files stored on disk by an app.

**Developer ID** The name of the feature that developers use to distribute code-signed applications outside the Mac App Store.

**developer profile** A file that contains a developer's development certificates, distribution certificates, and provisioning profiles.

**development certificate** A type of signing certificate used during development that identifies a single developer on a team. It allows an app to launch on a device through Xcode.

**development provisioning profile** A type of provisioning profile that authorizes an app to use certain technologies and run on designated devices during development. This profile consists of a name, multiple development certificates, multiple devices, and an App ID.

**device** Used to refer to a Mac computer—or to an iPad, iPhone, or iPod—when no further distinction between them is needed.

**device ID** A way of uniquely identifying an iOS or Mac device.

**distribution certificate** A type of signing certificate used to distribute an app and allow it to launch on a device without the assistance of Xcode. A distribution certificate identifies a team, not a team member.

**distribution provisioning profile** A type of provisioning profile that authorizes an app to run on devices without the assistance of Xcode and allows them to use certain technologies. A distribution provisioning profile is used to submit an app to the App Store or Mac App Store. Mac has one type of distribution provisioning profile, and iOS has two.

**entitlement** A single right granted to a particular app, tool, or other executable that gives it additional permissions beyond what it would ordinarily have.

**explicit App ID** An App ID that matches a single bundle ID, in contrast to a wildcard App ID, which can match one or more bundle IDs.

**Game Center** Apple's social gaming network that allows players to connect to the service and exchange information with other players.

**Gatekeeper** The OS X feature that enables users to choose to disallow the launching of applications that are not code signed by developers known to Apple.

**iCloud** A type of storage that allows developers to share a user's data among multiple instances of an app running on other iOS and Mac OS X devices.

**In-App Purchase** A mechanism for embedding items to purchase directly into an app. In this way, a developer can connect to the App Store or Mac App Store and securely process payments from the user.

**individual** Used to describe a type of Apple Developer Program account that has one developer.



**intermediate certificate** A certificate that is required to be in a developer's keychain to ensure that a signing certificate is issued by a trusted source.

**iOS App Store Package** A type of OS X file that, when double-clicked installs an app in iTunes, where it can be synced to an iOS device.

**iOS Dev Center** An Apple developer center that provides all the resources needed to develop iOS applications.

**iOS Developer Program** A program that allows developers to develop iOS apps, test them on iOS-based devices, and distribute them to users.

**Mac Developer Program** A program that allows developers to develop Mac apps, and distribute them to users.

**Mac Installer Package** A type of OS X file that, when double-clicked, launches the Installer and installs a Mac app on a computer.

**Newsstand** An iOS app for purchasing and organizing newspaper and magazine subscriptions into a folder.

**Passbook** An iOS app for organizing and using passes, tickets, and coupons.

**passes** Digital representations of information that allow users to redeem a real-world product or service, such as a coupon, a ticket for a show, or a boarding pass.

**provisioning** The process of preparing and configuring an app to launch on devices and use certain services.

**provisioning profile** A type of system profile used to provision one or more apps.

**push notification** A message sent from an app, that is not running in the foreground, to the user using [Apple Push Notification service \(APNs\)](#).

**quarantine** The state of a file or an application that, when a user first attempts to open the item, triggers the Gatekeeper feature. OS X imposes a quarantine on items downloaded from the web, from email, and so on.

**routing app** An app that offers routing information, such as turn-by-turn navigation services. An app can register as a routing app and make those directions available to Maps and other apps.

**signing certificate** A certificate used for signing other entries, such as installer packages, email messages, and the like.

**store** Used as a short form of the App Store or the Mac App Store when there is no distinction between the two.

**symbolicate** To replace memory addresses in a crash report with human-readable function names and line numbers.

**team admin** A person on a development team who has some of the privileges of a team agent but can't sign agreements. Team admins help team agents delegate some of their responsibilities. Compare [team agent](#); [team member](#).

**team agent** The person on a development team who has unrestricted access to the team and who is legally responsible for it. Compare [team admin](#); [team member](#).

**Team ID** A 10-character string that is generated by Apple to uniquely identify your team. The Team ID is used as the prefix for an [App ID](#).

**team member** A person on a development team who has the fewest privileges. A team member can sign apps during development after that request is approved by a team admin. Compare [team agent](#); [team admin](#).

**team provisioning profile** The development provisioning profile that Xcode creates and manages for you. The team provisioning profile contains all of a team's development certificates, its registered devices, and the wildcard App ID, which Xcode also creates.

**wildcard App ID** An App ID that matches one or more bundle IDs used by a development team. Compare [explicit App ID](#).

**Xcode iOS Wildcard App ID** The [wildcard App ID](#) that Xcode manages for iOS developers.

**Xcode Mac Wildcard App ID** The [wildcard App ID](#) that Xcode manages for Mac developers.



Apple Inc.  
© 2013 Apple Inc.  
All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, electronic, photocopying, recording, or otherwise, without prior written permission of Apple Inc., with the following exceptions: Any person is hereby authorized to store documentation on a single computer for personal use only and to print copies of documentation for personal use provided that the documentation contains Apple's copyright notice.

No licenses, express or implied, are granted with respect to any of the technology described in this document. Apple retains all intellectual property rights associated with the technology described in this document. This document is intended to assist application developers to develop applications only for Apple-labeled computers.

Apple Inc.  
1 Infinite Loop  
Cupertino, CA 95014  
408-996-1010

Apple, the Apple logo, eMac, Finder, Instruments, iPad, iPhone, iPhoto, iPod, iPod touch, iTunes, Keychain, Mac, Mac OS, Mac Pro, OS X, Passbook, Safari, Sand, Spotlight, and Xcode are trademarks of Apple Inc., registered in the U.S. and other countries.

Retina is a trademark of Apple Inc.

.Mac, iAd, iCloud, and iTunes Store are service marks of Apple Inc., registered in the U.S. and other countries.

App Store and Mac App Store are service marks of Apple Inc.

iOS is a trademark or registered trademark of Cisco in the U.S. and other countries and is used under license.

**Even though Apple has reviewed this document, APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS DOCUMENT IS PROVIDED "AS IS," AND YOU, THE READER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.**

**IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT OR INACCURACY IN THIS DOCUMENT, even if advised of the possibility of such damages.**

**THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. No Apple dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.**

**Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.**