App Distribution Guide

Developer

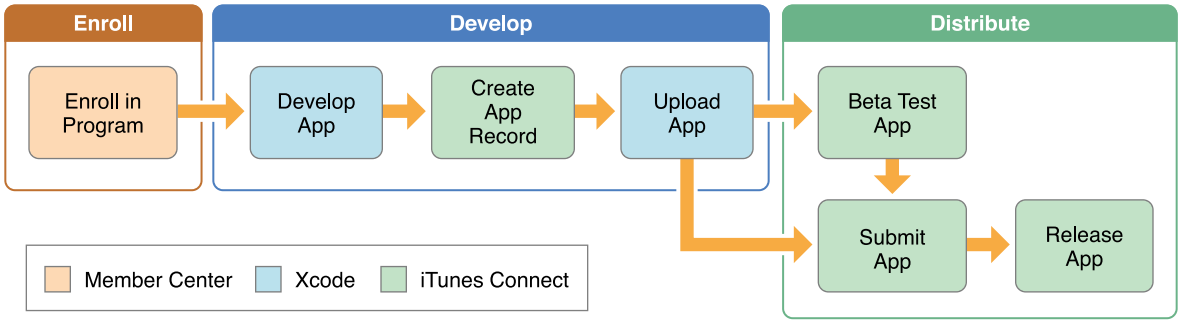
Contents

Tables

About App Distribution

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

- Get step-by-step guidance for enrolling in an Apple Developer Program and building, testing, and submitting your app.
- Configure services that are available only to apps submitted to the App Store or Mac App Store.
- Test your app on multiple devices and system versions, or offer testers a preview of your next release.
- Upload metadata about your app so the store can present it to customers.
- Verify that you've prepared your app correctly, and submit it to the store.
- Learn how to release and maintain your app after submission.



You perform these tasks using Xcode features and several web tools available only to members of an Apple Developer Program. Before you use certain app services, such as iCloud and Game Center, you must join an Apple Developer Program. Join a program even if you distribute an application outside of the store so that customers know your application comes from a known source.

Note: If you just want to use Xcode to run an app on a device or write code that uses a service, read *App Distribution Quick Start* first and then return to this document for additional tasks you'll perform throughout the lifetime of your app.

At a Glance

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.

Enroll in an Apple Developer Program to Distribute Your App

Joining an Apple Developer Program is the first step to submit your apps on the App Store and Mac App Store, to distribute iOS in-house apps, or to sign apps that you distribute outside the Mac App Store with a Developer ID. As a member, you have access to the resources you need to configure app services and to submit new apps and updates.

Related Chapter: Managing Accounts (page 18)

Add Services to Your App

Apple 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. These app services require additional configuration—both during development and later, when you submit your app to the App Store or Mac App Store. Good examples are Game Center and iCloud. In this guide, you'll learn how to add these capabilities to your app.

Related Chapter: Adding Capabilities (page 48), Configuring Push Notifications (page 76)

Prepare Your App for Distribution

Before you distribute your app for testing or submit it to the store for approval, complete the configuration of your Xcode project. Your final Xcode project should contain required app icons and launch images, contain additional entitlements for app services you enable, and specify which devices and operating systems your app supports.

Related Chapter: Configuring Your Xcode Project for Distribution (page 23)

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 per membership year for development and testing. After testing an app yourself, distribute a beta release of your app to testers. You can distribute a beta app yourself or use iTunes Connect to manage beta testing. For iOS apps distributed through TestFlight and the App Store, Apple provides a service that collects and aggregates crash logs that you can download and analyze in Xcode.

Related Chapters: Beta Testing iOS Apps (page 97), Analyzing Crash Reports (page 114)

Submit and Release Your App on the Store

Submitting your app to the App Store or Mac App Store is a multistep process. First, you sign in to iTunes Connect to create an app record and enter necessary information. If you're selling your app on the store, you also enter the information for your reimbursement in iTunes Connect. In Xcode, you create an archive and sign it with your distribution certificate. Then you upload your app using Xcode or Application Loader. Use iTunes Connect to submit your app to the store. When your app is approved, use iTunes Connect to release it by setting the date when the app will be available to customers.

Related Chapters: Submitting Your App to the Store (page 126), Releasing and Updating Your App on the Store (page 136), Managing Your App in iTunes Connect (page 137), Distributing iOS Developer Enterprise Program Applications (page 222), Distributing Applications Outside the Mac App Store (page 230)

Distribute Your App Outside the Store

Alternatively, join the iOS Developer Enterprise Program and distribute your in-house applications directly to employees. To distribute a Mac app outside of the Mac App Store, request and sign your application with a Developer ID certificate. If you're distributing your application outside the store, you follow a slightly different process. You don't have access to iTunes Connect and some app services so can skip those steps.

Related Chapters: Distributing iOS Developer Enterprise Program Applications (page 222), Distributing Applications Outside the Mac App Store (page 230)

Maintain Your Certificates, Identifiers, and Profiles

Apple implements an underlying security model to protect both user data and your app from being modified and distributed without your knowledge. Throughout the development process, you create assets and enter information that Apple uses to identify you, your devices, and your apps. Xcode automatically creates many certificates, identifiers, and profiles for you as you need them. Xcode maintains the App IDs and provisioning profiles it creates for you, but not the other assets. During your Developer Program membership, you may maintain various other certificates, identifiers, and profiles yourself.

Related Chapters: Maintaining Your Signing Identities and Certificates (page 153), Maintaining Identifiers, Devices, and Profiles (page 184)

How to Use This Document

How you use this document depends on the type of developer program you join (iOS Developer Program, Mac Developer Program, or iOS Developer Enterprise Program) and your role (team agent, admin, or member). For Mac apps, how you use this document also depends on whether you choose to submit your app to the Mac App Store or distribute it outside of the Mac App Store.

First choose a type of account (individual or company) and a developer program. If needed, create an Apple ID and join a developer program, as described in Managing Accounts (page 18). If you enroll in a developer program as an individual, you're the team agent for a one-person team. If you enroll in a developer program as a company, you're the team agent and can invite other people to join your team, as described in Inviting Team Members (page 147). You specify whether a person is a team admin, who can perform most of the same tasks as a team agent, or a team member who can't create assets in Member Center. To learn more about team roles, read About Apple Developer Program Team Roles and Privileges (page 144).

Then refer to the tables in this section for the tasks you perform depending on your role and developer program membership. (Refer to the glossary for the definitions of terms used in this guide.)

If you're a team agent or admin and want to submit your app to the App Store or Mac App Store:

	To learn how to	Read
☐	Add your Apple ID to Xcode	Adding Your Apple ID Account in Xcode (page 20)

	To learn how to	Read
☐	Set your bundle ID and assign your project to a team	Configuring Identity and Team Settings (page 26) Configuring Your Xcode Project for Distribution (page 23)
☐	Configure app services	Adding Capabilities (page 48)
☐	Launch your app on devices	Launching Your App on Devices (page 88)
☐	Perform final configuration steps before distributing your app	Configuring Your Xcode Project for Distribution (page 23)
☐	Test your iOS app on different devices	Beta Testing iOS Apps (page 97)
☐	Fix problems during testing	Analyzing Crash Reports (page 114)
☐	Upload your app to iTunes Connect for approval	Submitting Your App to the Store (page 126)
☐	Release your app by setting the availability date	Releasing and Updating Your App on the Store (page 136) Managing Your App in iTunes Connect (page 137)
☐	Maintain your Apple Developer Program assets	Maintaining Your Signing Identities and Certificates (page 153) Maintaining Identifiers, Devices, and Profiles (page 184)
☐	Fix issues with your code signing assets	Troubleshooting (page 241)

If you're a team agent or admin for a company:

	To learn how to	Read
☐	Add team members and assign roles for company accounts	Managing Your Team in Member Center (page 144)

If you're a team member for a company who is developing an app for the App Store or Mac App Store:

	To learn how to	Read
☐	Add your Apple ID to Xcode	Adding Your Apple ID Account in Xcode (page 20)
☐	Set your bundle ID and assign your project to a team	Configuring Identity and Team Settings (page 26) Configuring Your Xcode Project for Distribution (page 23)
☐	Request your development certificate and ask your team agent or admin to approve it	Approving Development Certificates (page 149)
☐	Ask your team agent or admin to register your device	Registering Team Member Devices (page 150)
☐	Launch your app on devices	Launching Your App on Devices (page 88)
☐	Fix issues with your code signing assets	Troubleshooting (page 241)

If you're a team agent or admin in the iOS Developer Enterprise Program:

	To learn how to	Read
☐	Manage your certificates and distribute your app	Distributing iOS Developer Enterprise Program Applications (page 222)

If you're a team agent and want to distribute your Mac application outside of the Mac App Store:

	To learn how to	Read
☐	Perform final configuration steps before distributing your app	Configuring Your Xcode Project for Distribution (page 23)
☐	Create a Developer ID-signed application	Distributing Applications Outside the Mac App Store (page 230)
☐	Request additional Developer ID certificates	Requesting Additional Developer ID Certificates (page 162)

For Mac apps, if you select None as the distribution method, as described in Choosing a Signing Identity (Mac Only) (page 29), you don't need to read this guide.

See Also

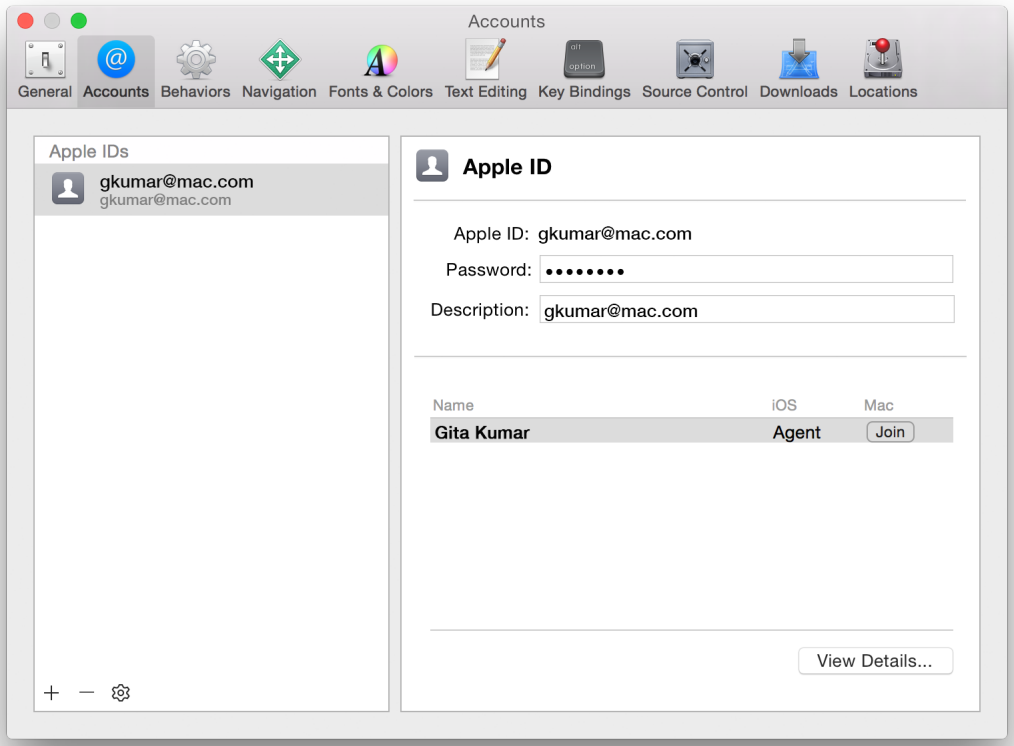
This guide assumes you are already familiar with the software and tools you use to write code. If not, start by reading a number of platform-specific tutorials. Next, read the technology overview documents followed by the appropriate human interface guidelines for your platform, and most important, the guidelines for submitting your app to the store.

	iOS	Mac
To get started . . .	*Start Developing iOS Apps Today* *App Distribution Quick Start*	*Start Developing Mac Apps Today* *App Distribution Quick Start*
To learn more about technologies . . .	*iOS Technology Overview* *App Programming Guide for iOS*	*Mac Technology Overview* *Mac App Programming Guide*
To learn about the user interface guidelines . . .	*iOS Human Interface Guidelines* *App Store Review Guidelines for iOS Apps*	*OS X Human Interface Guidelines* *App Store Review Guidelines for Mac Apps*
To learn more about tools . . .	*Xcode Overview* *iTunes Connect Developer Guide* *iOS Simulator User Guide* *Using Application Loader*	*Xcode Overview* *iTunes Connect Developer Guide* *Using Application Loader*
To learn about tools for large teams. . .	*Xcode Server and Continuous Integration Guide* *Testing with Xcode* *Source Control Management Help*	*Xcode Server and Continuous Integration Guide* *Testing with Xcode* *Source Control Management Help*

For more information on the app review process, go to App Review.

Managing Accounts

The Accounts preferences pane is the central location for managing all of the accounts your projects will use, including your Apple ID used to manage Apple Developer Program assets. By adding an Apple ID account, joining an Apple Developer Program, and assigning your project to a team, you provide Xcode with the credentials to manage your certificates, identifiers, and profiles. You'll learn how to manage your Apple Developer Program accounts in this chapter.



About Apple Developer Program Memberships

Apple Developer Programs offer a complete set of technical resources, and support, providing everything you need to create innovative apps for iOS and Mac, extensions for Safari, and accessories for iOS devices. After you enroll in the iOS Developer Program or Mac Developer Program, you have full access to Member Center and iTunes Connect. If you enroll in the iOS Developer Enterprise Program—which allows you to distribute iOS applications to employees but not submit them to the store—you only have access to Member Center.

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're considered a one-person team, one who can perform all the tasks described in this guide except manage multiple team members.

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

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. If you enroll in the iOS Developer Enterprise Program, your team is automatically treated as a company.

To learn about the different roles and privileges, read About Apple Developer Program Team Roles and Privileges (page 144).

You Can Join Multiple Teams

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

Registered Apple Developers receive 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's verified by Apple. You 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. Using the same Apple ID, you can enroll as an individual and join other teams. However, you can only be associated with a single iTunes Connect team. Consequently, developers should create another Apple ID for different individual or company accounts that they want to manage separately in iTunes Connect.

Emails from Apple Contain Further Instructions and Welcome You

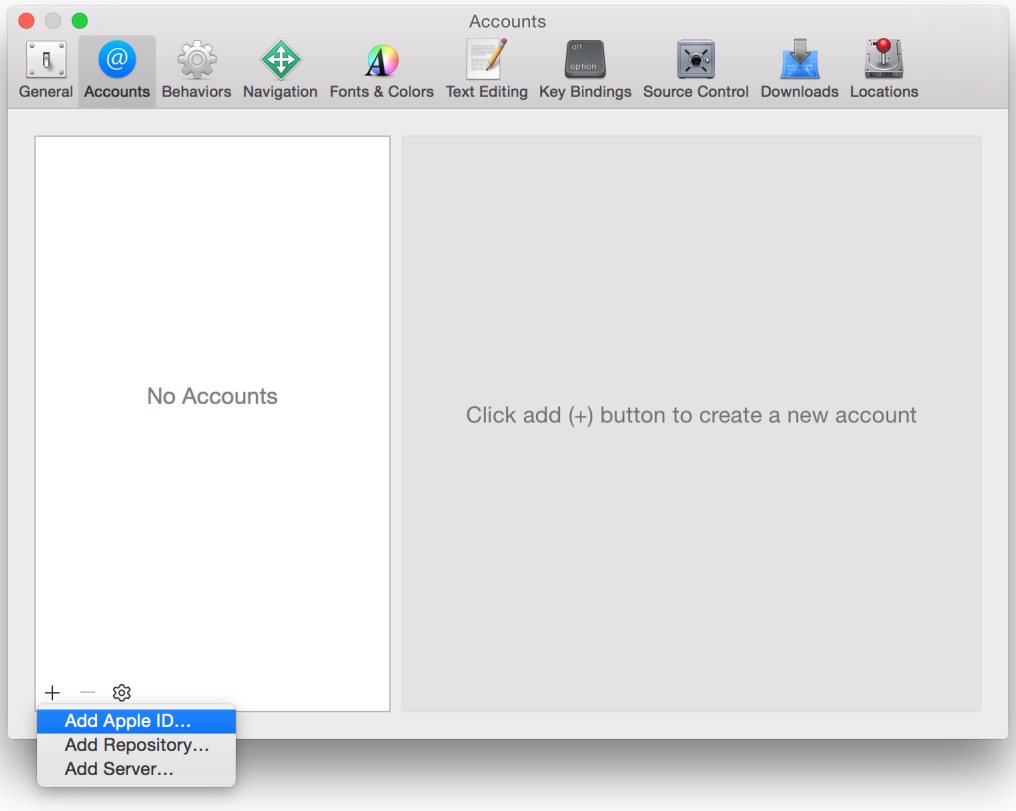
When you enroll in an Apple Developer Program or are invited to join a team, you receive a series of emails. For example, if you register as an Apple developer, Apple sends you an email requesting that you confirm your email address. Following the instructions in these emails promptly will streamline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

Adding Your Apple ID Account in Xcode

Start by adding your account using the Accounts preferences pane in Xcode. If you haven't joined an Apple Developer Program, you can join directly from Xcode. You can also add multiple Apple ID accounts.

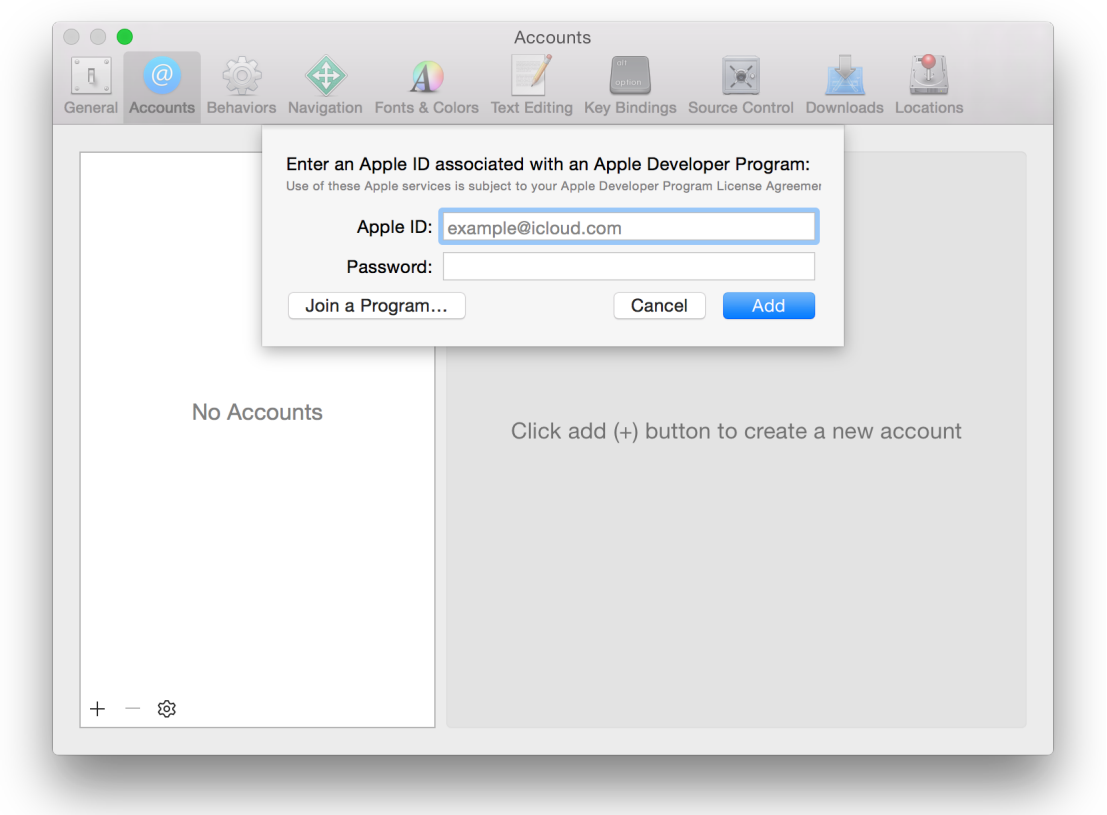
To add an Apple ID account

1. Choose Xcode > Preferences.
2. Click Accounts at the top of the window.
3. Click the Add button (+) in the lower-left corner.
4. Choose Add Apple ID from the pop-up menu.



5. If you have an Apple ID that belongs to an Apple Developer Program, enter your Apple ID and password, and click Add.

6. Otherwise, click "Join a Program" in the lower-left corner of the dialog.



Your default browser displays the Apple Developer Programs enrollment webpage. In your browser, click the developer program you want to join and follow the instructions.

To remove an Apple ID account

1. Choose Xcode > Preferences.
2. Click Accounts at the top of the window.
3. Select the Apple ID account you want to delete in the left column.
4. Click the Delete button (–) in the lower-left corner.

Adding a Developer Program to Your Team

Your team may join multiple Apple Developer Programs. For example, if you initially join the iOS Developer Program, you can add the Mac Developer Program to your team account.

To add a developer program to your team

1. Choose Xcode > Preferences.

2. Click Accounts at the top of the window.
3. Select your Apple ID in the left column.
4. Click Join in the row of the team and column of the program you want to join.

   For example, click Join under Mac to add the Mac Developer Program. Your default browser displays the developer program enrollment webpage.

5. In your browser, click “Enroll now” and follow the instructions.

To join the iOS Developer Enterprise Program, go to iOS Developer Enterprise Program and read Distributing iOS Developer Enterprise Program Applications (page 222).

Recap

In this chapter, you learned how to add your Apple ID account and join a developer program. Later, you’ll assign your Xcode project to one of your teams. Xcode uses this information to create your certificates, identifiers, and profiles for you.

Configuring Your Xcode Project for Distribution

You can edit your project settings anytime, but some settings are necessary during development. Others are recommended when you distribute your app for beta testing and required when you submit your app to the App Store or Mac App Store.

During development, your app must be provisioned and code signed to use certain app services and run on an iOS device (iPad, iPhone, or iPod touch). If you assign a bundle ID and team to your project, Xcode can create the necessary certificates, identifiers, and profiles for you in Member Center. You can enter this information now using the project editor or as needed when you add app services and launch your app.

Before you distribute your app for testing or to the store, provide all the required information about your app. For example, set app icons to pass iTunes Connect validation tests.

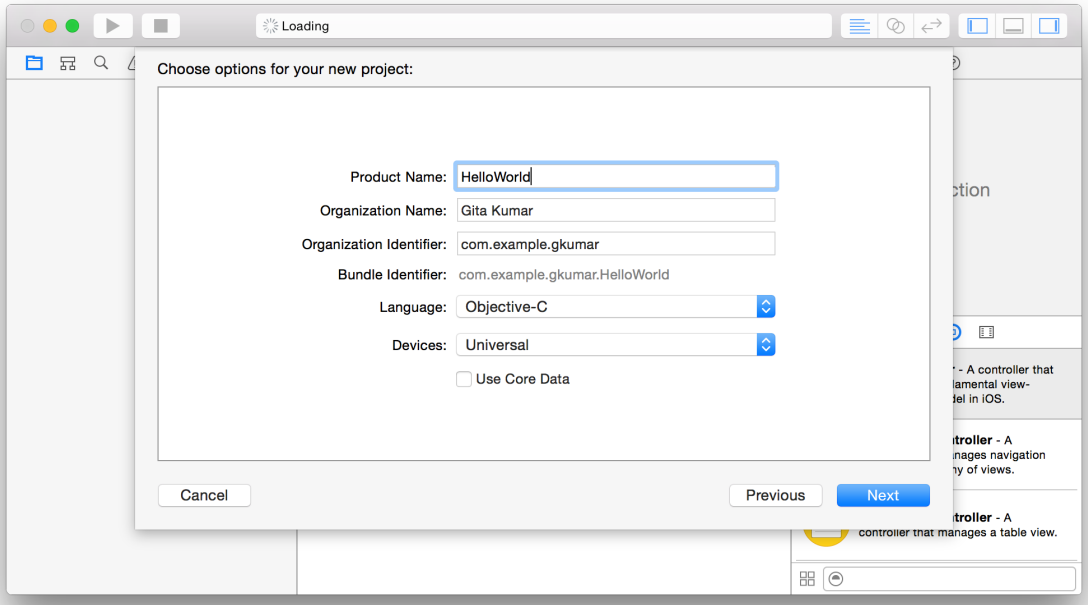
Before you upload your app to the store, verify your build settings and set the copyright key for Mac apps.

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's used to determine your app's capabilities and distribute it to customers. If you don't have this information when you create the project, you can set these properties later. If you are a member of the iOS Developer Program or Mac Developer Program, 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 upload your app to the store.

> **Note:** If you don't 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 can't perform the tasks in this book without first creating an Xcode project that builds and runs an app.

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



The *product name* is the name of your app as it will appear to customers in the store and should be similar to the app name you enter later in iTunes Connect. Most importantly, a customer should instantly associate the app name and icon in the App Store with the product name and app icon that installs on their devices. The product name is also the name that will appear in Springboard when the app is installed. The product name can't be longer than 255 bytes and can be no fewer than 2 characters. (Read Best Practices in *iTunes Connect Developer Guide* for guidelines on choosing an app name.)

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 isn't the same as the company name that you enter later in iTunes Connect.

The product name and *company identifier* you enter are concatenated to create the default bundle ID using reverse domain name service (reverse DNS) notation. The *bundle ID* needs to be unique to your app, so it's important to set the *company identifier* to a unique string as well.

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

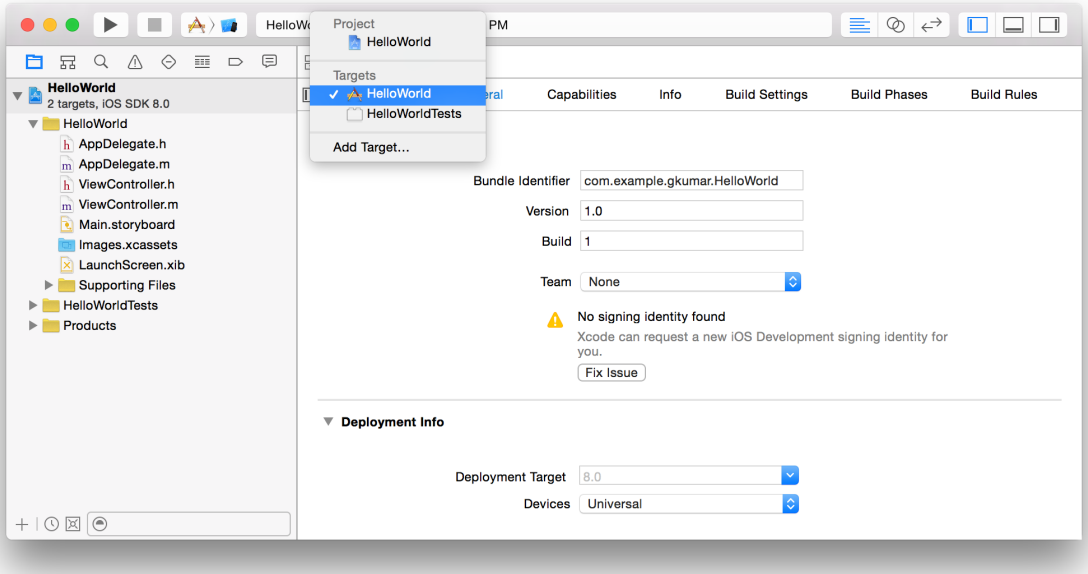
The other values used by the Xcode template are sufficient for building and running your app locally, but later you'll need to finalize properties, such as the bundle ID. Also, the assistant doesn't set all required properties for the store. You complete the basic store configuration before you submit.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

After your app is released, you can't change some of this metadata, so it's important to choose your settings carefully. To learn which app states cause some properties to be locked in iTunes Connect, refer to iTunes Connect App Properties in *iTunes Connect Developer Guide*.

Before You Begin Configuring Your Project

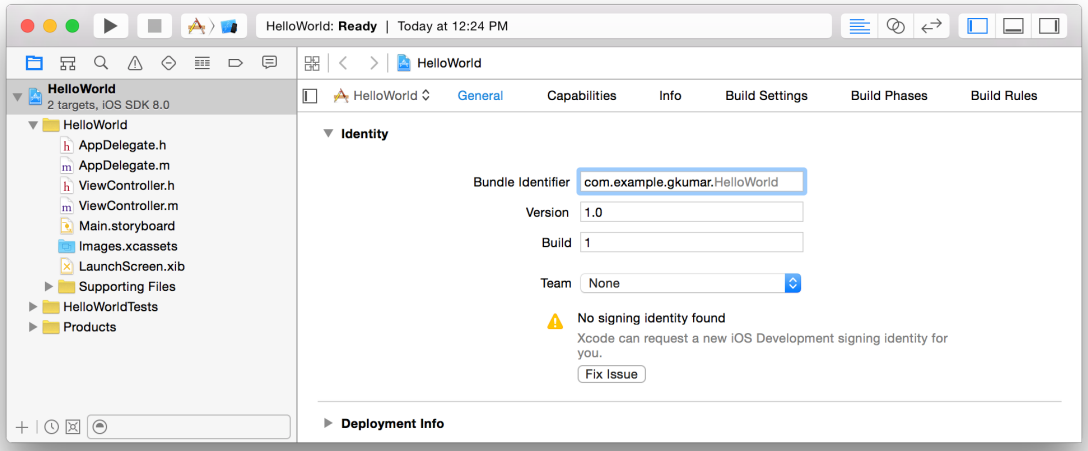
All of the options discussed in this chapter are located in the General pane in the project editor for your target. To open the project navigator, choose View > Navigators > Show Project Navigator. Choose the target from the Project/Targets pop-up menu or in the Targets section of the second sidebar if it appears. Click General to view settings discussed in this chapter.

The screenshot below shows the General pane for an iOS app.



Configuring Identity and Team Settings

For Xcode to create the team provisioning profile, the app's bundle ID needs to be unique and the project assigned to a team. Later, you provide other information that identifies this version of your app. The identity settings appear in the Identity section of the target's General pane. For iOS apps, the Identity section appears as shown here:

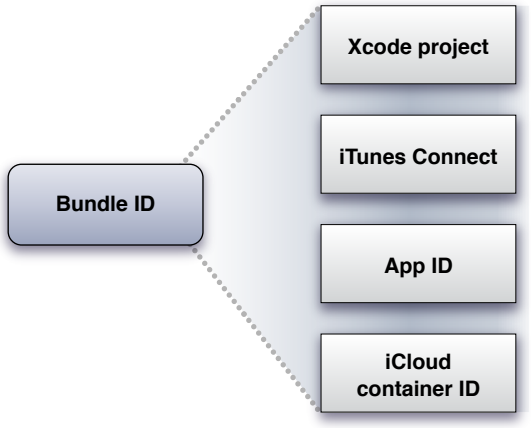


About Bundle IDs

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, Game Center and In-App Purchase use a bundle ID to identify your app when using these app services. The preferences system uses this string to identify the app for which a given preference applies. Similarly, 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 `Acme.com` and you create an app named Hello, you could assign the string `com.Acme.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.



Specifically, the bundle ID is located and used as follows:

- In the Xcode project, the bundle ID is stored in the information property list file (`Info.plist`). This file is later copied into your app's bundle when you build the project.
- In Member Center, you create an App ID that matches the app's bundle ID. If the App ID is an explicit App ID, it exactly matches the 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.
- In iCloud, the container IDs you specify in your Xcode project are based on the bundle IDs of one or more apps.
- In iTunes Connect, you enter the bundle ID to identify your app. After your first version is available on the store, you can't change your bundle ID or delete the associated explicit App ID.

> **Note:** A Mac app and an iOS app can't share the same bundle ID.

Setting the Bundle ID

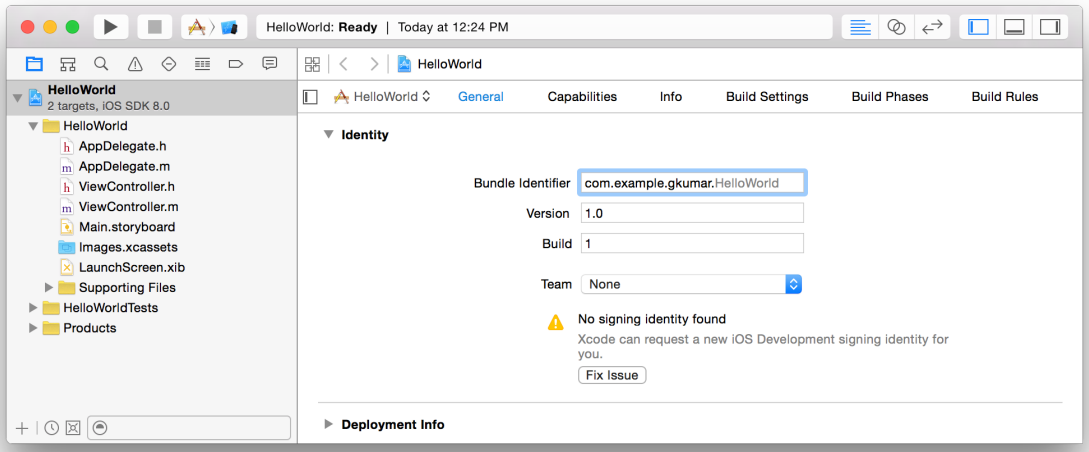
The default bundle ID in your Xcode project is a string formatted as a reverse-domain—for example, `com.MyCompany.MyProductName`. To create the default bundle ID, Xcode concatenates the company identifier with the product name you entered when creating the project from a template, as described in Setting Properties When Creating Your Xcode Project (page 23). (Xcode replaces spaces in the product name to create the default bundle ID.) It may be sufficient to replace the company identifier prefix in the bundle ID, or you can replace the entire bundle ID. For example, change the company identifier prefix to match your company domain name, or replace the entire bundle ID to match an explicit App ID.

For Mac apps, ensure that every bundle ID is unique within your app bundle. For example, if your app bundle includes a helper app, make sure that its bundle ID is different from your app's bundle ID.

Follow these steps to change the bundle ID prefix in the General pane in the project editor.

To set the bundle ID prefix

1. In the project navigator, select the project and your target to display the project editor.
2. Click General and, if necessary, click the disclosure triangle next to Identity to reveal the settings.



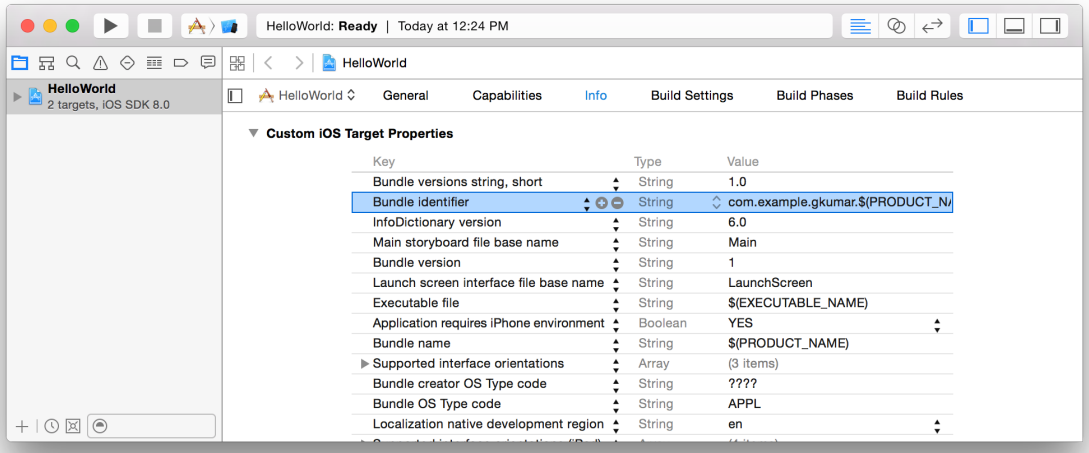
3. Enter the bundle ID prefix in the Bundle Identifier field.

Alternatively, follow these steps to change the entire bundle ID in the Info pane in the project editor.

To set the bundle ID

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

3. Enter the bundle ID in the Value column of the “Bundle identifier” row.



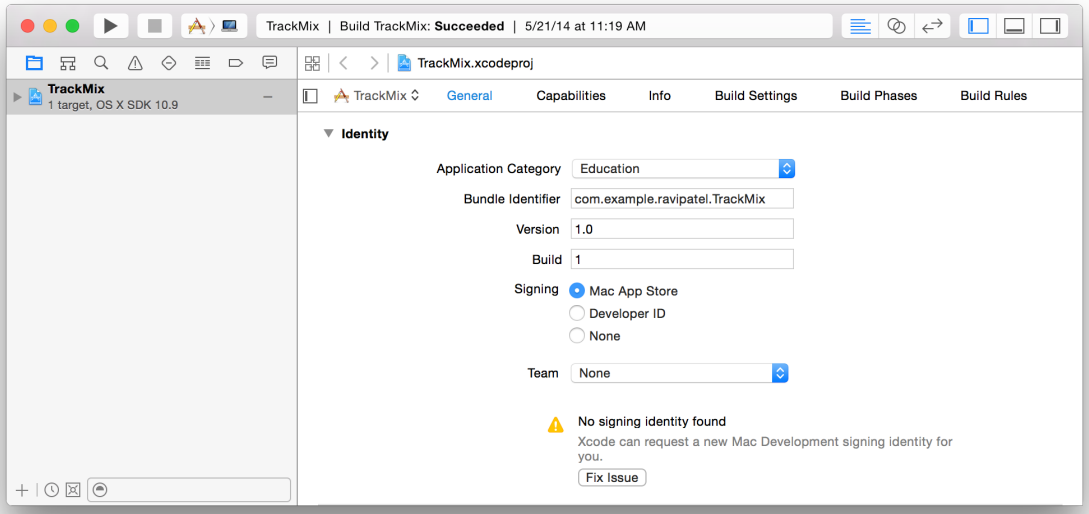
Choosing a Signing Identity (Mac Only)

You have the choice of submitting your Mac app to the store, signing it with a Developer ID certificate to distribute it outside of the store, or not code signing it at all. If you select Mac App Store, you assign your Xcode project to a team and can enable app services, as described in Adding Capabilities (page 48). If you select Developer ID, you assign your Xcode project to a team but can’t use any app services in your application (read Distributing Applications Outside the Mac App Store (page 230) for how to create a Developer ID-signed application). If you select None, the Team pop-up menu is disabled and you don’t need to read this guide.

To select a signing identity

1. In the project navigator, select the project and your target to display the project editor.
2. Click General and, if necessary, click the disclosure triangle next to Identity to reveal the settings.
3. Select the type of signing identity you want to use.
    - If you want to submit your app to the store, select Mac App Store.
    - If you want to distribute your Mac application outside of the store and not use app services, select Developer ID.

- If you don't want to sign your application or use app services, select None.



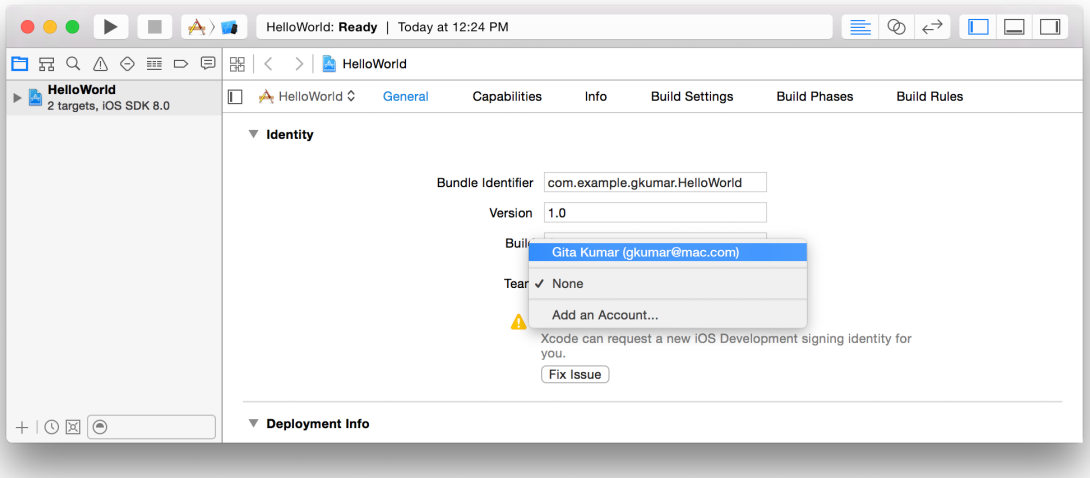
Assigning the Xcode Project to a Team

Each Xcode project is associated with a single team. If you enroll as an individual, you're considered a one-person team. The team account is used to store the certificates, identifiers, and profiles needed to provision your app. All iOS apps and some Mac apps need to be code signed and provisioned to run on a device and use certain app services. Xcode creates these assets for you when needed, but you can avoid warnings and dialogs later if you set the team now.

To assign the project to a team

1. In the project navigator, select the project and your target to display the project editor.
2. Click General and, if necessary, click the disclosure triangle next to Identity to reveal the settings.
3. Choose your team from the Team pop-up menu.
    - If you're an individual, choose your name from the pop-up menu.

- If you're a company, choose your company name from the pop-up menu.

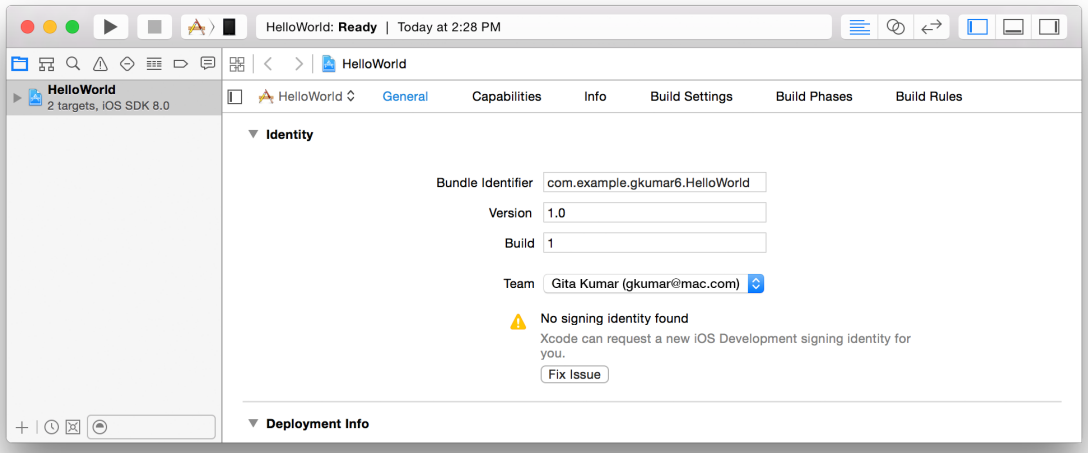


Xcode may attempt to create a team provisioning profile. For iOS apps, Xcode creates a team provisioning profile if you have an iOS device connected or have previously registered an iOS device. To resolve issues related to the team provisioning profile, read Creating the Team Provisioning Profile (page 32).

4. If a team doesn't appear in the Team pop-up menu, choose "Add an Account" and follow the steps described in Adding Your Apple ID Account in Xcode (page 20).

Creating the Team Provisioning Profile

After you select a team, Xcode attempts to create your code signing identity and a development provisioning profile. Xcode creates a specialized development provisioning profile called a *team provisioning profile* that it manages for you. A team provisioning profile allows an app to be signed and run by all team members on all their devices. If Xcode fails to create the team provisioning profile, a warning and Fix Issue button appear below the Team pop-up menu.



To create the team provisioning profile, Xcode performs these steps:

1. Requests your development certificate
2. Registers the iOS device chosen in the Scheme toolbar menu or your Mac
3. Creates an App ID that matches your app's bundle ID and enables app services
4. Creates a team provisioning profile containing these assets
5. Sets your project's code signing build settings accordingly

Important: If you are a team member for a company account, ask your team agent or admin to register your Mac on your behalf, as described in Registering Team Member Devices (page 150). Also, ask the team agent or admin to follow these steps to create the team provisioning profile if it doesn't already exist. To download the team provisioning profile to your Mac, refresh the provisioning profiles, as described in Refreshing Provisioning Profiles in Xcode (page 205), or click the Fix Issue button under the Team pop-up menu.

You can assist Xcode and avoid common problems by creating the team provisioning profile now.

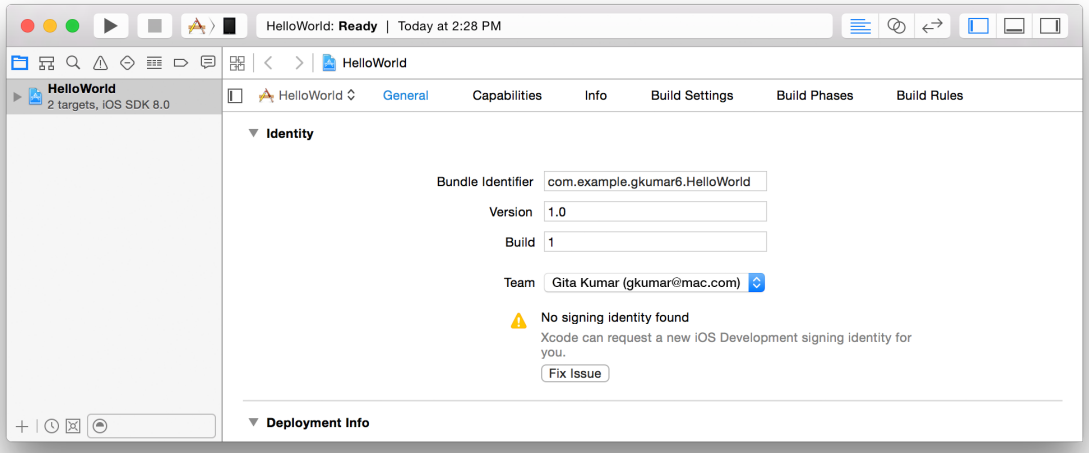
To create a team provisioning profile

1. In the project navigator, select the project and your target to display the project editor.
2. Click General and, if necessary, click the disclosure triangle next to Identity to reveal the settings.
3. Verify that your bundle ID is unique by using a unique prefix.

   Depending on your project's configuration, Xcode may create either a wildcard App ID or an explicit App ID. Because Xcode uses the bundle ID to register an explicit App ID, the bundle ID also needs to be unique during development. To avoid an App ID registration error, enter a unique bundle ID now, as described in Setting the Bundle ID (page 27).
4. For iOS apps, connect an iOS device you want to use for development and choose the device from the Scheme toolbar menu.

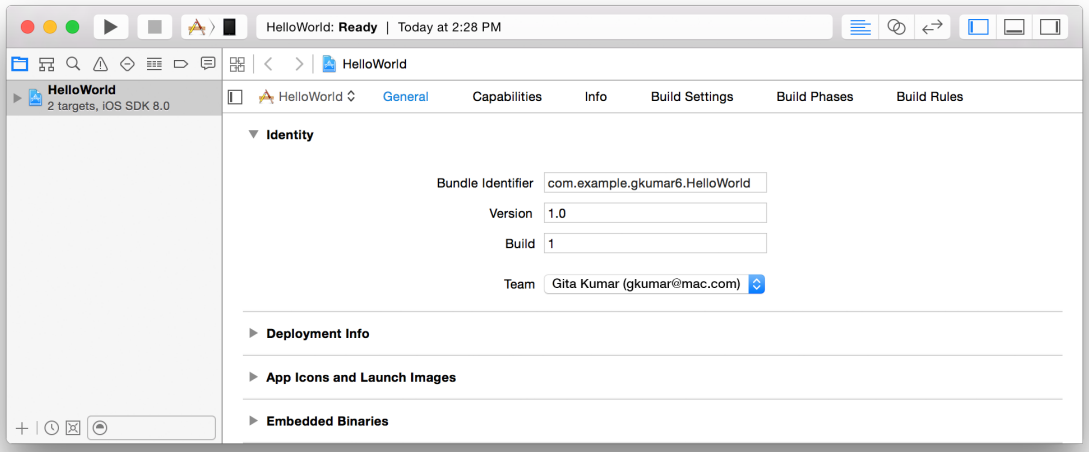
   Xcode requires one or more registered devices in your account before it can create a team provisioning profile. (For Mac apps, Xcode automatically registers the Mac that's running Xcode.) If you connect an iOS device to your Mac that matches the deployment target, Xcode registers the device for you. If the iOS device isn't eligible because it doesn't match the deployment target, upgrade the iOS on the device or change the deployment target, as described in Setting Deployment Info (page 36).
5. If necessary, choose your team from the Team pop-up menu.

   For Mac apps, Mac App Store must be selected as the type of signing identity before you can choose a team.
6. If a Fix Issue button appears below the Team pop-up menu, click it.



   Xcode starts performing all the steps needed to create a team provisioning profile for your app. If no warning message appears below the Team pop-up menu, Xcode already created your team provisioning profile.
7. If dialogs or warnings appear, use the information to correct the problem and click Fix Issue again.

After Xcode creates the team provisioning profile, the warning message under the Team pop-up menu disappears.

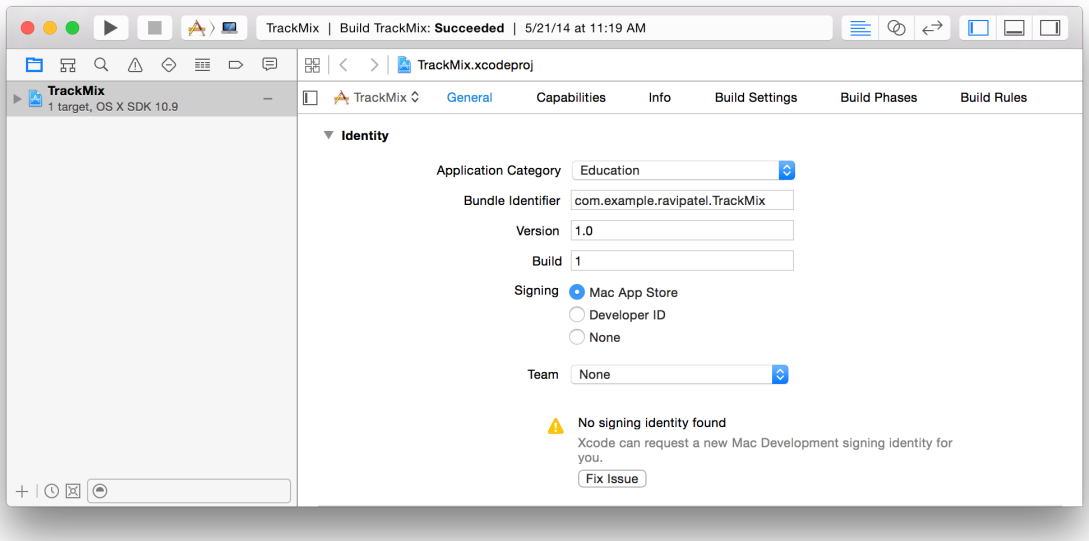


Setting the Application Category (Mac Only)

Set the category under which your Mac 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 General and, if necessary, click the disclosure triangle next to Identity to reveal the settings.
3. Choose the category from the Application Category pop-up menu.



iOS app categories are set in iTunes Connect only. For more details on app categories, read Best Practices in *iTunes Connect Developer Guide*.

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. For details on possible values, see CFBundleShortVersionString in *Information Property List Key Reference*.

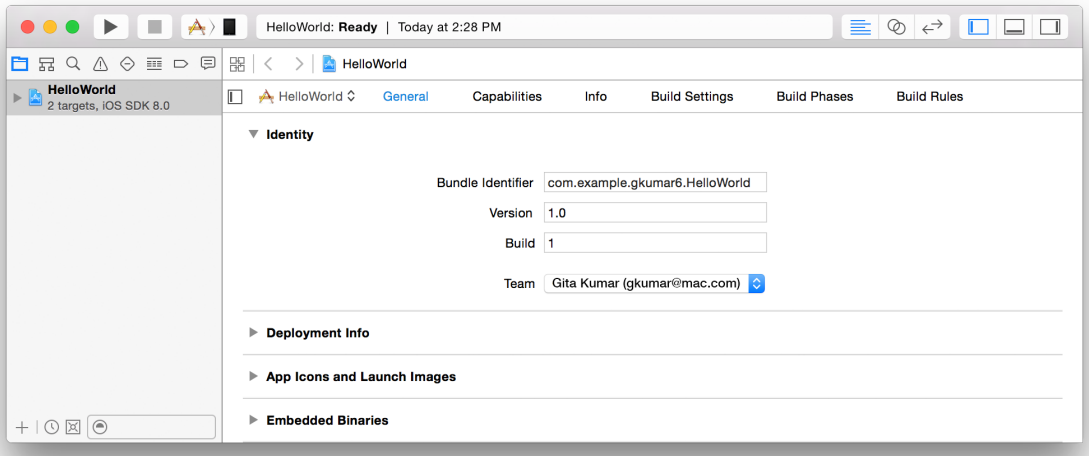
The build string represents an iteration (released or unreleased) of the bundle and is also a two-period-separated list of positive integers, as in `1.2.3`. For Mac apps, the user can click the version number in the About window to toggle between the version number and the build string. For details on possible values, see CFBundleVersion in *Information Property List Key Reference*.

For iOS apps, 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 to the device. For how to configure your app for testing, read Beta Testing iOS Apps (page 97).

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

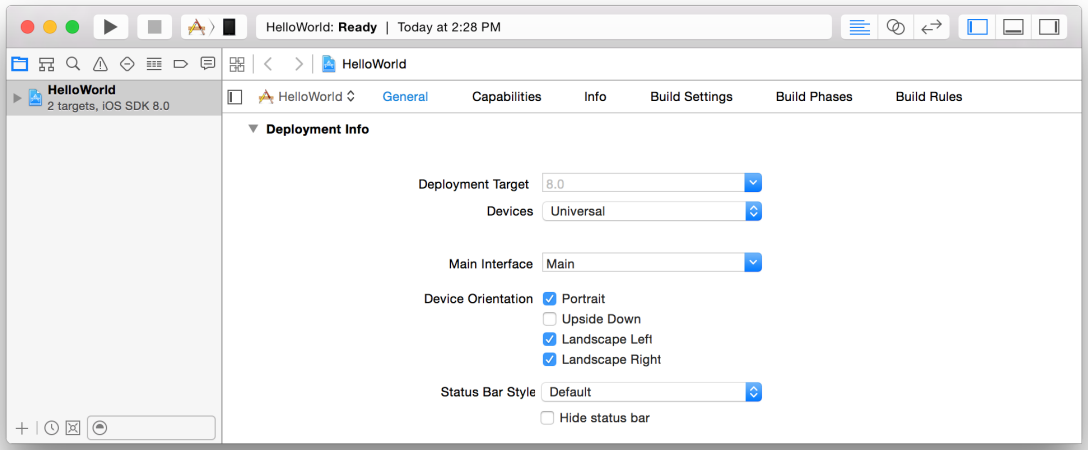
To set the version number and build string

1. In the project navigator, select the project and your target to display the project editor.
2. Click General and, if necessary, click the disclosure triangle next to Identity to reveal the settings.
3. Enter the version number in the Version field, and enter the build string in the Build field.



Setting Deployment Info

The default deployment settings are sufficient for development, but it's best to review these settings before you distribute your app. Some settings must match values you enter in iTunes Connect later. For iOS apps, the deployment settings appear as shown here:



Setting the Deployment Target

The deployment target setting specifies the lowest operating system version that your app can run on. For example, the lowest available setting for iPad apps is iOS 4.3.

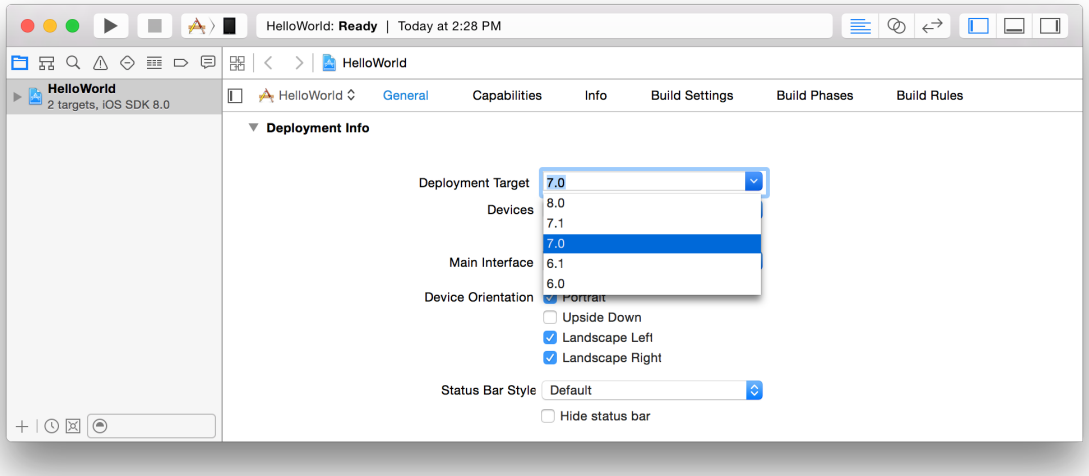
There are several strategies for choosing the deployment target when developing your app. Each version of iOS or OS X includes features and capabilities not present in earlier versions. As new versions are published, some users may upgrade immediately, while other users may wait before moving to the 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 General and, if necessary, click the disclosure triangle next to Deployment Info to reveal the settings.

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.
>
> 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*.

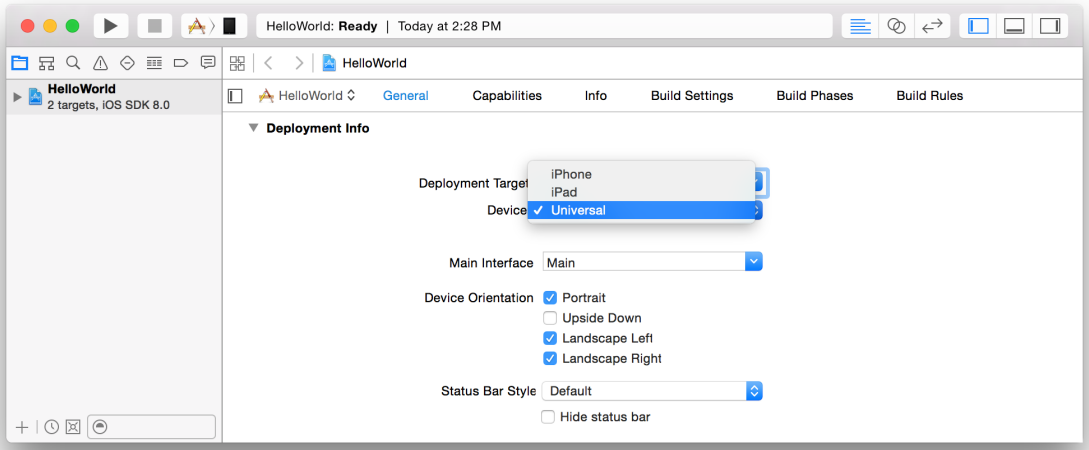
Setting the Target Devices (iOS Only)

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 General and, if necessary, click the disclosure triangle next to Deployment Info to reveal the settings.

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 *App Programming Guide for iOS*.

Adding App Icons and a Launch Screen File

App icons and the launch screen file are stored in the app bundle, not uploaded as separate assets to iTunes Connect. The operating system uses these images and the launch screen file (a `.xib` file) 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. Your app needs an app icon to represent it and pass validation tests.

You use an asset catalog that manages the app icons for you or maintain individual image files yourself. If you create a new project, asset catalogs are used by default to store app icons and a launch screen file is used as a splash screen. For iOS 7 deployment targets, you can supply both a launch screen file and launch images. In iOS 8, the launch screen file is used, and in iOS 7, the launch images are used. If you have an older project, you can migrate from managing individual icon image files to an asset catalog. You can also add a launch screen file to an older Xcode project.

If you prefer to manage your assets as individual image files, read Setting Individual App Icon and Launch Image Files (page 42). If you want to migrate to an asset catalog, read Migrating Your Images to an Asset Catalog (page 42). To learn more about creating and managing asset catalogs, read *Asset Catalog Help*.

Preparing Your Artwork

For all artwork, keep the file size as small as possible for a positive purchase experience for your users.

For iOS apps, see Icon and Image Sizes in *iOS Human Interface Guidelines* for the sizes of all required app icons, launch images, and other icons.

In iOS 7, 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. To take advantage of Retina displays, provide high-resolution images for each device you support. For guidance on creating launch images, read Launch Images.

For the required Mac app icons, see Creating Great Icons for Any Resolution in *OS X Human Interface Guidelines*. This table includes icon sizes that may be used on the Mac App Store. To create your app icon files, read Provide High-Resolution Versions of All App Graphics Resources in *High Resolution Guidelines for OS X*.

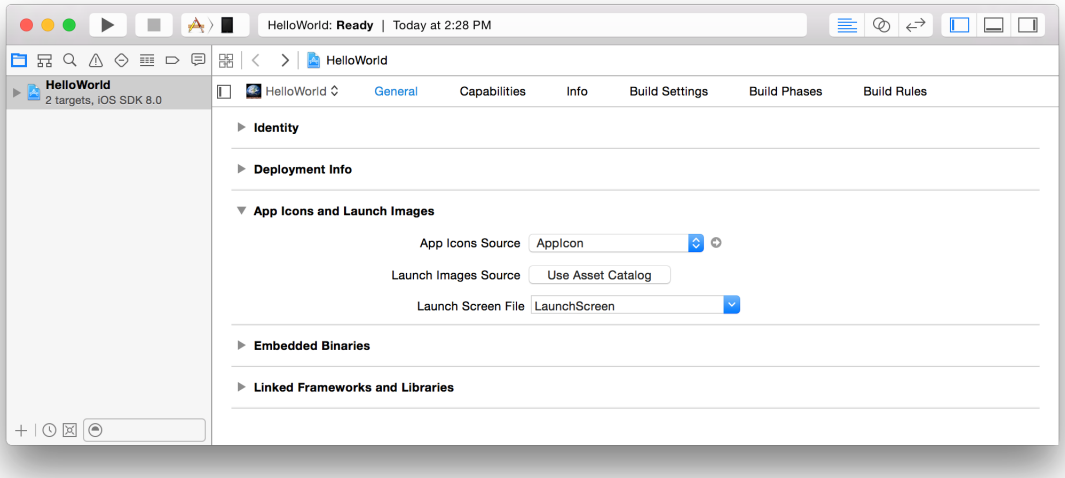
For the specification of screenshots and other artwork that you upload later using iTunes Connect, read iTunes Connect App Properties in *iTunes Connect Developer Guide*.

Adding App Icons to an Asset Catalog

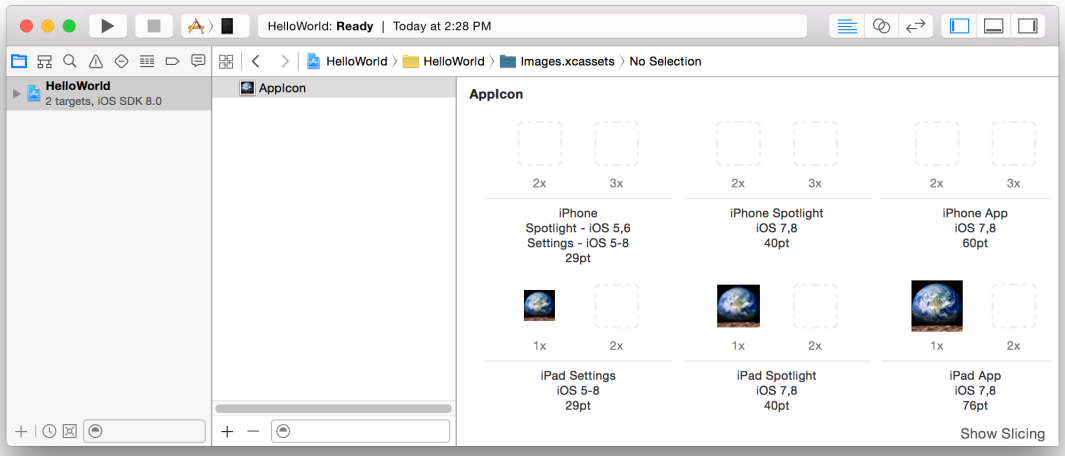
Versions of your app icons are organized into image sets in an asset catalog. Xcode automatically creates image sets for the app's target device—for example, both iPhone and iPad image sets appear if your iOS app's target is universal.

To add an app icon to an asset catalog

1. In the project navigator, select the project and your target to display the project editor.
2. In the "App Icons and Launch Images" section of the General pane, click the arrow button to the right of the App Icons Source pop-up menu.



3. In the Finder, drag an app icon to the image well that matches its resolution in the project navigator.



Creating a Launch Screen File

The launch screen file is displayed as a splash screen while your app is launching. It's a single, atomic `.xib` file that uses size classes to support different device resolutions. It contains basic UIKit views, such as UIImageView and UILabel objects, and uses Auto Layout constraints. Xcode adds a default launch screen file, called `LaunchScreen.xib`, to your project.

Follow these guidelines when creating a launch screen file:

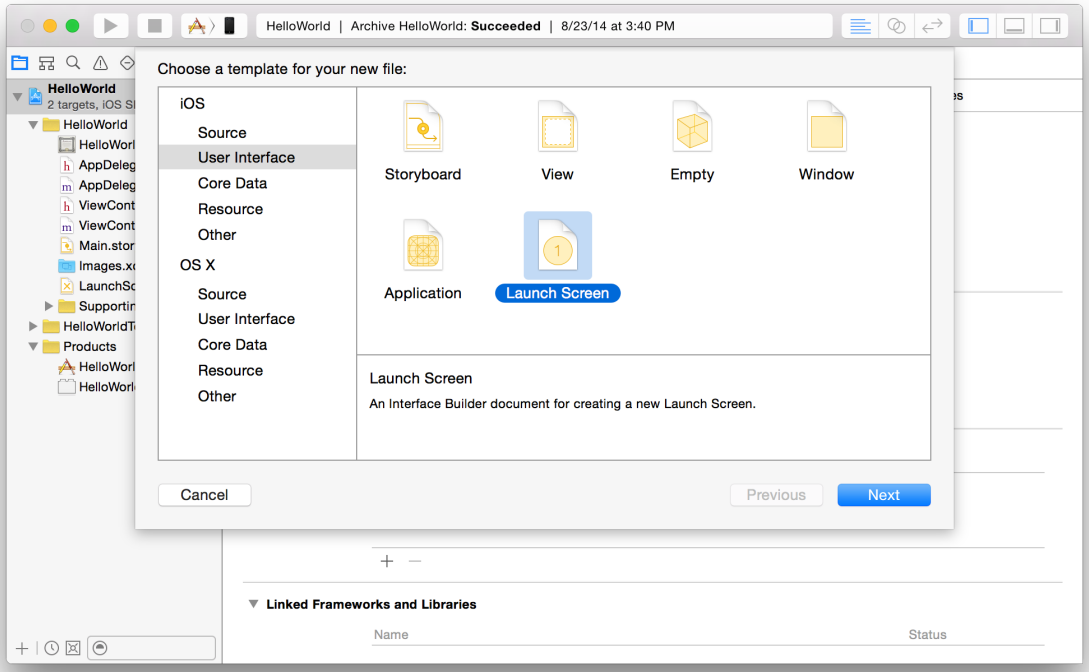
- Use only UIKit classes.
- Use a single root view that is a UIView or UIViewController object.
- Don't make any connections to your code (don't add actions or outlets).
- Don't add UIWebView objects.
- Don't use any custom classes.
- Don't use runtime attributes.

You can add a launch screen file to an older Xcode project.

To create a launch screen file for an existing project

1. Choose File > New > File.
2. Under iOS, select User Interface.

3. Click Launch Screen and click Next.



4. Enter a filename in the Save As text field, and click Create.

To set the launch screen file

1. If necessary, open the "App Icons and Launch Images" section of the General pane.
2. From the Launch Screen File pop-up menu, choose a launch screen file.

Adding Launch Images and Capturing Screenshots

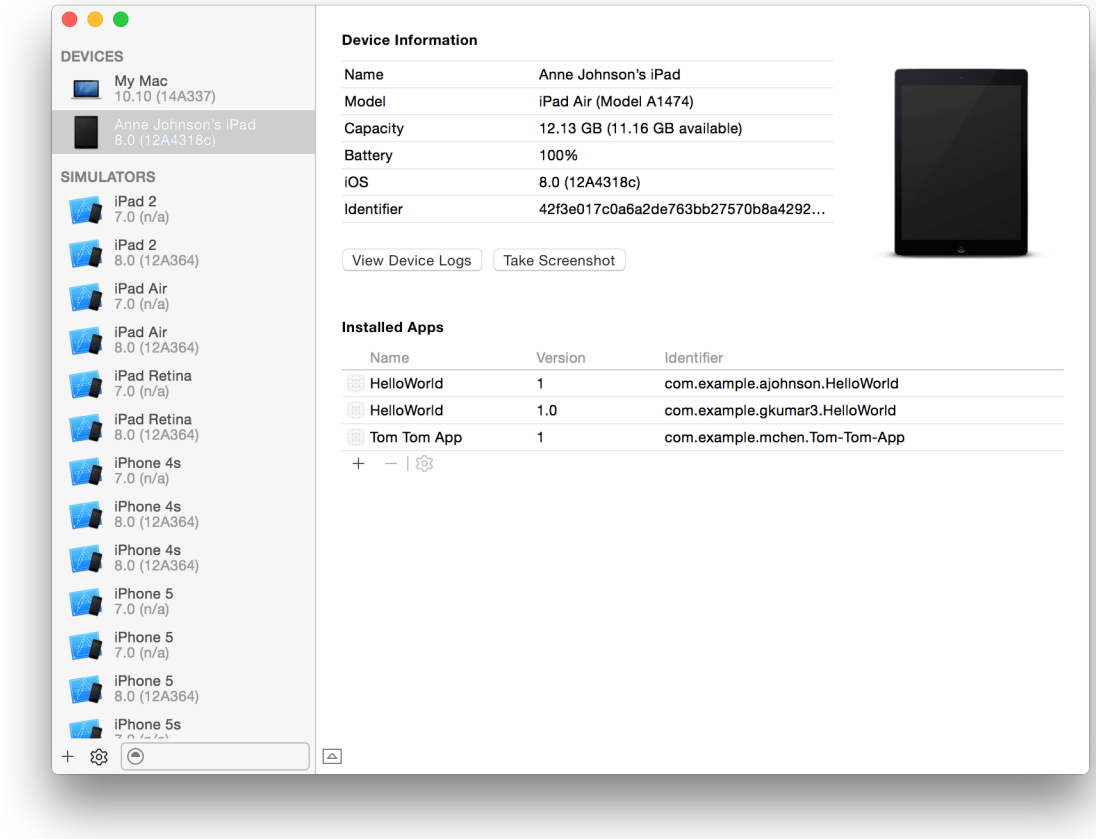
For iOS 7 deployment targets, add an asset catalog for launch images, described in Migrating Your Images to an Asset Catalog (page 42). Capture screenshots to create the launch images and add them to the asset catalog, similar to adding app icons to an asset catalog, described in Adding App Icons to an Asset Catalog (page 39).

Follow these steps to capture a screenshot of your iOS app while your device is connected to your Mac. Although the screenshot includes the status bar as it looks when the screenshot is captured, iOS replaces it with the current status bar when your app launches.

To capture a screenshot on your iOS app

1. Connect the iOS device to your Mac.
2. Choose Window > Devices, and select the device under Devices.

3. In the Device Information section, click Take Screenshot.



The screenshot appears on your desktop.

Setting Individual App Icon and Launch Image Files

If you don't want to use an asset catalog, choose "Don't use asset catalogs" from either the App Icons Source or Launch Images Source pop-up menu in the "App Icons and Launch Images" section of the General pane. If a Use Asset Catalog button appears, you are not using an asset catalog.

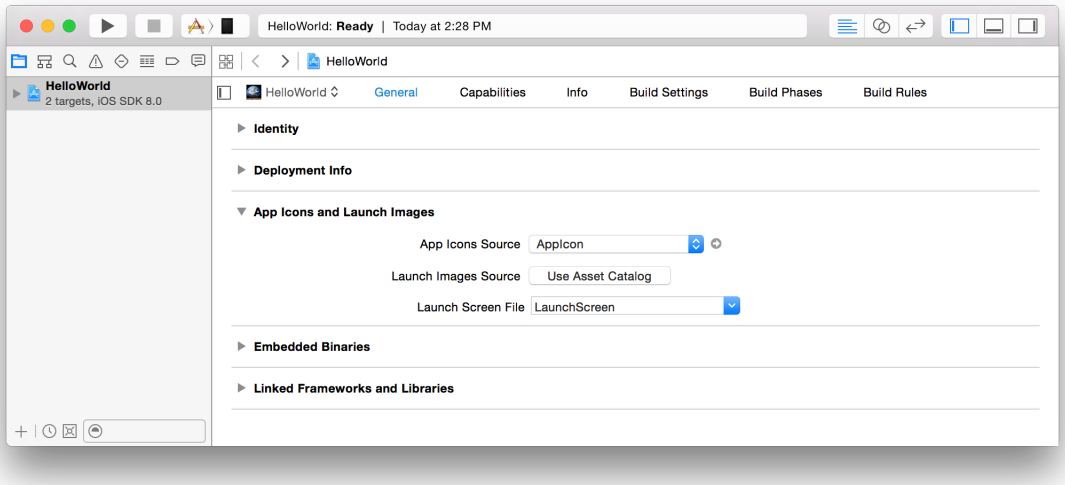
Migrating Your Images to an Asset Catalog

If you have an older project, you can migrate your image files to an asset catalog that manages your app icons and launch images for you. Xcode moves the image files from the tables to the new asset catalog. You create a separate asset catalog for app icons and launch images, but the steps are identical.

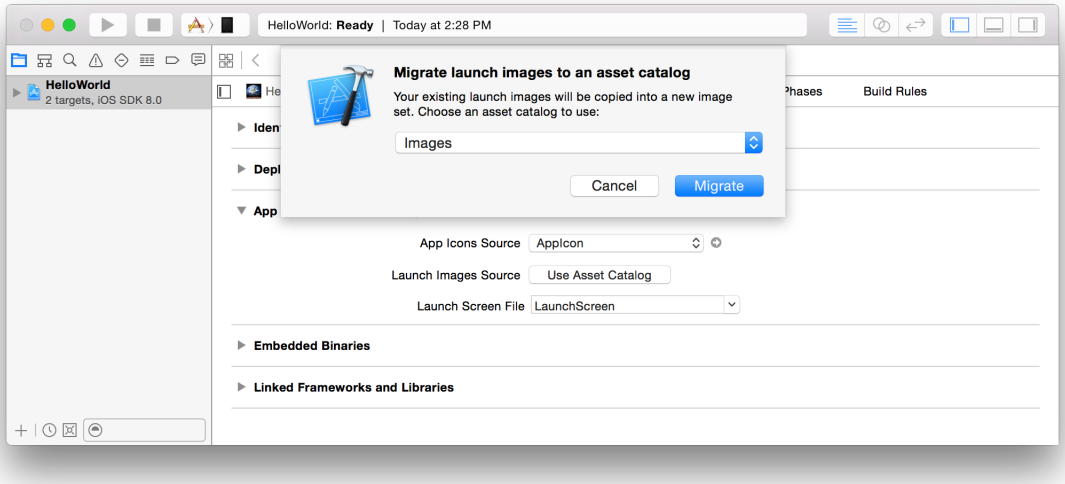
To migrate to an asset catalog

1. In the project navigator, select the project and your target to display the project editor.
2. Click General and, if necessary, click the disclosure triangle next to "App Icons and Launch Images."

3. Click the Use Asset Catalog button next to either App Icons Source or Launch Images Source.



4. In the dialog that appears, choose either Images or New Asset Catalog from the pop-up menu, and click Migrate.



To view the asset catalog, click the arrow button to the right of the App Icons Source or Launch Images Source pop-up menu.

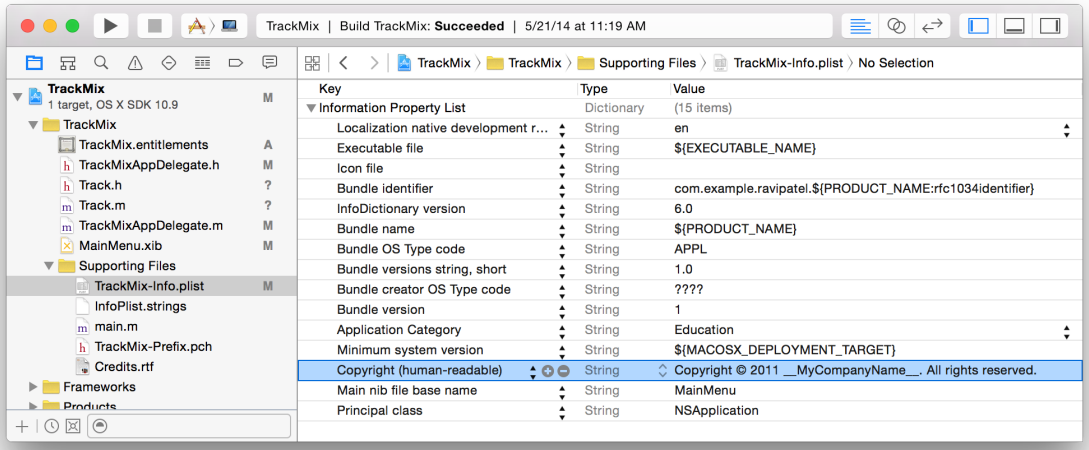
Setting the Copyright Key (Mac Only)

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*.

To edit the copyright key in 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 Copyright key.
6. Enter a new value for the key.

For how to edit other cells in a property list, refer to *Property List Editor Help*.

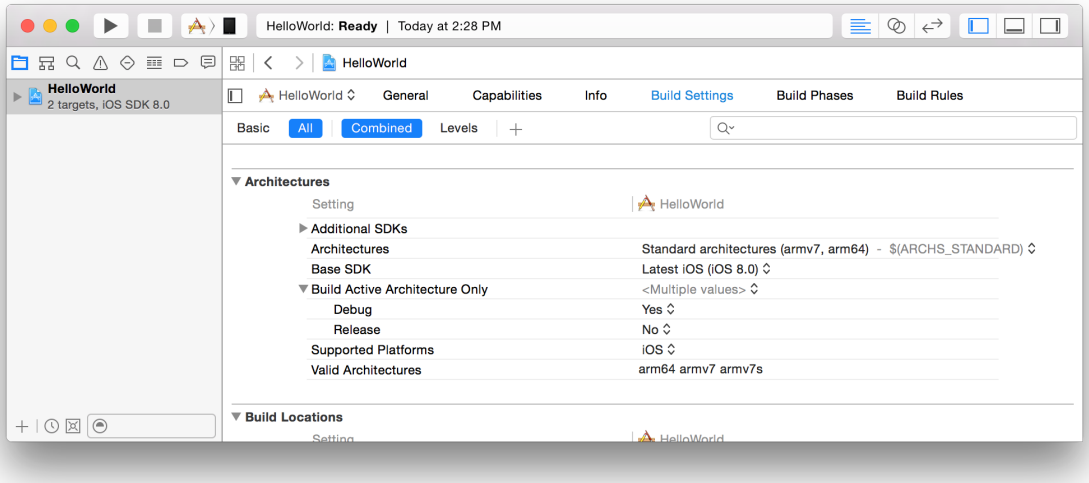
Verifying Your Build Settings

If you changed the default build settings, verify some of the settings before submitting your app to the store. 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. At the top of the project editor, click Build Settings.



3. Locate the build setting in the left column, or enter the name of the build setting in the search field in the upper-right corner.
4. If some build settings don't appear, click All.
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 a set of architectures, which include `armv7` and `arm64`. 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.

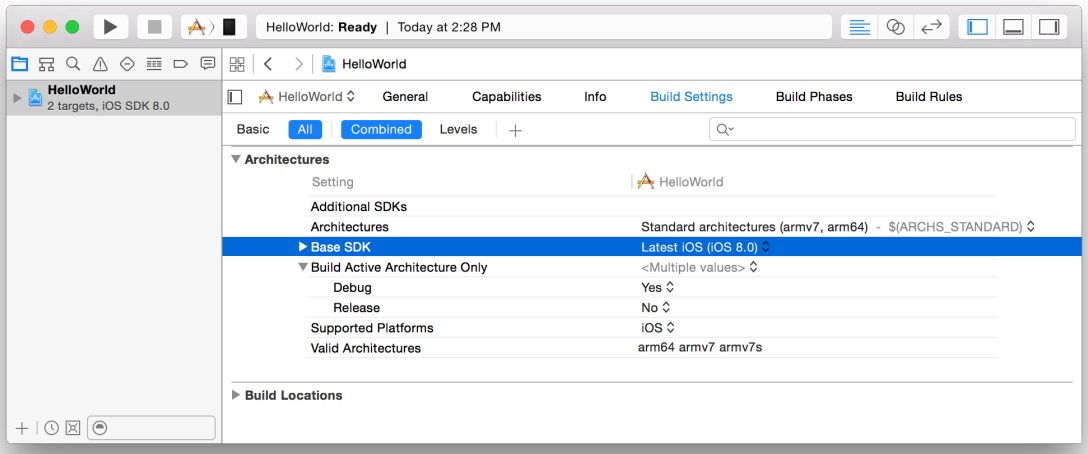
If you select Other from the Architectures build-setting value list, click the Add button (+) to enter the custom iOS-device architecture names you support.

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

Setting the Base SDK

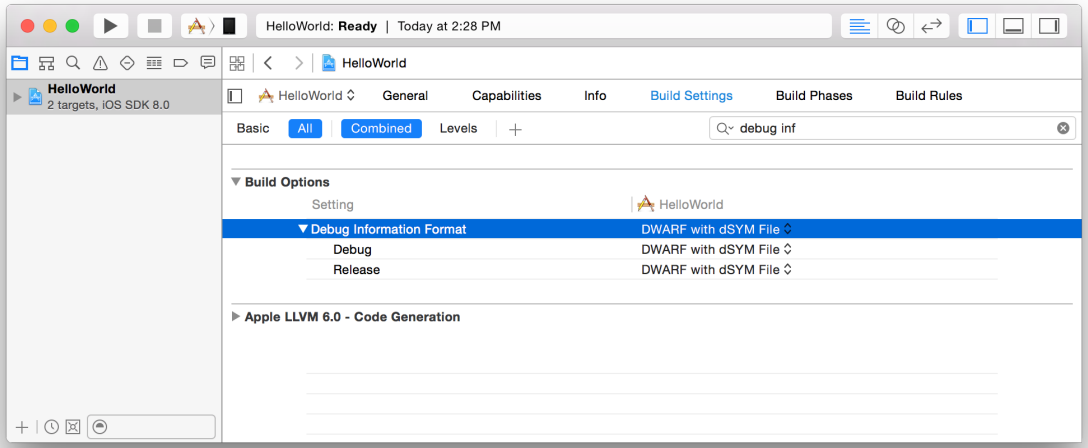
The Base SDK version number must be greater than or equal to the software version number on your development device; otherwise, Xcode can't initiate a debugging session with the device. Set the Base SDK for your project and targets to the latest operating system, which is the default value. The Base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

To go to the Architectures area, select the project or target and click Build Settings. The Architectures area appears first in the Build Settings pane.



Setting the Debug Information Format

Set the Debug Information Format build setting to “DWARF with dSYM File.” This is required to symbolicate crash reports, as described in Analyzing Crash Reports (page 114).



Recap

In this chapter, you learned how to configure your Xcode project from a template, set the app’s identity information, and create a team provisioning profile for development. During development, refer to this chapter as needed. Later, use this chapter as a checklist for the settings that are required by the App Store and Mac App Store.

Adding Capabilities

Certain app services—such as Game Center and In-App Purchase—are available only to iOS Developer Program and Mac Developer Program apps distributed through the store. These services require additional configuration in your Xcode project, Member Center, and sometimes iTunes Connect. App services that don't require iTunes Connect configuration are also available to iOS Developer Enterprise Program apps. Some app services are for certain types of apps, such as games and Newsstand apps, and provide additional sources of revenue, such as In-App Purchase and iAd Network.

Apple implements an underlying security model to protect both user data and your app from being modified and distributed without your knowledge. Hence, your app is code signed and provisioned to use only the app services that you specify. When you add capabilities to your app using Xcode, Xcode automatically configures your project to use them. Xcode edits the entitlements and information property list files for you and adds technology-specific frameworks as needed. For entitlements to take effect, Xcode creates code signing and provisioning assets for your team and sets your code signing build settings for you. Xcode creates a wildcard App ID and explicit App ID, if needed, to enable the app services you choose. Some app services—such as Game Center and In-App Purchase—require additional setup in Member Center and iTunes Connect.

This chapter describes all the steps that you perform to access app services from your app.

About Entitlements

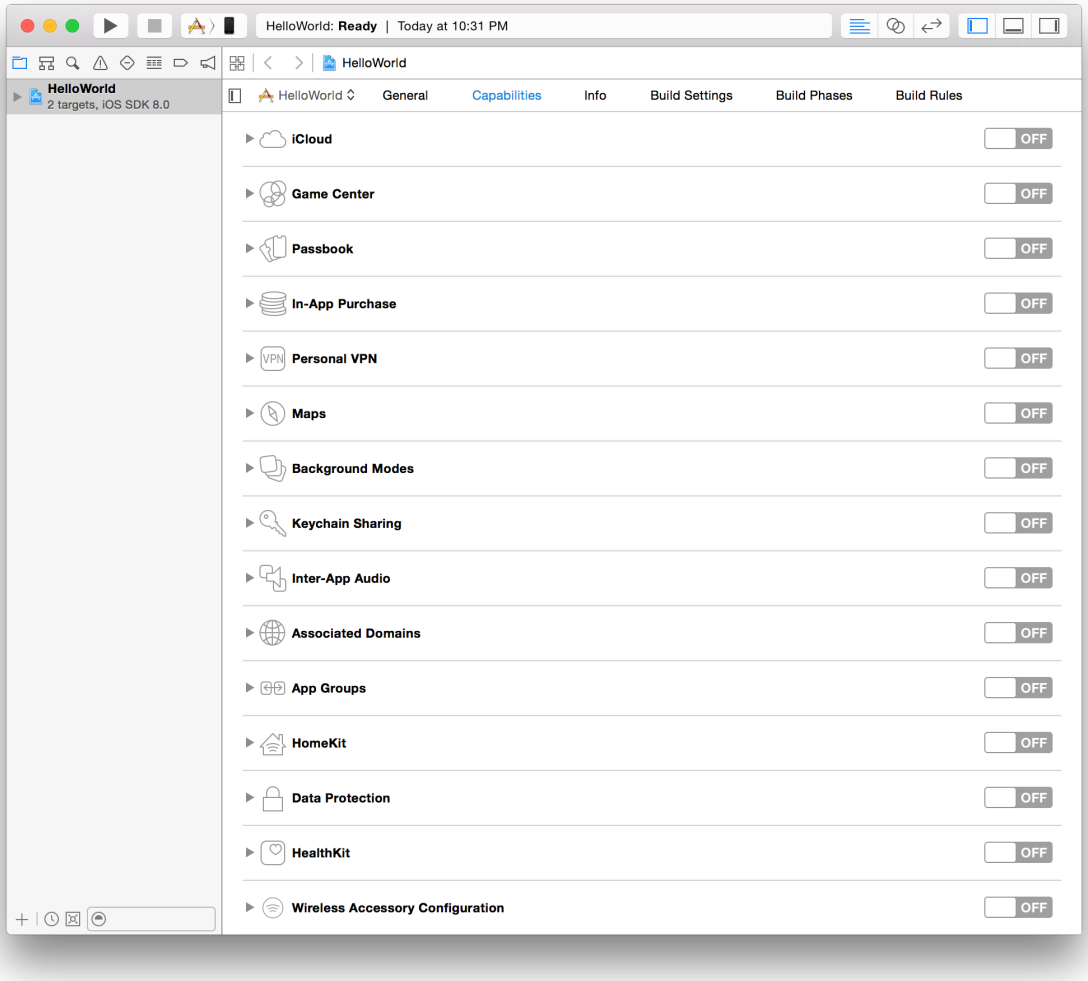
An **entitlement** is a single right granted to a particular app, tool, or other executable that gives it additional permissions above and beyond what it would ordinarily have. The term *entitlement* is most commonly used in the context of a sandbox, and to a lesser degree for an App ID. Regardless of 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.

By enabling app services in Xcode, you set some entitlements in the Xcode project and for an App ID in Member Center. The app services enabled for an App ID serve as a whitelist of the services one or more apps may use. Some app services are enabled by default for an explicit App ID that exactly matches the bundle ID. The Xcode project configuration specifies which services the app actually uses.

Before You Begin

All of the options discussed in this chapter are located in the Capabilities pane in the project editor for your target.

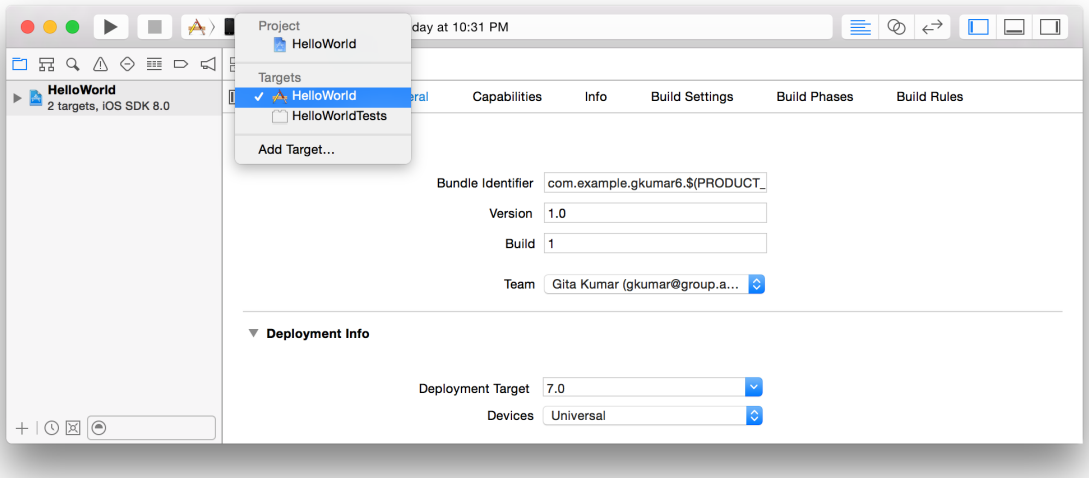
The screenshot below shows the capabilities for an iOS app. A subset of these app services is available to Mac apps.



To open the Capabilities pane

1. Choose View > Navigators > Show Project Navigator.

2. Choose the target from the Project/Targets pop-up menu or in the Targets section of the second sidebar if it appears.



3. Click Capabilities to view app services you can add to your app.

Xcode creates code signing and provisioning assets for you as you need them. However, because some assets depend on others, dialogs may appear asking you to fix problems while you enable capabilities. For example, you may be asked to assign a team to your project, create a development certificate, and for iOS apps, connect an iOS device so that Xcode can create your team provisioning profile. A development provisioning profile is not required to enable capabilities but is required to build and launch an app that uses the capabilities. To avoid these dialogs and warnings, create your code signing identity and team provisioning profile now, as described in Creating the Team Provisioning Profile (page 32). Otherwise, read Troubleshooting (page 75) for how to resolve issues as they occur.

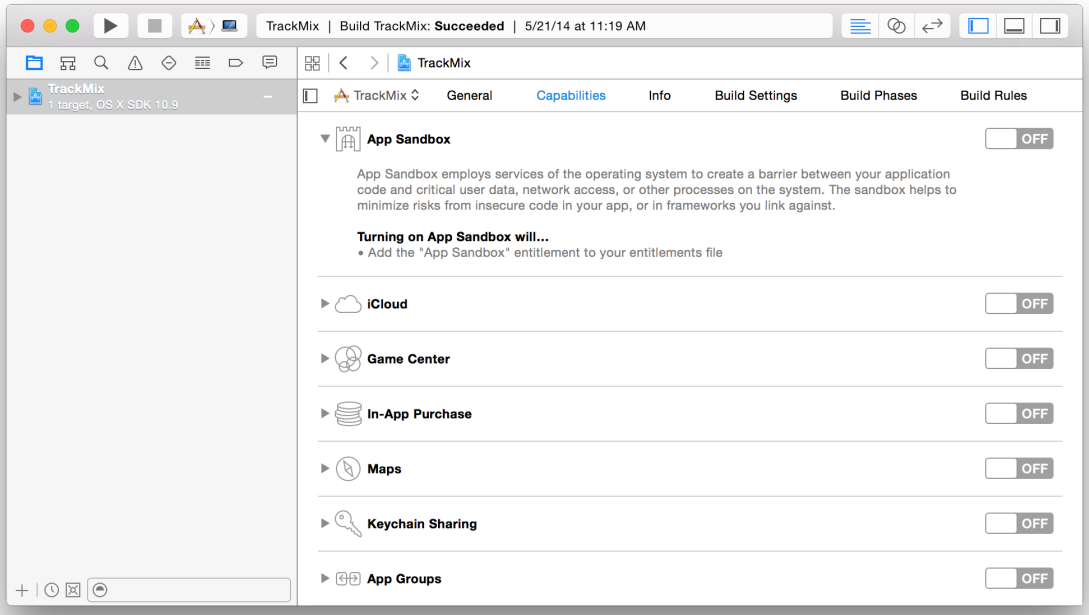
Configuring App Sandbox (Mac Only)

Sandboxing provides the last line of defense against stolen, corrupted, or deleted user data if malicious code exploits your Mac app. Sandboxing 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 distributed by the Mac App Store are required to use sandboxing. Therefore, if you upload your app to iTunes Connect, enable sandboxing during development.

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

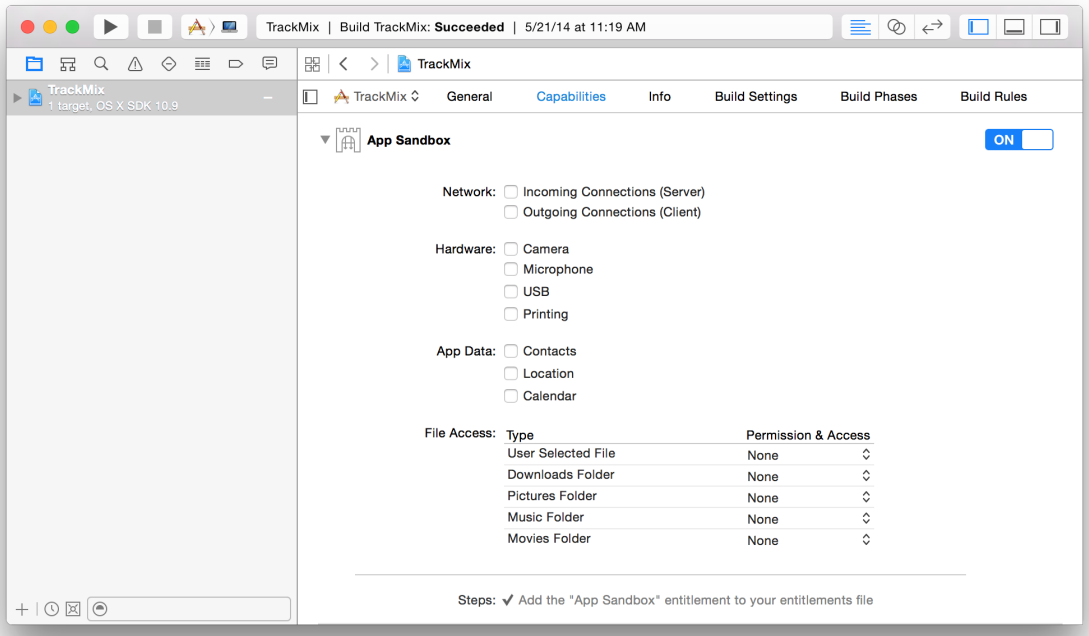
To configure App Sandbox

1. In the Capabilities pane, if App Sandbox isn't enabled, click the switch in the App Sandbox section.

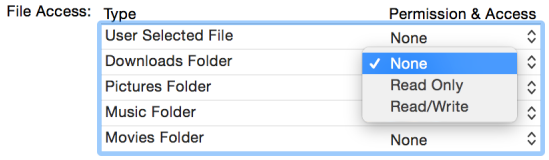


Xcode adds an entitlements file to your project and automatically enters default values for some entitlements. Xcode also enables the App Sandbox entitlement.

2. Use the App Sandbox checkboxes in this area to describe the minimum set of capabilities the target needs to do its job.



You can set specific permissions for file types, too. To set the access for a file type, choose a permission from the pop-up menu in the row that best describes the file type.



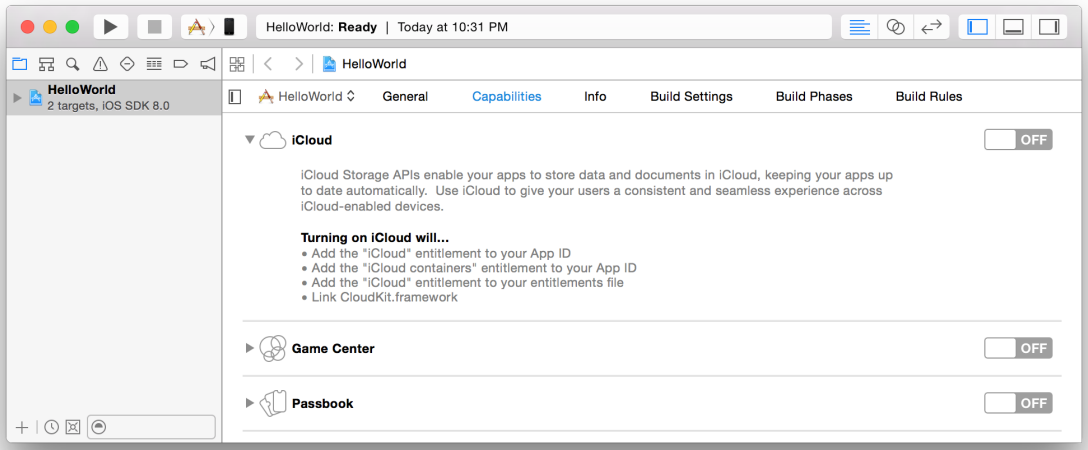
For a complete description of App Sandbox entitlements, refer to *Entitlement Key Reference*. If you're enabling sandboxing for an existing app, read *App Sandbox Design Guide* to learn the locations that a sandboxed app can access.

Adding iCloud Support

iCloud storage allows you to share user or app data among multiple instances of your app running on different iOS devices and Mac computers. You can also share data between different apps developed by your team. You choose which iCloud services—key-value storage, document storage, or CloudKit—to use depending on how you want to store and retrieve data. For document storage and CloudKit, you can specify the containers your app will use and create custom containers shared by multiple apps. Your app needs to be provisioned to use iCloud, which includes creating an explicit App ID if it doesn't already exist and setting service-specific entitlements in your Xcode project.

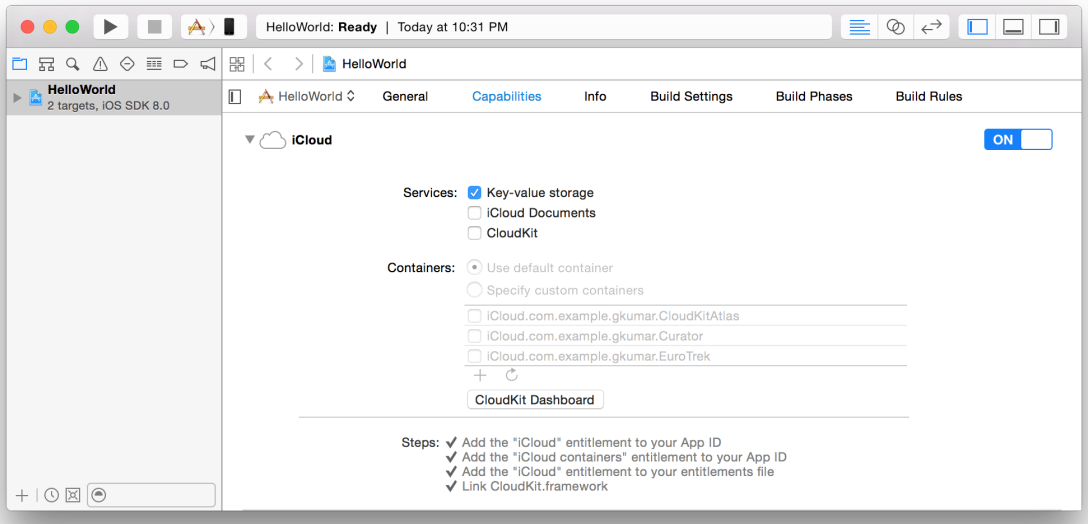
Enabling iCloud

Before you choose and configure iCloud services, you enable iCloud in Xcode. To enable iCloud, click the switch in the iCloud section. If a dialog appears asking whether Xcode should request a development certificate on your behalf, click Request. Xcode provisions your app to use iCloud.



Configuring Key-Value Storage

Key-value storage allows an app to share small amounts of data with other instances of itself running on the user's other devices. The container ID for key-value storage is `iCloud.[$(TeamIdentifierPrefix)].[$(CFBundleIdentifier)]` where the Team ID is a unique string assigned to your team. To enable key-value storage, select the "Key-value storage" checkbox. To learn how to use key-value storage for preferences, read *iCloud Design Guide*.



Configuring Document Storage

Document storage stores user documents and app data in the user's iCloud account. To enable iCloud document storage, select the "iCloud Documents" checkbox. If necessary, Xcode creates a default iCloud container for document storage. To learn how to use document storage, read *iCloud Design Guide*.

Using CloudKit

Use CloudKit to store and retrieve the app's data as records and to access it from multiple devices. In addition, you can store data in a public area where all instances of your app run by different users can access it. To enable CloudKit and add the CloudKit framework to your project, select the CloudKit checkbox. To manage your CloudKit container data model and records, click the CloudKit Dashboard button. To get started using CloudKit, read *CloudKit Quick Start*, and for details on CloudKit, read *CloudKit Framework Reference*. If necessary, Xcode creates a default iCloud container for CloudKit.

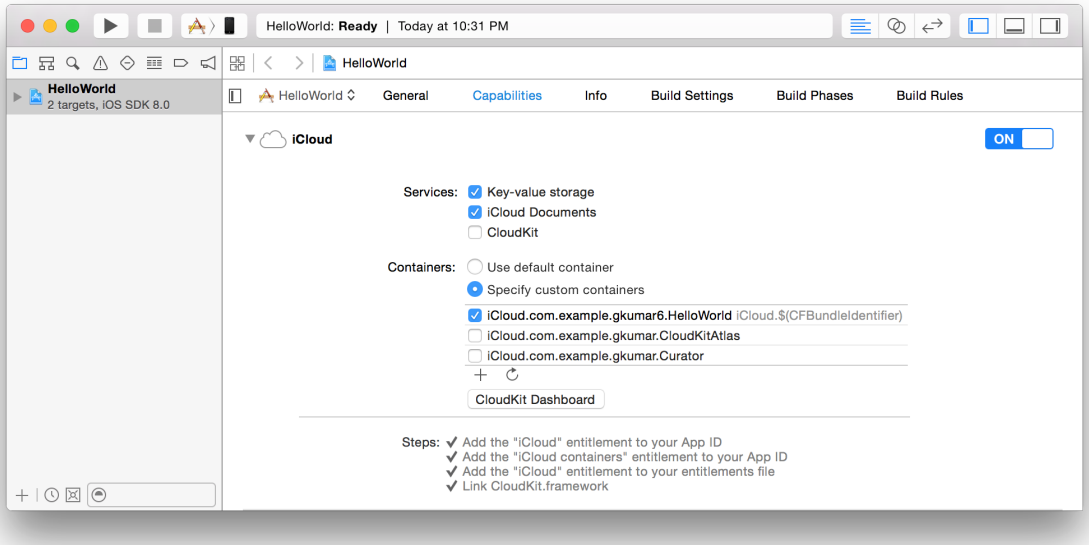
Specifying Custom Containers

For document storage and CloudKit, the default container ID is `iCloud.$(CFBundleIdentifier)` which matches the explicit App ID. Optionally, add one or more custom containers and share them between apps. You can select an existing container ID used by another app or create a new one.

To select or deselect a container ID

1. In the iCloud settings, select "Specify custom containers."
2. If necessary, click the Refresh button below the table to download container IDs used by other apps.
3. In the left of the container ID, select the checkbox to use the container and deselect the checkbox to not use the container.

Xcode updates the list of container IDs in the Xcode project entitlements file.



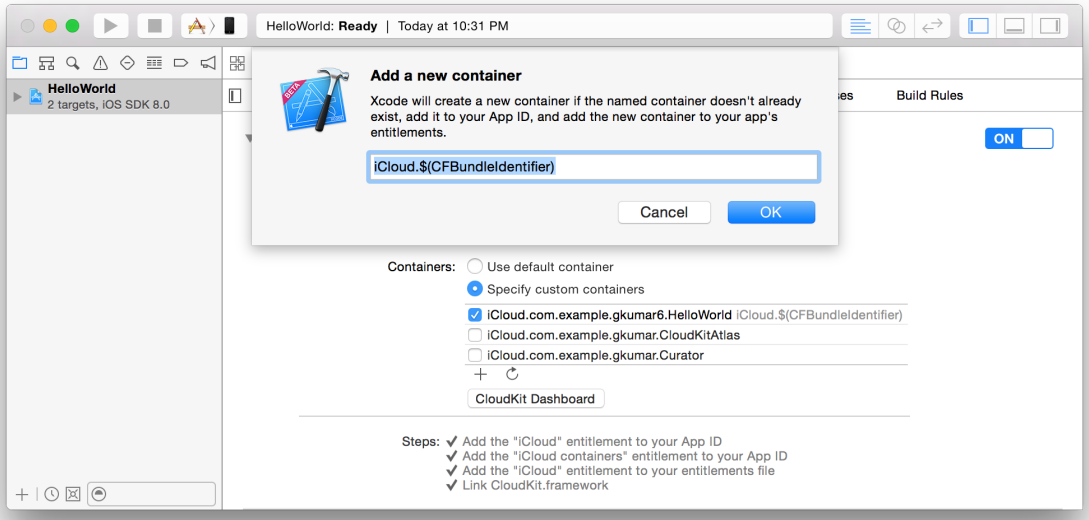
If an existing container ID is not sufficient, create another container ID for the app.

To add a container ID

1. If necessary, select “Specify custom containers.”
2. Click the Add button (+) at the bottom of the table.
3. In the dialog that appears, enter an identifier for the container you want to add.

   A container ID begins with `iCloud.` followed by a string in reverse DNS notation, as in `iCloud.com.example.gkumar.shared`.

> **Warning:** You can't delete a container ID so choose the string carefully.



4. Click OK.

   Xcode adds the new container ID to the Xcode project entitlements file and Member Center.

For guidance on selecting iCloud containers, read *iCloud Design Guide*.

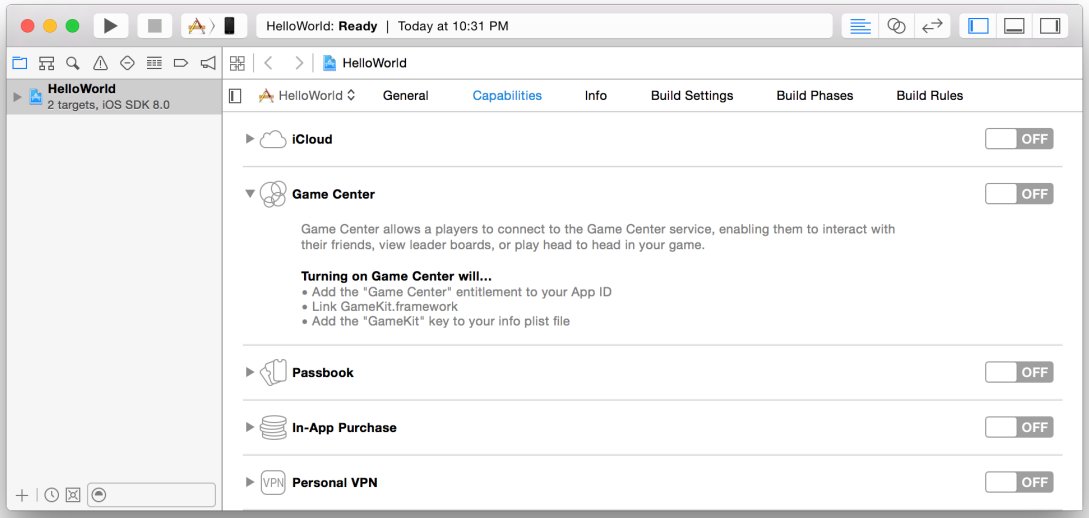
Enabling Game Center

Game Center is Apple's social gaming network. It allows players to connect their devices to the Game Center service and to exchange information.

To use Game Center, first enable Game Center in Xcode.

To enable Game Center

1. In the Capabilities pane, if Game Center isn't enabled, click the switch in the Game Center section.



2. If a dialog appears asking whether Xcode should request a development certificate on your behalf, click Request.

Xcode automatically provisions your app to use Game Center and adds the GameKit framework to your project.

For Mac apps, Xcode also sets your Outgoing network entitlements in the App Sandbox section, located in the Capabilities pane in Xcode. If your app also listens for network connections, it needs to allow incoming connections. To set additional network entitlements, read Configuring App Sandbox (Mac Only) (page 50).

For how to write your GameKit code, read *Game Center Programming Guide*. To configure your app in iTunes Connect, read Adding New Apps in *iTunes Connect Developer Guide* to create the app record (enter your explicit App ID), and read *Game Center Configuration Guide for iTunes Connect* to configure game features.

Configuring Passbook (iOS Only)

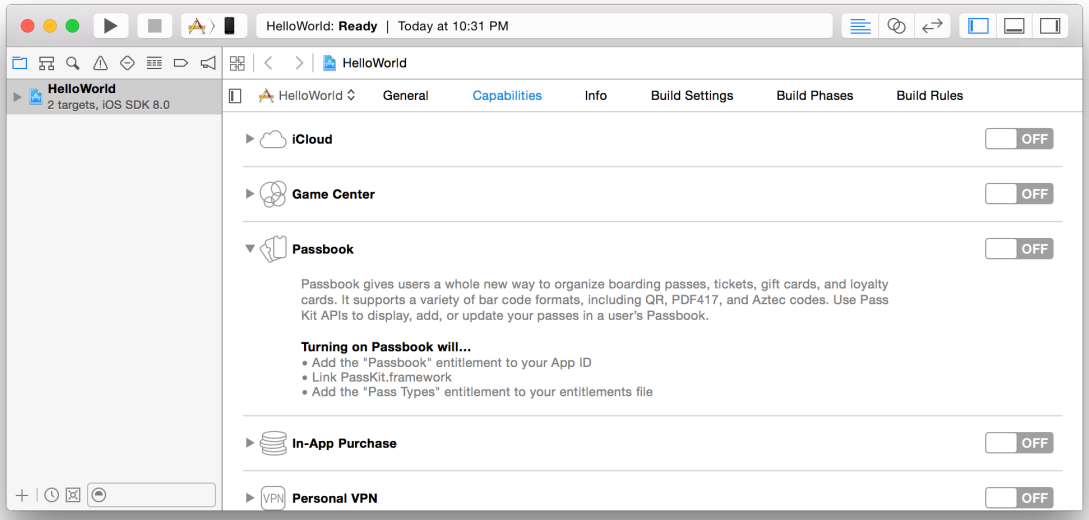
Passbook presents digital representations of information—such as a coupon, ticket for a show, or boarding pass—that allow users to redeem a real-world product or service. You can use Passbook in several ways:

- 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 Passbook from your app, use the 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.

First, you enable Passbook in your Xcode project.

To enable Passbook

1. In the Capabilities pane, if Passbook isn't enabled, click the switch in the Passbook section.



2. If a dialog appears asking whether Xcode should request a development certificate on your behalf, click Request.

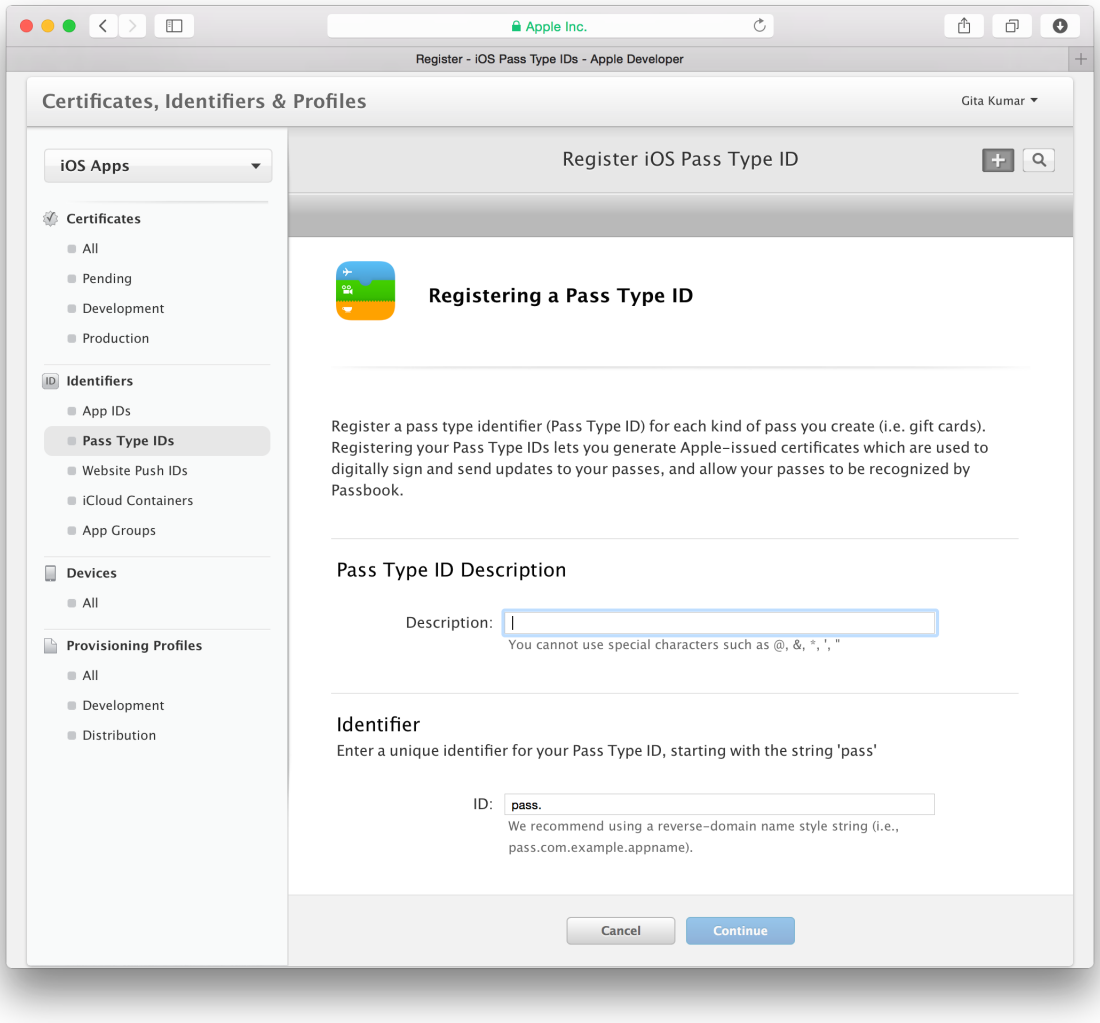
Xcode automatically provisions your app to use Passbook and adds the PassKit framework to your project.

Optionally, you can restrict your app to a subset of your pass type identifiers. This is especially useful if you develop multiple apps that use passes. If you don't have a pass type identifier, create one before enabling this feature.

To create a pass type identifier

1. In Certificates, Identifiers & Profiles, select Identifiers.
2. Under Identifiers, select Pass Type IDs.
3. Click the Add button (+) in the upper-right corner.

4. Enter a description and identifier, and click Continue.



5. Review the settings, and click Register.
6. Click Done.

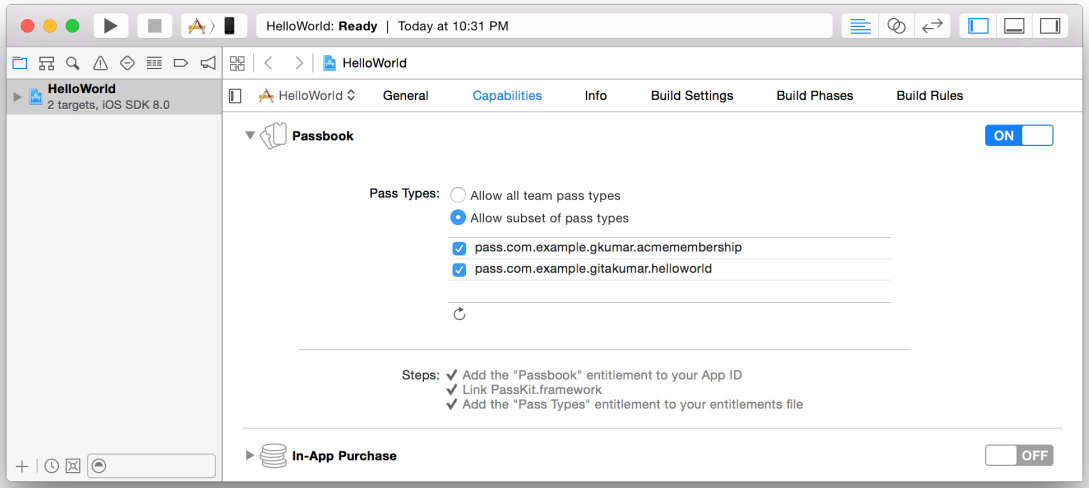
You can then use Xcode to restrict your app to a set of pass type identifiers.

To limit your app to using a subset of pass type identifiers

1. In the Capabilities pane, if necessary, click the Passbook disclosure triangle.
2. Select “Allow subset of pass types.”

   If there are no pass type identifiers in Member Center, the radio button reverts to “Allow all team pass types.”

3. If necessary, click the Refresh button under the Pass Types list to display your pass type identifiers.



4. Select the pass type identifiers you want to use.

To use a pass type identifier in your app, read Setting the Pass Type Identifier and Team ID in *Passbook Programming Guide*.

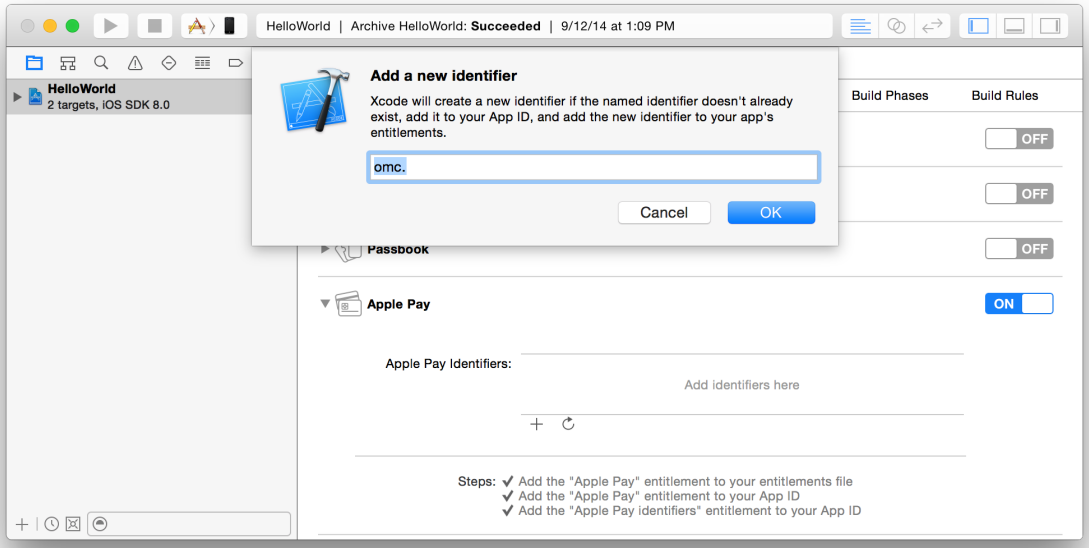
Configuring Apple Pay (iOS Only)

Apple Pay allows users to securely pay for physical goods and services using payment information stored in their iOS device.

To enable Apple Pay and create a merchant identifier

1. In the Capabilities pane, if Apple Pay isn't enabled, click the switch in the Apple Pay section.
2. Click the Add button (+) at the bottom of the Apple Pay Identifiers table.

3. In the dialog that appears, enter the identifier name and click OK.



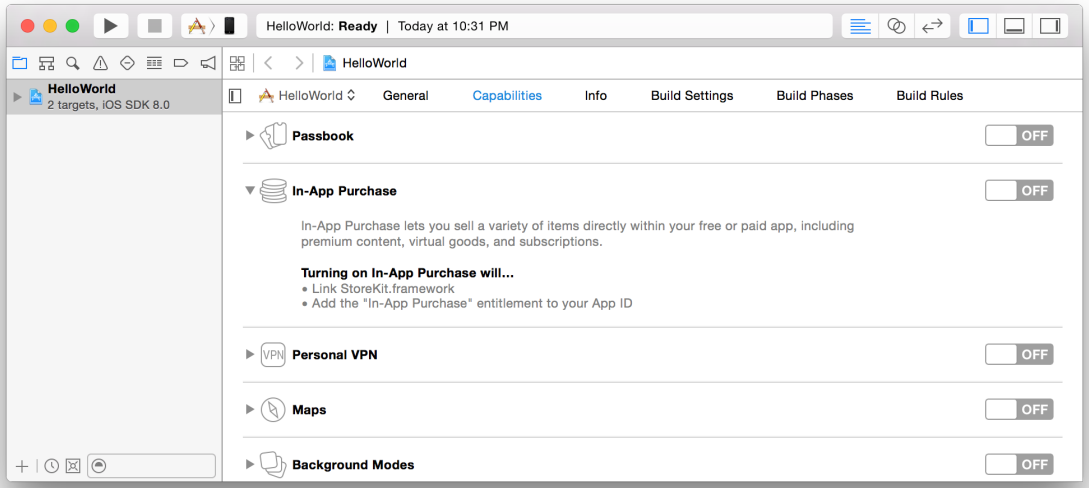
To learn more about Apple Pay, read *Apple Pay Programming Guide* and *PassKit Framework Reference*.

Enabling In-App Purchase

In-App Purchase embeds a store directly into your app by enabling 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 for additional content usable by your app. After configuring this technology in your Xcode project, you configure it in iTunes Connect. You also use iTunes Connect to create your in-app purchases.

To enable In-App Purchase

1. In the Capabilities pane, if In-App Purchase isn't enabled, click the switch in the In-App Purchase section.



2. If a dialog appears asking whether Xcode should request a development certificate on your behalf, click Request.

Xcode automatically provisions your app to use In-App Purchase and adds the StoreKit framework to your project for you.

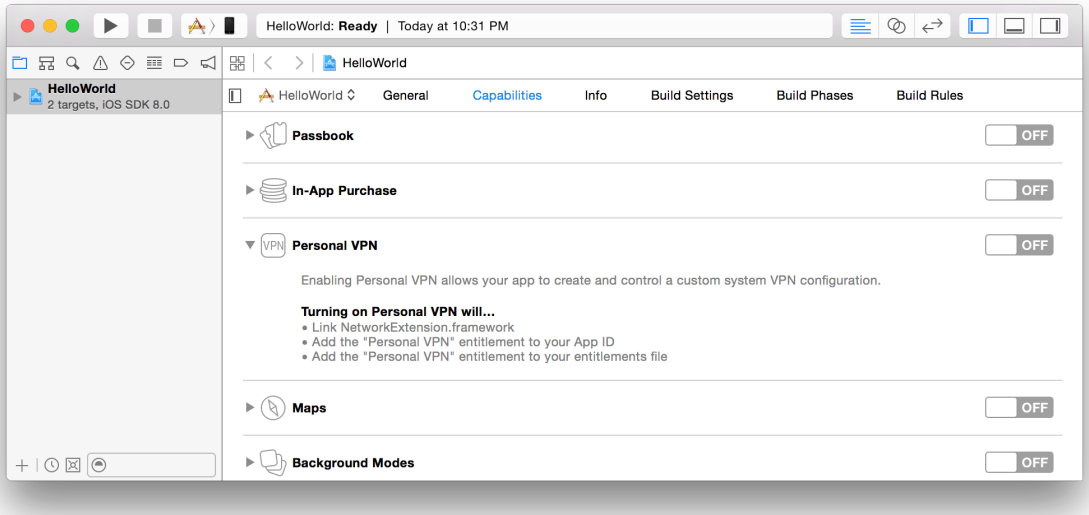
For how to write your In-App Purchase code, read *In-App Purchase Programming Guide*. To create an app record and enter the explicit App ID in iTunes Connect, read Adding New Apps in *iTunes Connect Developer Guide*. To create and upload in-app purchases, read *In-App Purchase Configuration Guide for iTunes Connect*.

Enabling Personal VPN (iOS Only)

Enable personal VPN to allow your app to create and control a custom system VPN configuration using the Network Extension framework.

To enable Personal VPN

1. In the Capabilities pane, if Personal VPN isn't enabled, click the switch in the Personal VPN section.



2. If a dialog appears asking whether Xcode should request a development certificate on your behalf, click Request.

Xcode automatically provisions your app to use personal VPN and adds the Network Extension framework to your project for you.

For information about the Network Extension framework, see the framework header files.

Configuring Maps

The Maps service allows apps to get directions or ask the Maps app to display directions. In addition, iOS 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. For both iOS and Mac apps, you use Xcode to enable the Maps service. For iOS routing apps, you use iTunes Connect to upload a geographic coverage file.

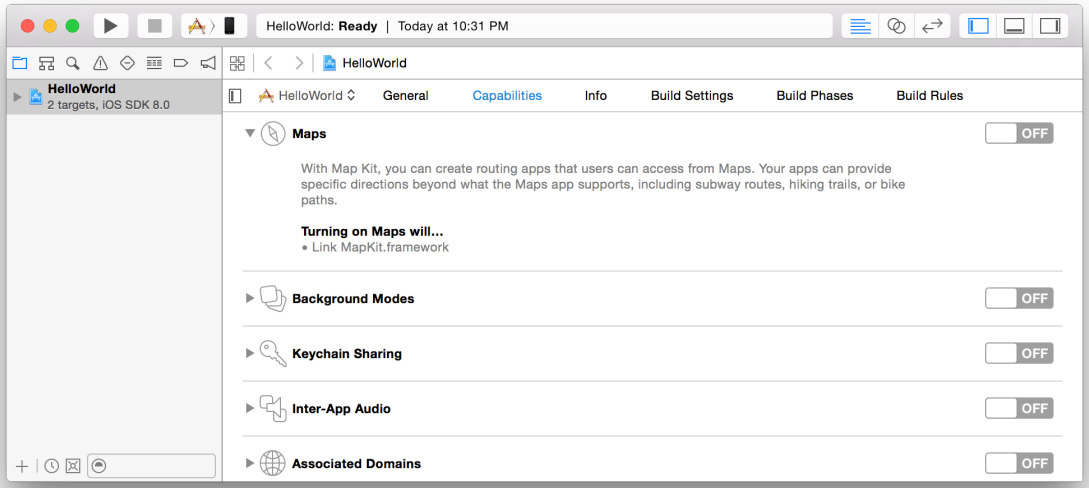
For how to write your MapKit framework code, read *Location and Maps Programming Guide*.

Enabling Maps in Xcode

Enable Maps in your Xcode project, and for iOS routing apps, select one or more supported modes.

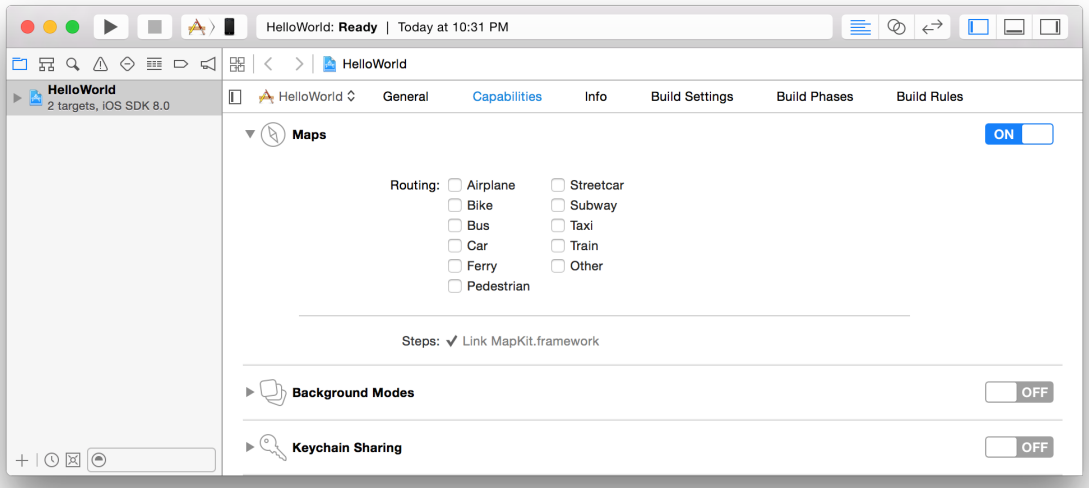
To enable Maps and select modes

1. In the Capabilities pane, if Maps isn't enabled, click the switch in the Maps section.



2. If a dialog appears asking whether Xcode should request a development certificate on your behalf, click Request.
3. For iOS routing apps, select one or more supported modes from the checkboxes below.

   You're required to select one or more supported Routing modes.



For iOS apps, Xcode adds necessary keys to your information property list and adds the MapKit framework to your project. For Mac apps, Xcode adds a Maps entitlement to the App ID and adds the MapKit framework to your project.

Configuring a Routing App (iOS Only)

You perform additional steps to configure an iOS app that provides point-to-point directions for other apps. Before continuing, review the tasks that you perform to configure an iOS routing app:

	Task
☑	Enable Maps in Xcode.
☑	Select one or more supported modes in Xcode.
☐	Write the code to provide routing directions.
☐	Create an app record and optionally, upload your app's geographic coverage file.
☐	Upload a binary of your app to the store.
☐	If necessary, upload your app's geographic coverage file.

Providing Routing Directions

To learn how to create a routing app, read Registering as a Routing App (iOS Only) in *Location and Maps Programming Guide*.

Creating an App Record in iTunes Connect

To create an app record in iTunes Connect, follow the steps in Adding New Apps in *iTunes Connect Developer Guide*. Routing apps must provide a geographic coverage file that defines the regions that your app supports. You can upload the geographic coverage file when you create the app record or later after you upload a binary, as described in Uploading the Geographic Coverage File to iTunes Connect.

Submitting a Binary to the Store

To upload a binary to iTunes Connect, follow the steps in Submitting Your App to the Store (page 126).

Uploading the Geographic Coverage File to iTunes Connect

If you submit a routing app, Apple doesn't start the approval process until you upload the geographic coverage file.

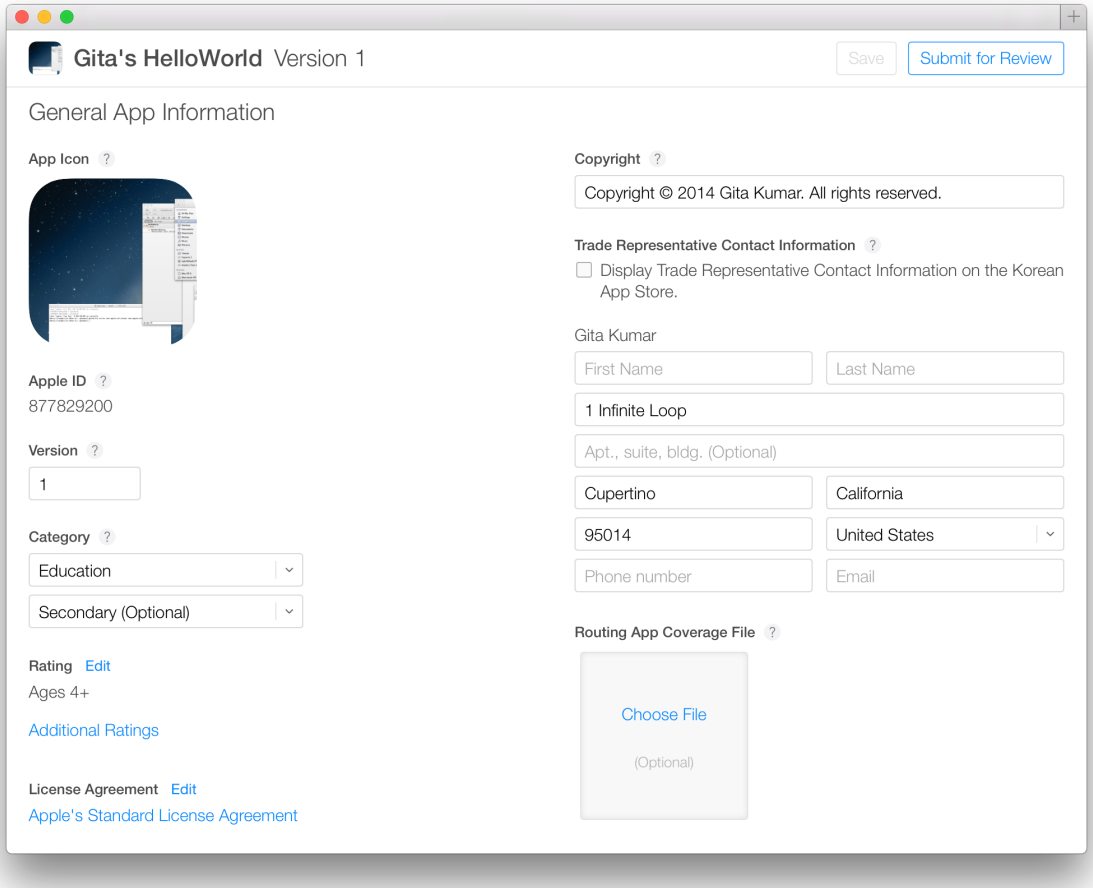
To upload the geographic coverage file after you upload your app

1. Sign in to iTunes Connect.

2. On the iTunes Connect homepage, click My Apps.
3. Locate the app you want to edit, and click the large icon or app name.

   The Versions pane appears.
4. Click the version of your app that you want to edit.

   Version information appears below.
5. Scroll to the General App Information section.
6. Click the Choose File button under Routing App Coverage File.



7. Locate the file, and click Choose.

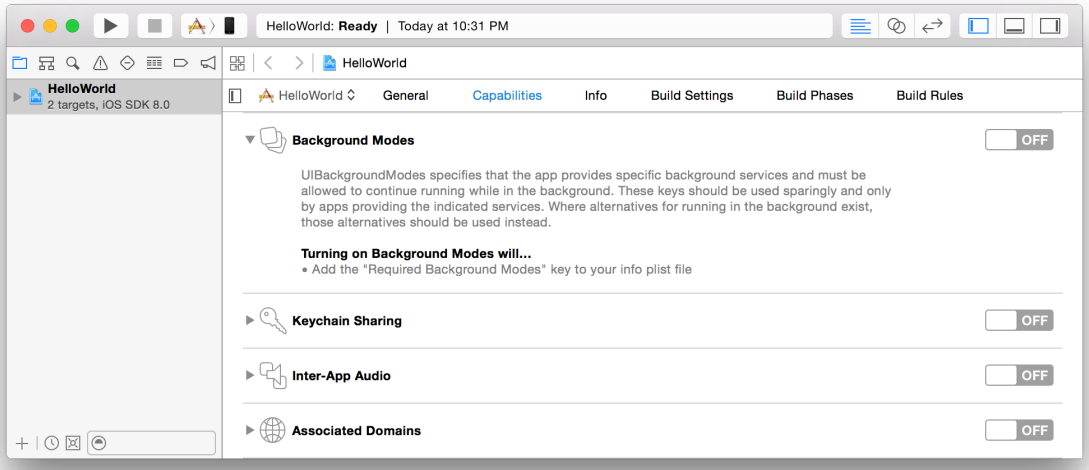
   An error message appears if the file isn't formatted correctly or has the wrong file extension.

Configuring Background Modes (iOS Only)

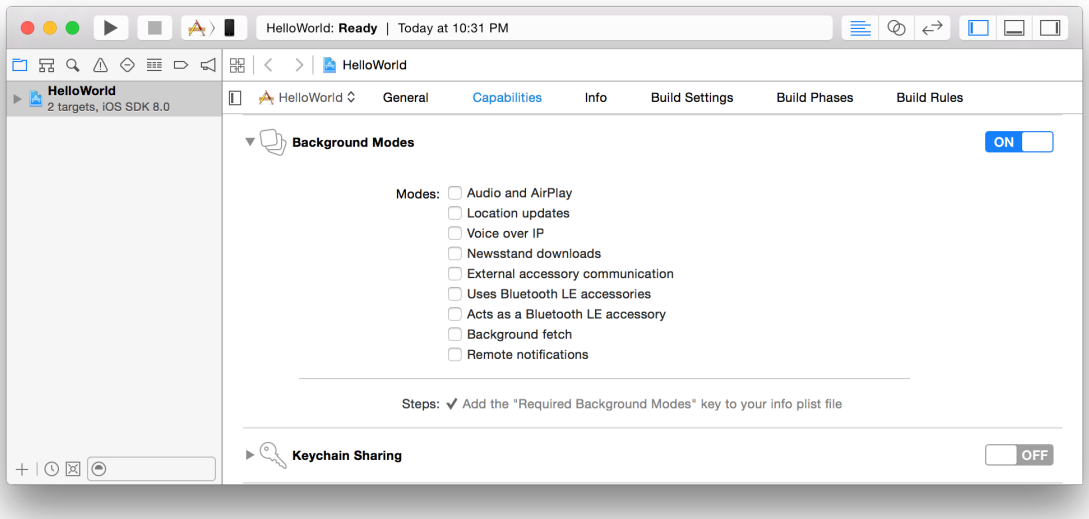
Enabling background modes allows your iOS app to continue running in the background.

To enable background modes

1. In the Capabilities pane, if Background Modes isn't enabled, click the switch in the Background Modes section.



2. If a dialog appears asking whether Xcode should request a development certificate on your behalf, click Request.
3. Optionally, select the supported modes from the checkboxes below.



Xcode adds the background modes to the information property list.

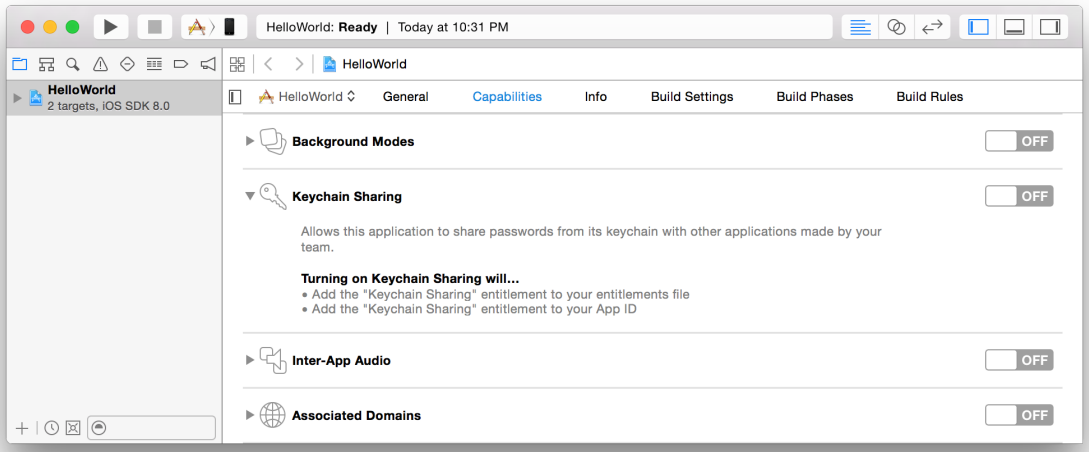
For guidance on selecting background modes, read Background Execution in *App Programming Guide for iOS*.

Configuring Keychain Sharing

Enabling keychain sharing allows your app to share passwords in the keychain with other apps developed by your team.

To enable keychain sharing

1. In the Capabilities pane, if Keychain Sharing isn't enabled, click the switch in the Keychain Sharing section.



2. If a dialog appears asking whether Xcode should request a development certificate on your behalf, click Request.

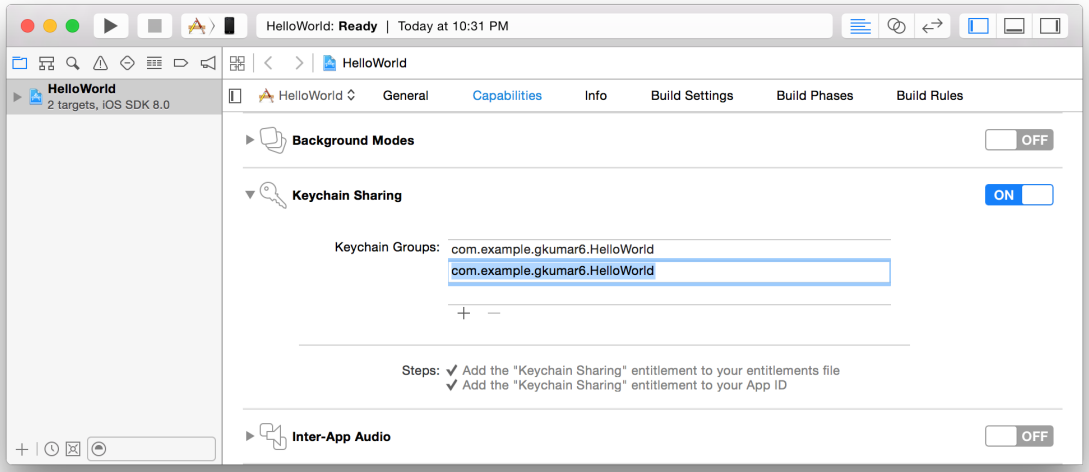
Xcode adds the `keychain-access-groups` key to the entitlements file.

If you want, you can restrict your app to a set of keychain access groups.

To limit your app to a set of keychain access groups

1. In the Capabilities pane, if necessary, click the Keychain Sharing disclosure triangle.
2. Click the Add button (+) at the bottom of the Keychain Groups area.

3. Double-click the placeholder text in the table, and enter the keychain access group you want to add.



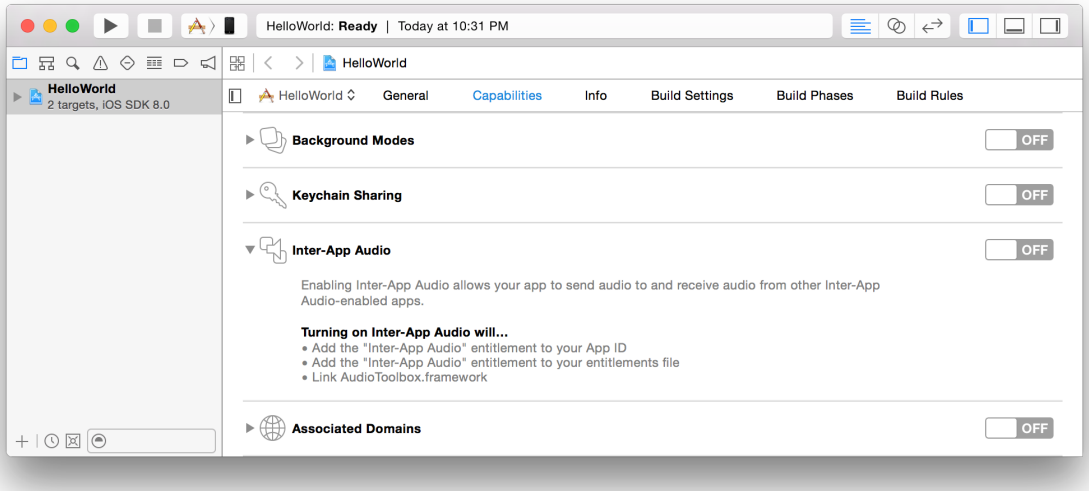
To delete a keychain access group, select it in the Keychain Groups area and click the Delete button (–).

Enabling Inter-App Audio (iOS Only)

Inter-app audio allows your iOS app to export audio that other apps can use.

To enable inter-app audio

1. In the Capabilities pane, if Inter-App Audio isn't enabled, click the switch in the Inter-App Audio section.



2. If a dialog appears asking whether Xcode should request a development certificate on your behalf, click Request.

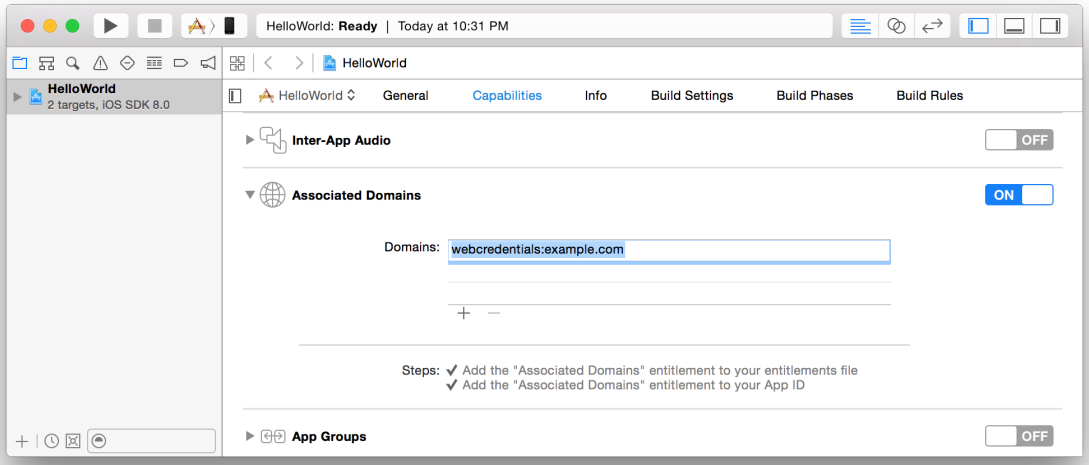
Xcode automatically provisions your app to use inter-app audio and adds the Audio Toolbox framework to your project. For how to write Audio Toolbox framework code, read *Audio Toolbox Framework Reference*.

Configuring Associated Domains (iOS Only)

Enable associated domains if you want your app to be associated with a domain to access specific services—such as Safari saved passwords and activity continuation.

To enable associated domains

1. In the Capabilities pane, if Associated Domains isn't enabled, click the switch in the Associated Domains section.
2. Click the Add button (+) at the bottom of the Domains table.
3. Double-click the placeholder text in the table, and enter the domain name you want to add.



Configuring App Groups

Use app groups to allow multiple apps access to shared containers and allow additional interprocess communication between apps. To enable app groups, in the Capabilities pane, click the switch in the App Groups section. You can select existing app groups from the table or add app groups.

To select or deselect app groups

1. If necessary, click the Refresh button below the table to download container IDs from Member Center.

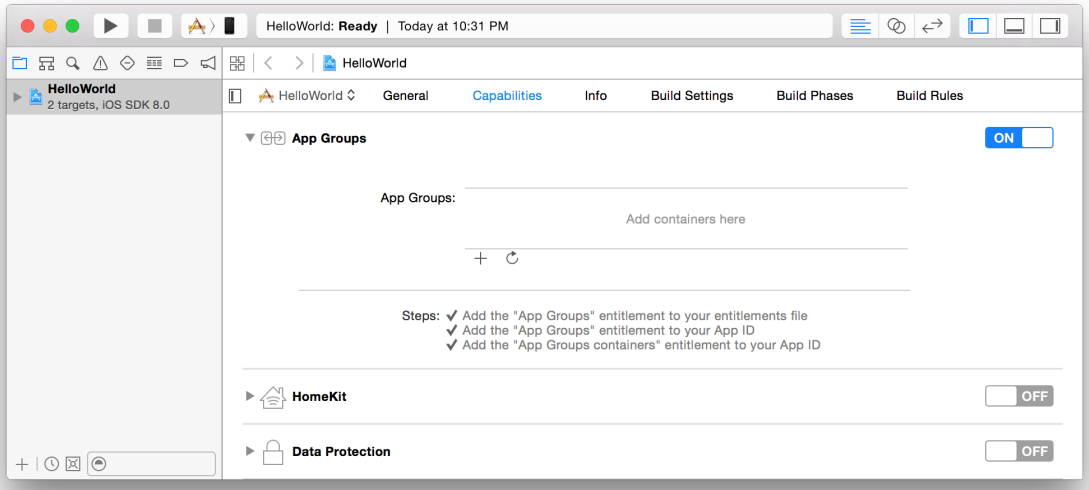
2. In the row of the container ID, select the checkbox to use the container and deselect the checkbox to not use the container.

   Xcode updates the list of app groups in the Xcode project entitlements file.

If an existing app group is not sufficient, create another app group.

To create an app group

1. Click the Add button (+) at the bottom of the App Groups table.



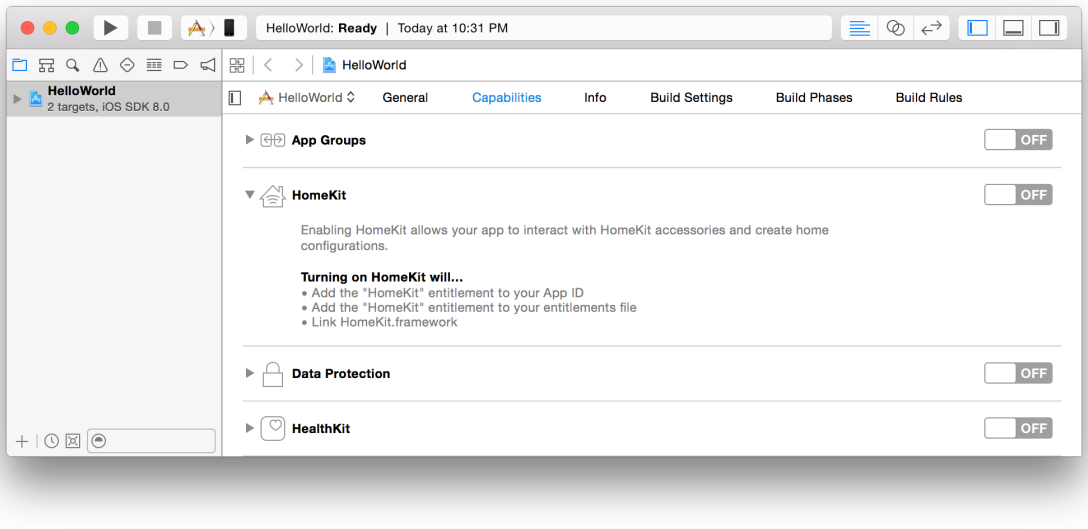
2. In the dialog that appears, enter a container ID in the text field and click OK.

   An app group container ID begins with `group.` followed by a string in reverse DNS notation.

Adding HomeKit (iOS Only)

HomeKit allows your app to communicate with and control connected accessories in a user's home. New accessories being introduced for the home are offering more connectivity and a better user experience. HomeKit provides a standard way to communicate with those accessories and create home configurations. Use HomeKit Accessory Simulator to test the communication of your HomeKit app to simulated accessories.

To add the HomeKit entitlement and the HomeKit framework to your project, click the switch in the HomeKit section.



HomeKit Accessory Simulator is not distributed with Xcode.

To download HomeKit Accessory Simulator

1. Click Download HomeKit Accessory Simulator.

   Alternatively, choose Xcode > Open Developer Tool > More Developer Tools.
2. In a browser, locate and download the “Hardware IO Tools for Xcode” DMG file.
3. In the Finder, Double-click the DMG file in `~/Downloads`.
4. Drag HomeKit Accessory Simulator to the `/Applications` folder.

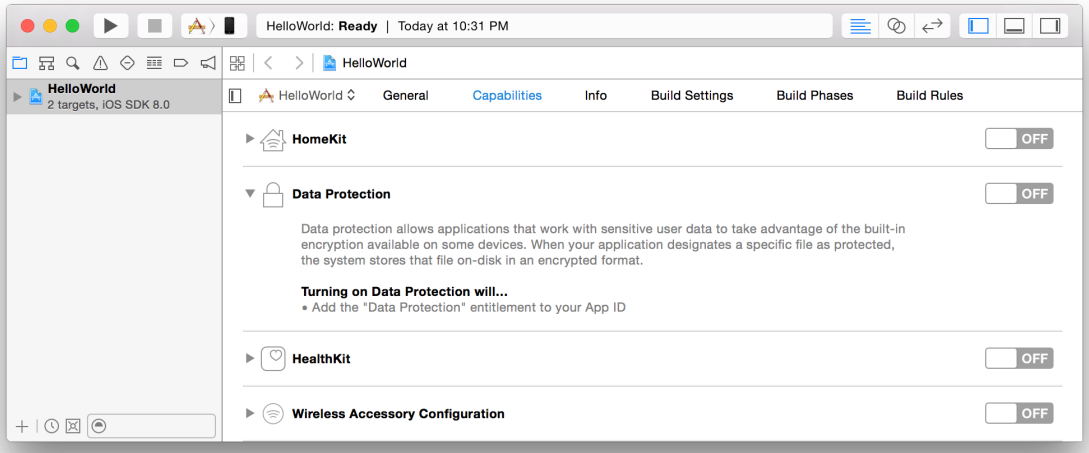
For information about the HomeKit framework, see *HomeKit Framework Reference*.

Enabling Data Protection (iOS Only)

Data protection adds a level of security to files stored on disk by your iOS 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.

To enable data protection

1. In the Capabilities pane, if Data Protection isn't enabled, click the switch in the Data Protection section.

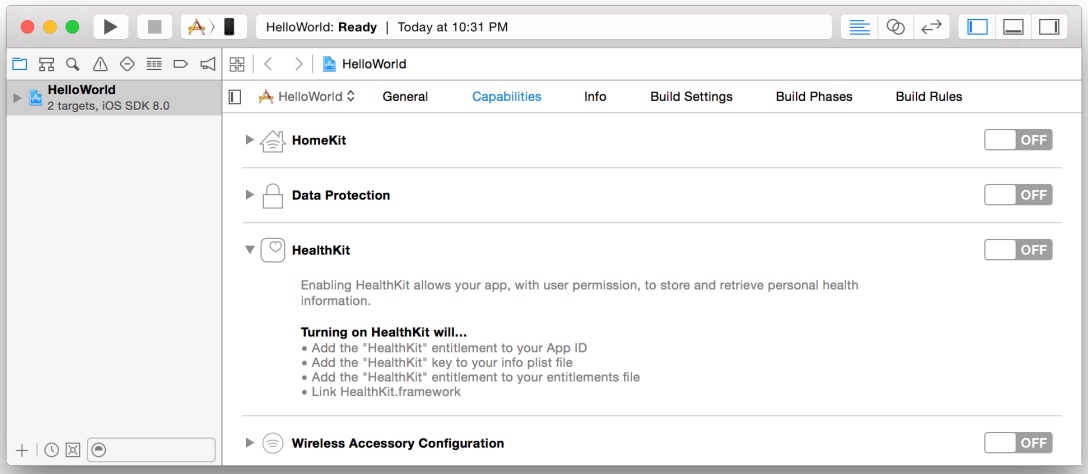


2. If a dialog appears asking whether Xcode should request a development certificate on your behalf, click Request.

The default level of protection is *complete protection*, in which files are encrypted and inaccessible when the device is locked. You can programmatically set the level of protection for files created by your app, as described in Protecting Data Using On-Disk Encryption in *App Programming Guide for iOS*.

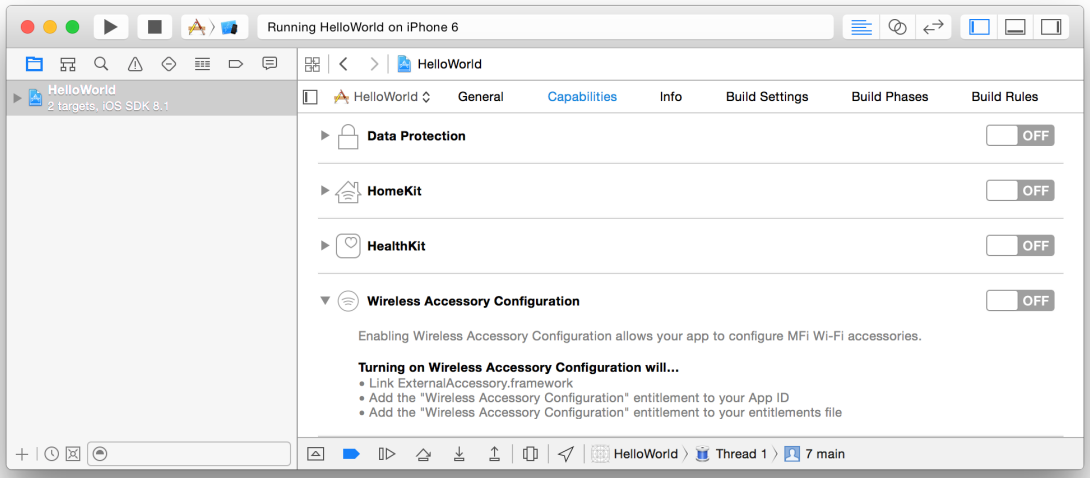
Adding HealthKit (iOS Only)

HealthKit allows your app, with user permission, to store and retrieve personal health information. To add this entitlement to your App ID and the HealthKit framework to your project, click the switch in the HealthKit section.



Enabling Wireless Accessory Configuration (iOS Only)

Enabling wireless accessory configuration adds the External Accessory framework to your project and allows your app to configure MFi Wi-Fi accessories. To enable wireless accessory configuration, click the switch in the Wireless Accessory Configuration section. Xcode adds entitlements to both your entitlements file and App ID.



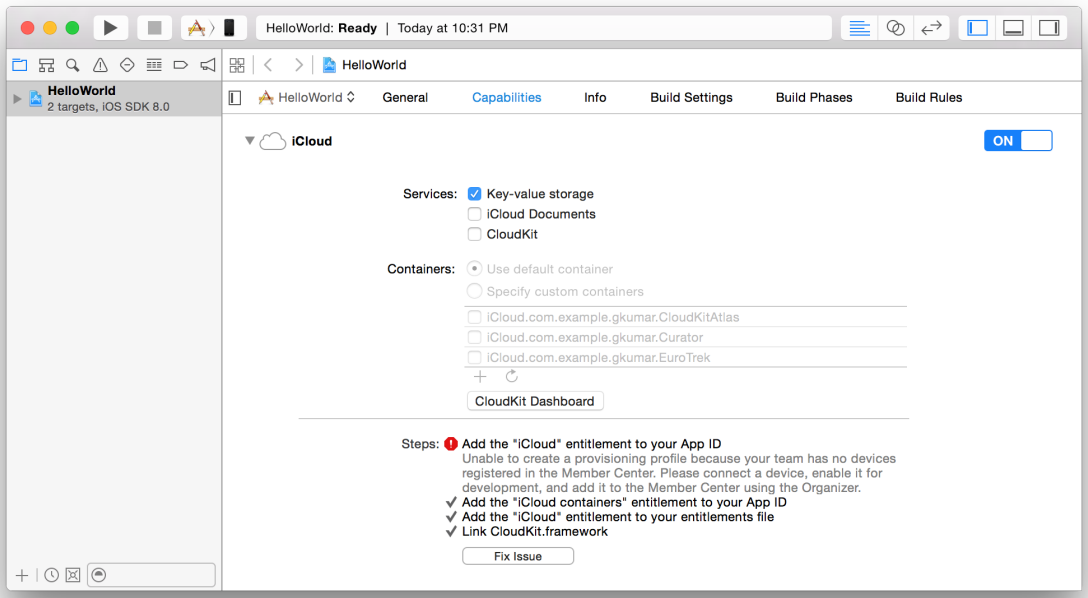
For information about the External Accessory framework, read *External Accessory Programming Topics*.

Configuring Newsstand (iOS Only)

Newsstand enables an iOS app to organize a user's magazine and newspaper app subscriptions into a folder. To use Newsstand, add some keys to the information property list and add artwork to your Xcode project. For more information on creating a Newsstand app, refer to Newsstand for Developers. For how to add Newsstand cover icons to your Xcode project, read Newsstand Icons in *iOS Human Interface Guidelines*.

Troubleshooting

If there is a problem enabling an app service, an error message appears in that area of the project editor under Steps. After reading the error message, click Fix Issue to repair the problem. If you have a development certificate and for iOS apps, an iOS device chosen from the Scheme toolbar menu, Xcode creates your team provisioning profile for you.



Recap

In this chapter, you learned how to configure app services in Xcode and, in some cases, in Member Center and iTunes Connect.

Configuring Push Notifications

Apple Push Notification service (APNs) allows an app that isn't running in the foreground to notify the user that it has information for the user. Unlike other capabilities, you don't configure push notifications in your Xcode project. To enable push notifications, you create an explicit App ID that enables push notifications and a corresponding client SSL certificate.

You can create only one explicit App ID that matches your bundle ID. Therefore, if Xcode created an explicit App ID for you—for example, when you added another capability that requires an explicit App ID—you should use it; otherwise, you create an explicit App ID that matches your bundle ID. You then generate and download a corresponding client SSL certificate—this step fully enables push notifications—and refresh provisioning profiles in Xcode. Later, you install the client SSL certificate and key on your server.

To learn more about using push notifications in your app, read *Local and Remote Notification Programming Guide*.

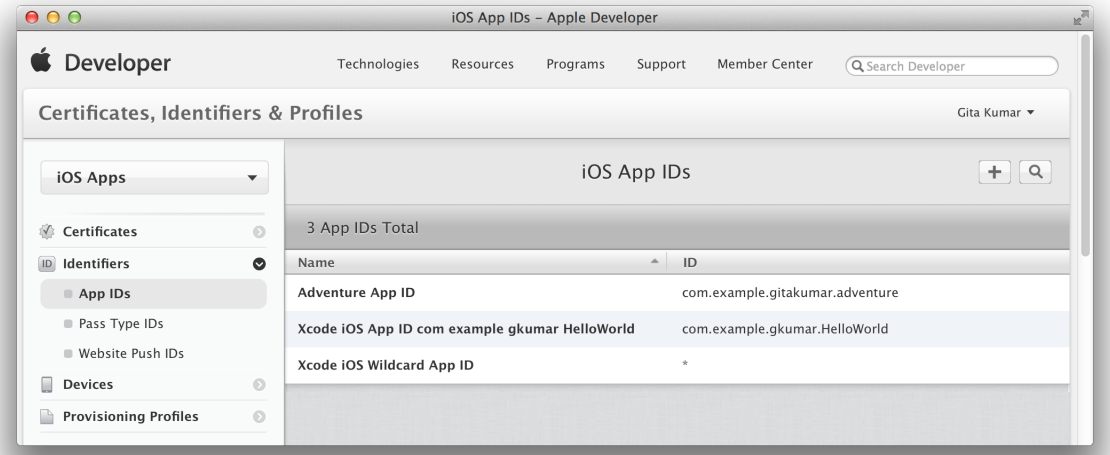
Locating Your App's Explicit App ID

Normally, Xcode creates and manages App IDs for you. For example, if you enable Maps first, Xcode uses the wildcard App ID—either for iOS apps (Xcode iOS Wildcard App ID) or for Mac apps (Xcode Mac Wildcard App ID). If you later enable Game Center or In-App Purchase, Xcode creates an explicit App ID. However, if you disable these app services, Xcode continues using the explicit App ID. So, first check Member Center to see if your app has an explicit App ID.

To locate an explicit App ID

1. In Member Center, select Certificates, Identifiers & Profiles.
2. Under Identifiers, select App IDs.
3. Locate an App ID whose ID is the same as the bundle ID.

Compare the values in the ID column with the bundle ID that appears in the General pane in the Xcode project editor. In Member Center, the name of an Xcode-managed explicit App ID begins with the text "Xcode iOS App ID" or "Xcode Mac App ID."



Creating an Explicit App ID

If no explicit App ID matches the bundle ID, create an explicit App ID, as described in Registering App IDs (page 184). When prompted to select app services, select the Push Notifications checkbox and follow the steps in Enabling Wireless Accessory Configuration (iOS Only) to generate a client SSL certificate.

Enabling Push Notifications

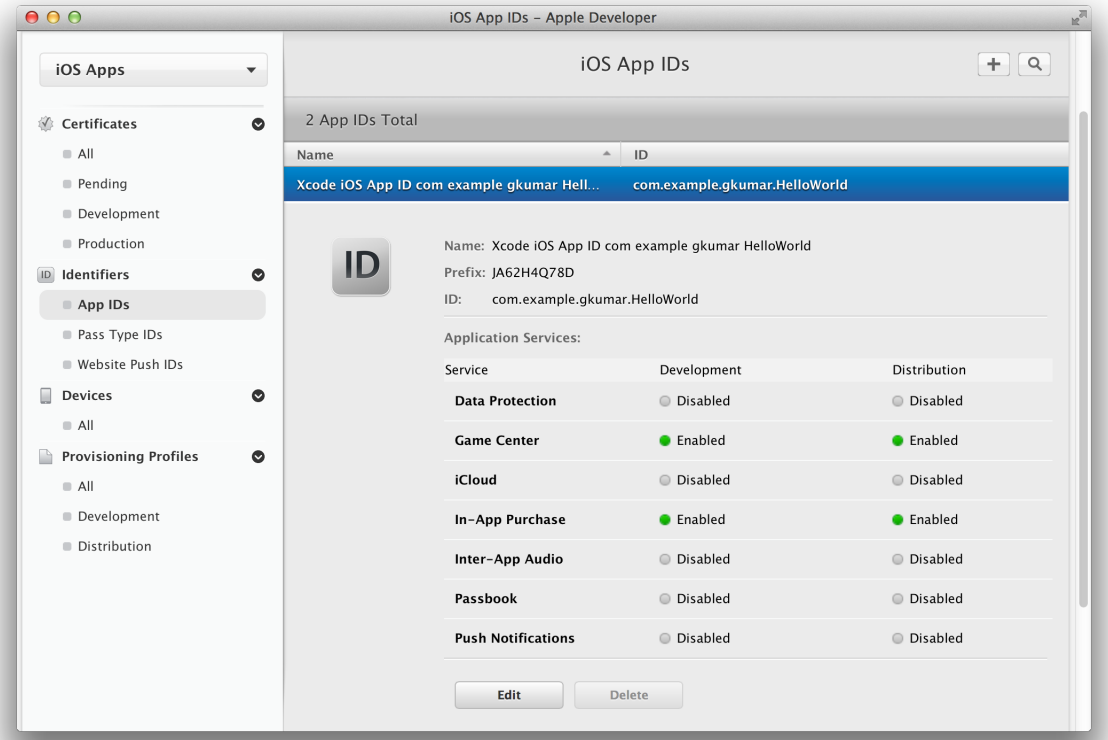
You enable push notifications when you create or edit an explicit App ID, but push notifications aren't fully enabled until you generate a client SSL certificate. A *client SSL certificate* allows your notification server to connect to the APNs. Each App ID is required to have its own client SSL certificate. As with signing certificates, you use separate client SSL certificates for development and production.

Create the development SSL certificate when you first enable push notifications and later return to Member Center to create the production SSL certificate, as described in Creating Push Notification Client SSL Certificates (page 83).

To enable push notifications

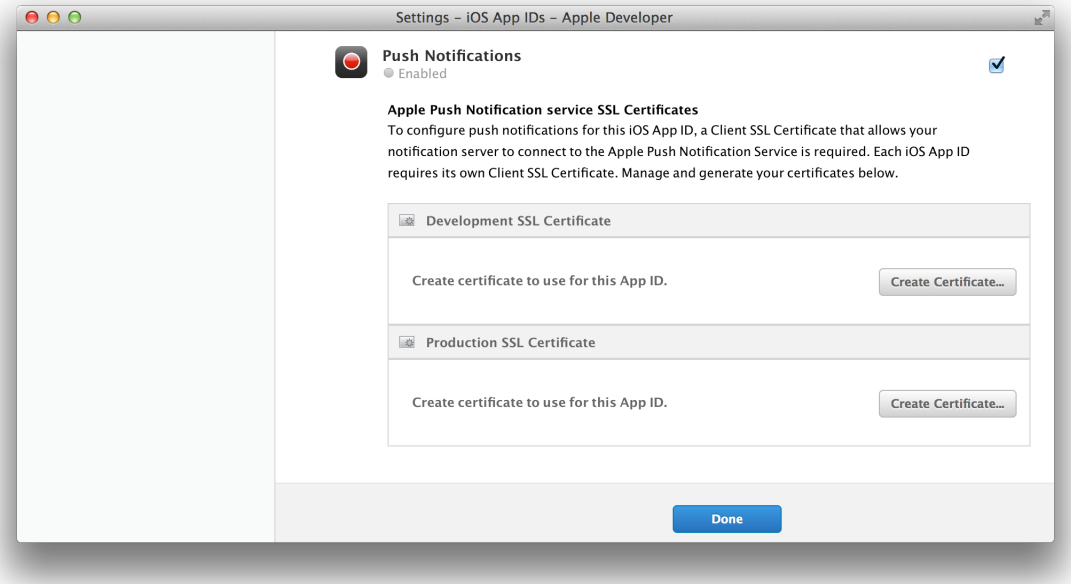
1. In Certificates, Identifiers & Profiles, select Identifiers.
2. Under Identifiers, select App IDs.

3. Select the explicit App ID, and click Edit.



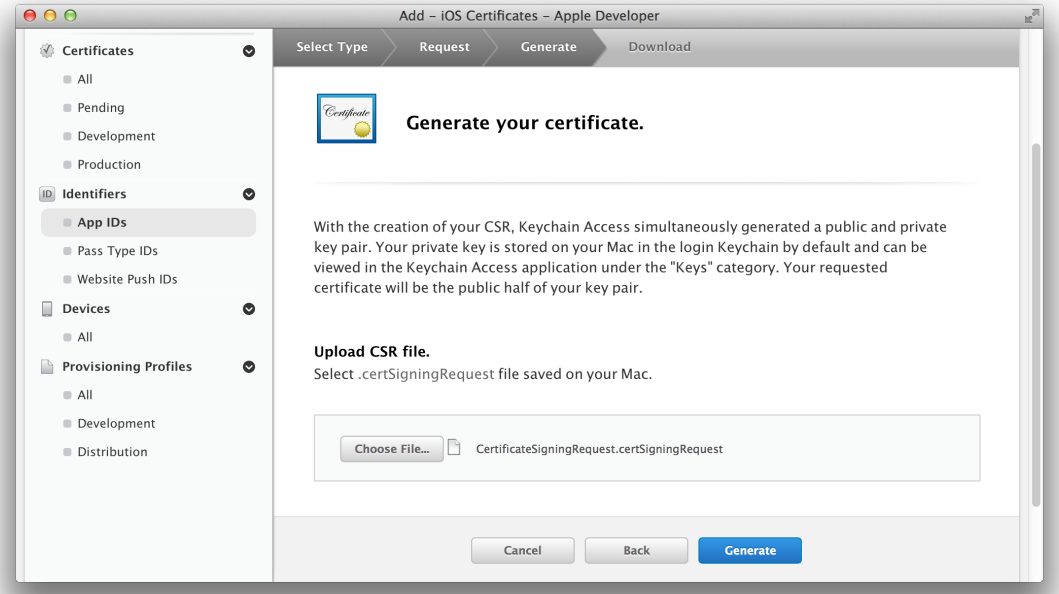
4. Scroll down and select the Push Notifications checkbox.

5. Click Create Certificate under the type of SSL certificate you want to create.



6. On the next webpage, follow the instructions to create a certificate request on your Mac, and click Continue.

7. Click Choose File.
8. In the dialog that appears, select the certificate request file (with a `.certSigningRequest` extension) and click Choose.
9. Click Generate.



10. Optionally, click Download.

    You can also download the certificate from Member Center later.
11. Click Done.

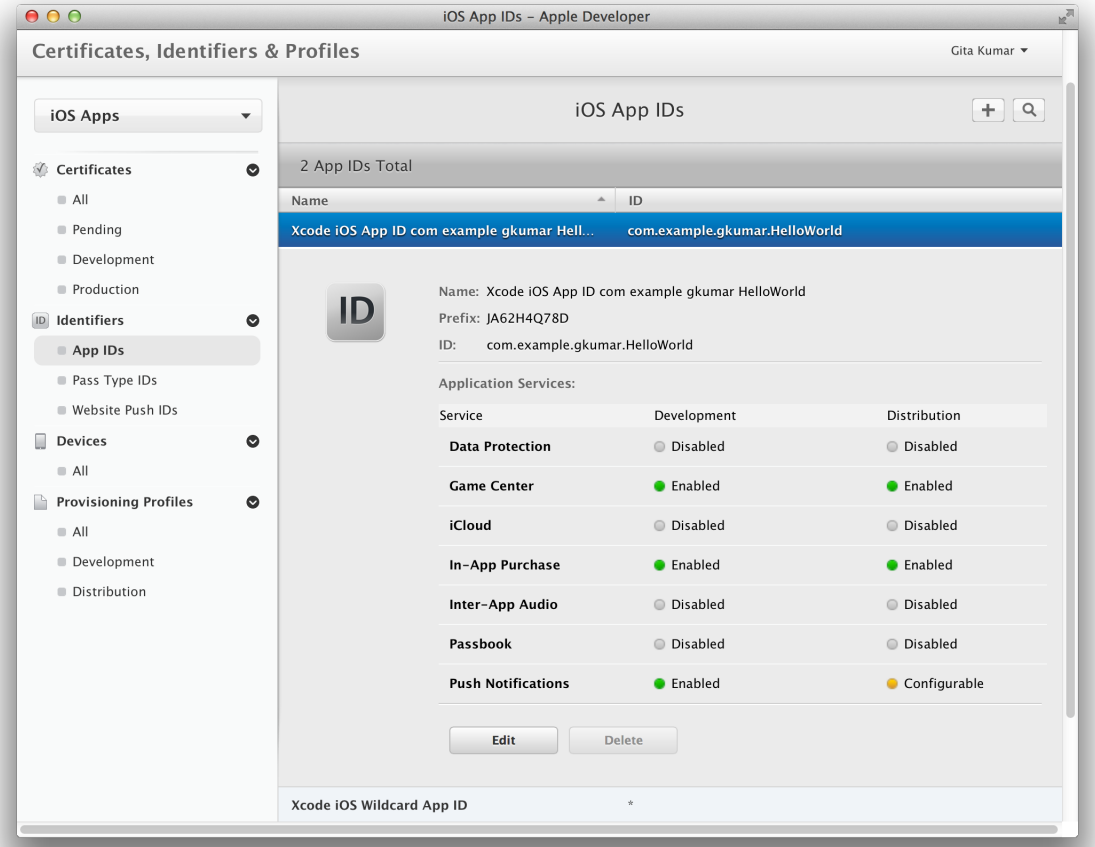
Verifying Your Steps

Verify that the App ID enables push notifications.

To verify the App ID settings

1. In Certificates, Identifiers & Profiles, select Identifiers and under Identifiers, select App IDs.
2. Select the explicit App ID that matches the bundle ID.

A green circle followed by “Enabled” appears in the Push Notifications row and Development or Distribution column depending on the type of client SSL certificate you created earlier. A yellow circle followed by “Configurable” in either the Development or Distribution column indicates a missing client SSL certificate.



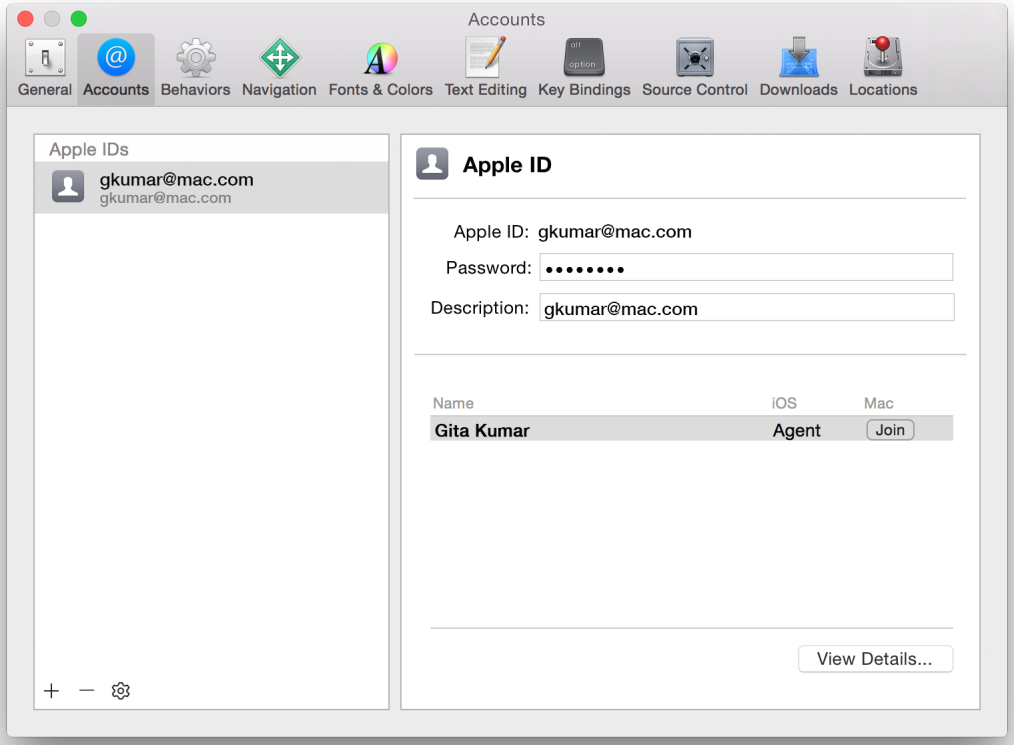
If a development SSL certificate is missing, read Creating Push Notification Client SSL Certificates (page 83) to create it.

Refreshing Provisioning Profiles in Xcode

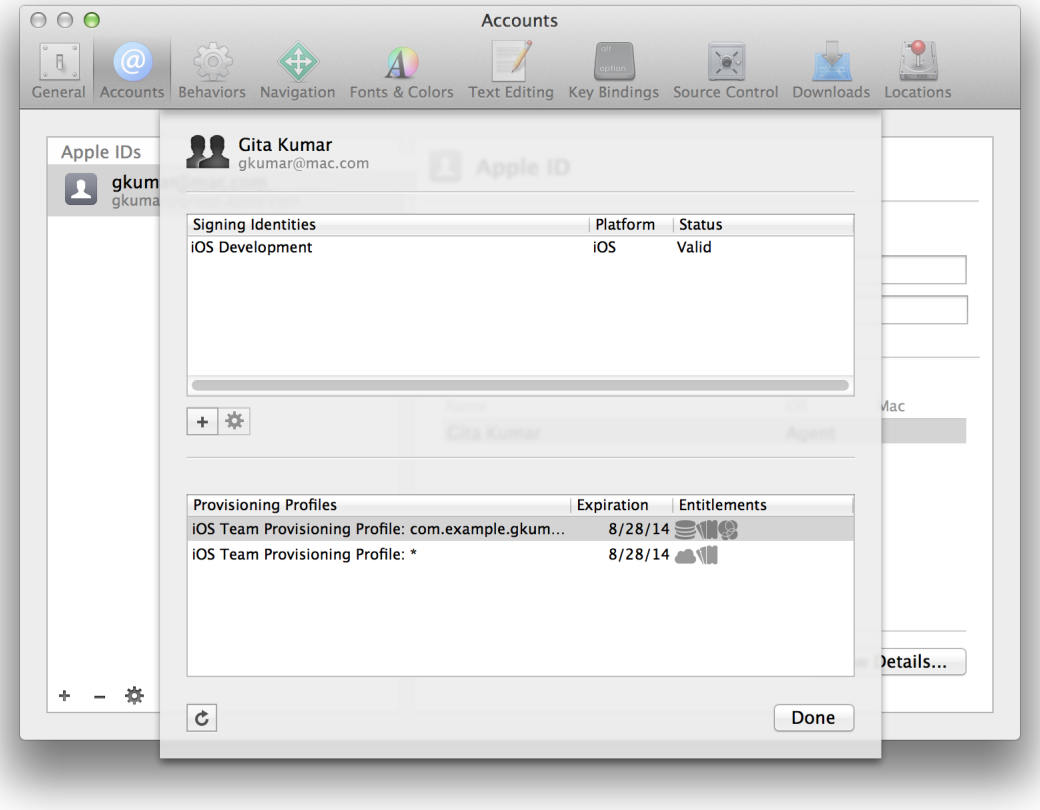
Changes you make using Member Center don’t automatically appear in Xcode. Therefore, refresh the provisioning profiles in Xcode before you start using push notifications. Xcode creates a corresponding development provisioning profile or regenerates an existing provisioning profile for you. An Xcode-managed provisioning profile that uses an explicit App ID begins with the text “iOS Team Provisioning Profile:” or “Mac Team Provisioning Profile:” followed by the bundle ID.

To refresh provisioning profiles

1. In the Xcode Preferences window, click Accounts.
2. Select your team, and click View Details.



3. In the dialog that appears, click the Refresh button in the lower-left corner.



4. If a dialog appears asking to create your distribution certificate, click Not Now or Request.
5. Click Done.

Verifying Your Steps

Use Member Center to verify that the provisioning profile was either created or regenerated to enable push notifications.

To verify that a provisioning profile enables push notifications

1. In Certificates, Identifiers & Profiles, select Identifiers.
2. Under Provisioning Profiles, select Development.
3. Select the Xcode-managed provisioning profile that matches the bundle ID.

Push notifications should appear in the Enabled Services list. (In-App Purchase and Game Center are enabled by default for an explicit App ID.)

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.

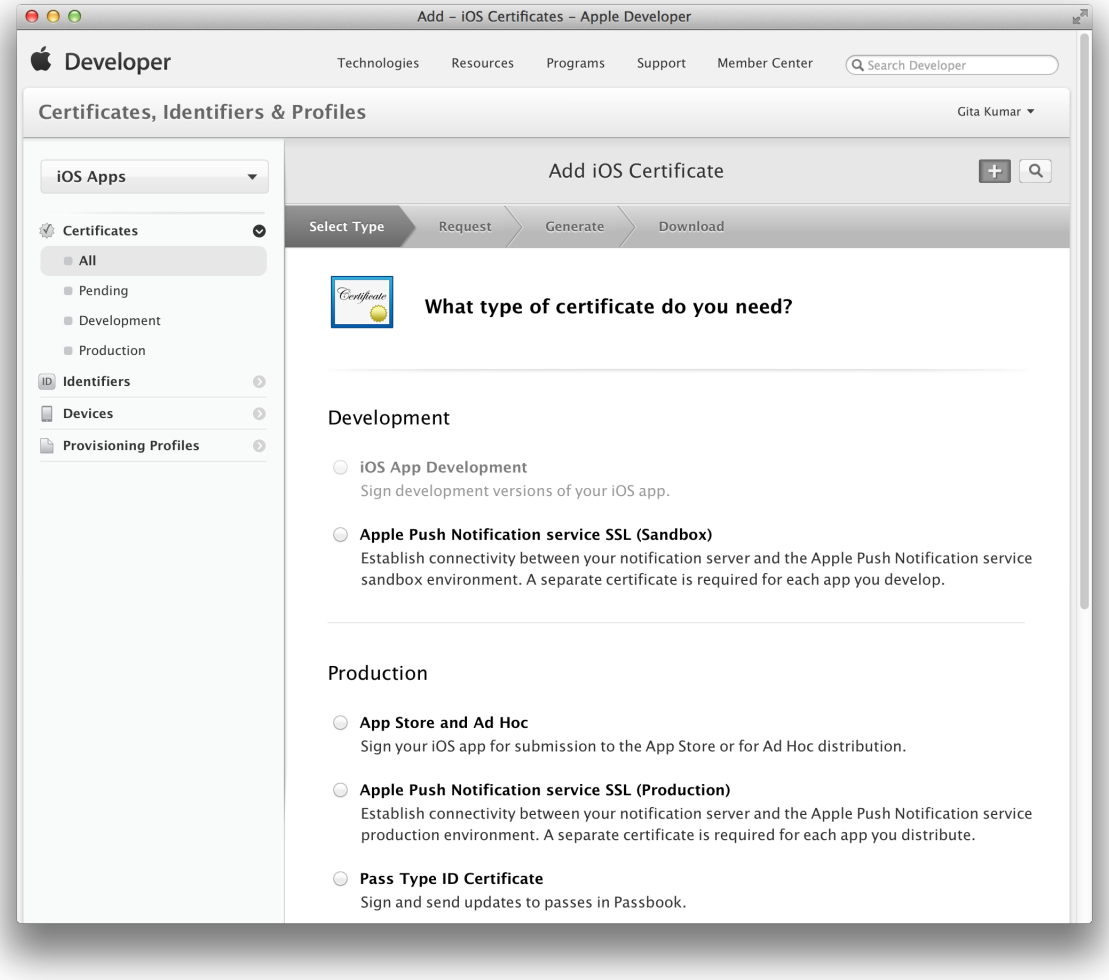
If you don't have an explicit App ID that matches the bundle ID, create an explicit App ID, as described in Creating an Explicit App ID (page 77), before continuing. Otherwise, follow these steps to create either a development or production client SSL certificate.

To generate client SSL certificates

1. In Certificates, Identifiers & Profiles, select Certificates.

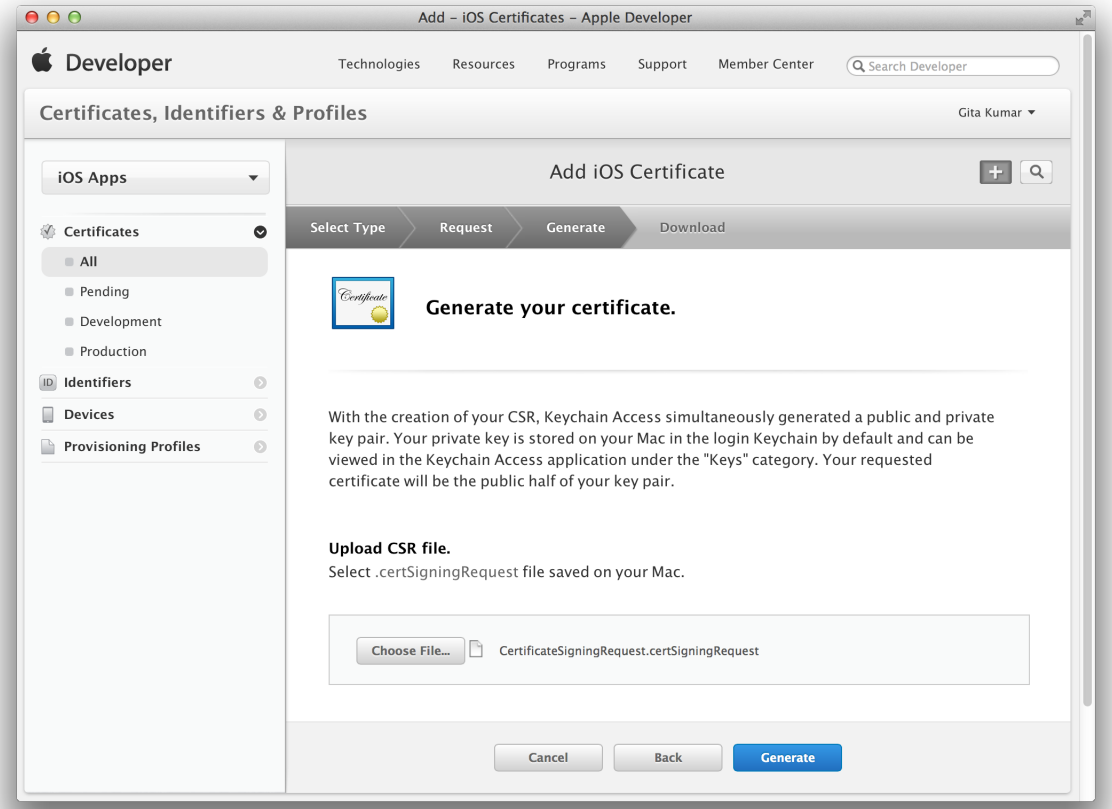
2. Click the Add button (+) in the upper-right corner.

3. Under either Development or Production, select the “Apple Push Notification service SSL” checkbox, and click Continue.



4. Choose an App ID from the App ID pop-up menu, and click Continue.
5. Follow the instructions on the next webpage to create a certificate request on your Mac, 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.



9. Optionally, click Download.
10. Click Done.

Installing a Client SSL Signing Identity on the Server

Install the SSL distribution signing identity you obtained earlier on the server that runs the provider code and connects with the development or production version of APNs. Export the signing identity from the keychain on the Mac where you created it, and copy it to the appropriate place on the server.

To export a client SSL signing identity

1. Launch Keychain Access.
2. In the Category section, select My Certificates.
3. Find the certificate you want to export and disclose its contents.

   You'll see both a certificate and a private key.

4. Select both the certificate and the key, and choose File > Export Items.
5. From the File Format pop-up menu, choose a file format that your server accepts.
6. Enter a filename in the Save As field, and click Save.

   Files in the Personal Information Exchange format have a `.p12` extension, and files in the Privacy Enhanced Mail format have a `.pem` extension.

For techniques to resolve push notification server issues, read *Troubleshooting Push Notifications*.

Launching Your App on Devices

All iOS apps and most Mac apps must be code signed and provisioned to launch on a device. *Provisioning* is the process of preparing and configuring an app to launch on devices and use app services. Xcode uses information you provide to create a team provisioning profile for you when you assign your Xcode project to a team or the first time you add capabilities to your app. For example, Xcode offers to create your development certificate and automatically registers a connected iOS device or your Mac. Xcode uses this information to create a team provisioning profile that's ultimately installed on the device. For iOS apps, Xcode runs an app on a device (an iPad, iPhone, or iPod touch) if that device is in the provisioning profile. Similarly, a Mac app that uses certain app services launches only if the Mac is in the provisioning profile.

Tip: To avoid issues later when you distribute your app, test your app on a device using the debug navigator, as described in *Debug Navigator Help*, test navigator, as described in *Test Navigator Help*, and Instruments app, as described in *Instruments User Guide*.

Launching Your Mac App

To launch your Mac app, click the Run button in the project navigator. Xcode automatically registers your Mac and adds it to the team provisioning profile. If a prompt appears asking whether `codesign` can sign the app using a key in your keychain, click Always Allow.

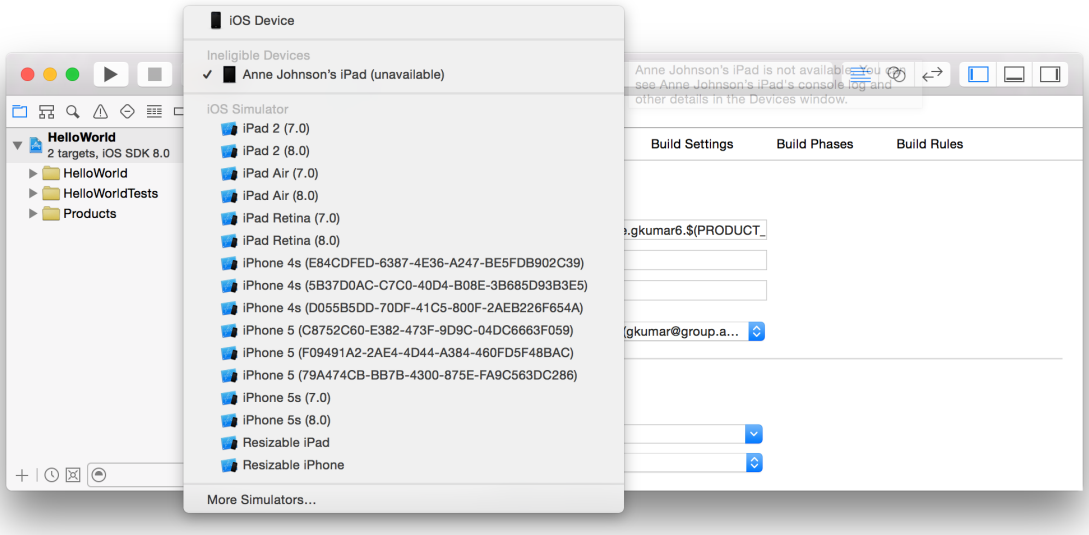
Launching Your iOS App on a Connected Device

It takes just a few steps to launch your app on a device if you previously created your code signing identity and team provisioning profile, as described in Creating the Team Provisioning Profile (page 32). Otherwise, a series of dialogs and warnings may appear as Xcode resolves the code signing issues in the process of launching your app.

To launch an iOS app on a device

1. Connect an iOS device to your Mac.
2. In the project navigator, choose your device from the Scheme toolbar menu.

Xcode assumes you intend to use the selected device for development and automatically registers it for you.



If your iOS device is disabled in the Scheme toolbar menu because it is ineligible, fix the issue before continuing. For example, if the device doesn't match the deployment target, upgrade the version of iOS on the device or choose the version you want to target from the Deployment Target pop-up menu in the Deployment Info section and then select the iOS device from the Scheme toolbar menu.

3. Click the Run button.

   Xcode installs the app on the device before launching the app.

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

While developing your app, run and test it on all of the iOS devices and iOS versions that you intend to support. Because different instruments are available in iOS Simulator, also test your app in iOS Simulator using Instruments and other tools before distributing your app. For details on how to use iOS Simulator to test your app, read *iOS Simulator User Guide*.

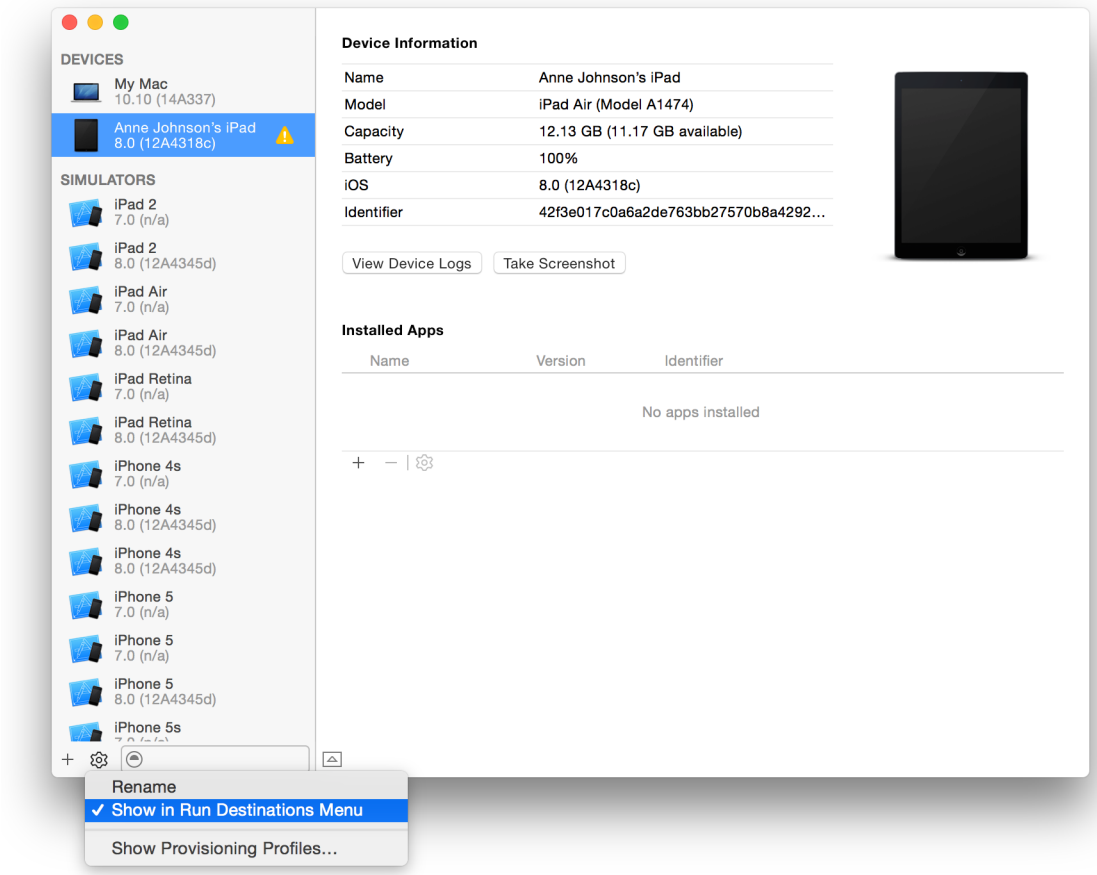
Removing an iOS Device from the Scheme Menu

If you want Xcode to ignore a connected iOS device—you don't want Xcode to add it to the team provisioning profile—remove it from the Scheme menu.

To remove an iOS device from the Scheme menu

1. Connect the iOS device to your Mac.
2. Choose Window > Devices, and select the iOS device under Devices.

3. In the lower-left corner of the Devices window, click the Action button (the gear icon to the right of the Add button).



4. Deselect "Show in Run Destinations Menu" from the pop-up menu.

   In the project editor, the iOS device disappears from the Scheme menu.

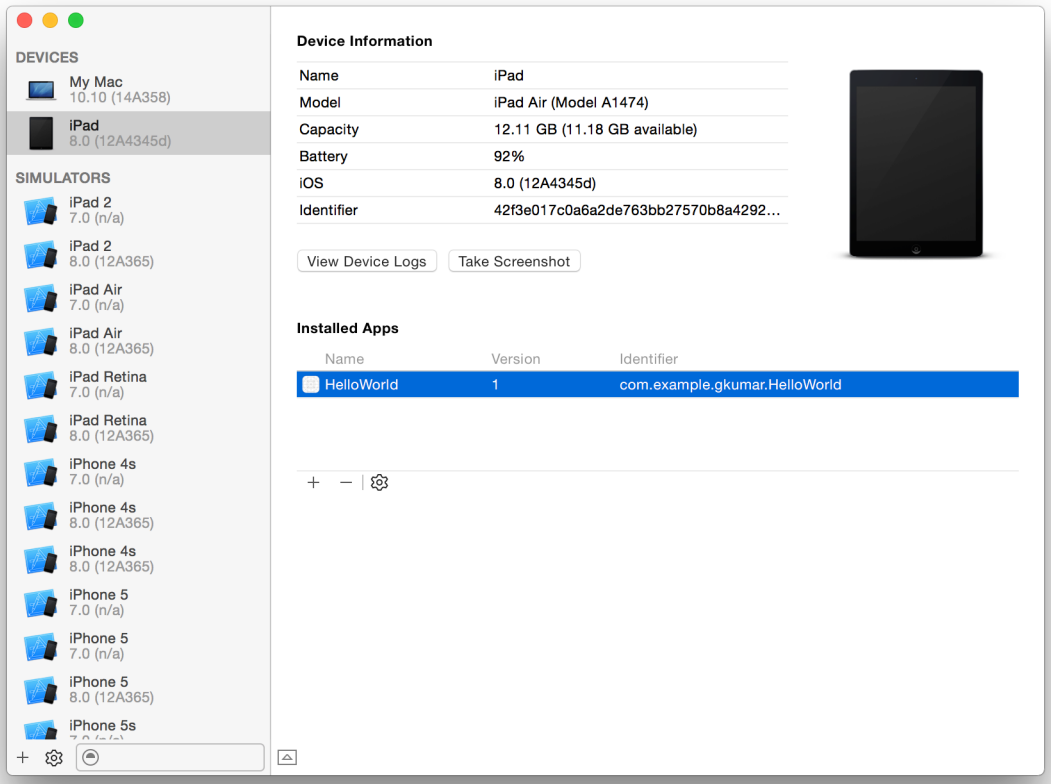
Removing an App from an iOS Device

When you launch an app on an iOS device, Xcode installs the app on the device. Later, you can remove the app from the device using the Devices window in Xcode.

To remove an app from an iOS device

1. Connect the iOS device to your Mac.
2. Choose Window > Devices, and select the iOS device under Devices.

3. In the Apps section, select the app from the table and click the Delete button (–) in the lower-left corner of the table.



Verifying Your Steps

To learn more about how Xcode provisions your app, examine the team provisioning profile in Member Center. You can verify that the device was registered and added to the team provisioning profile.

To verify that your device is registered

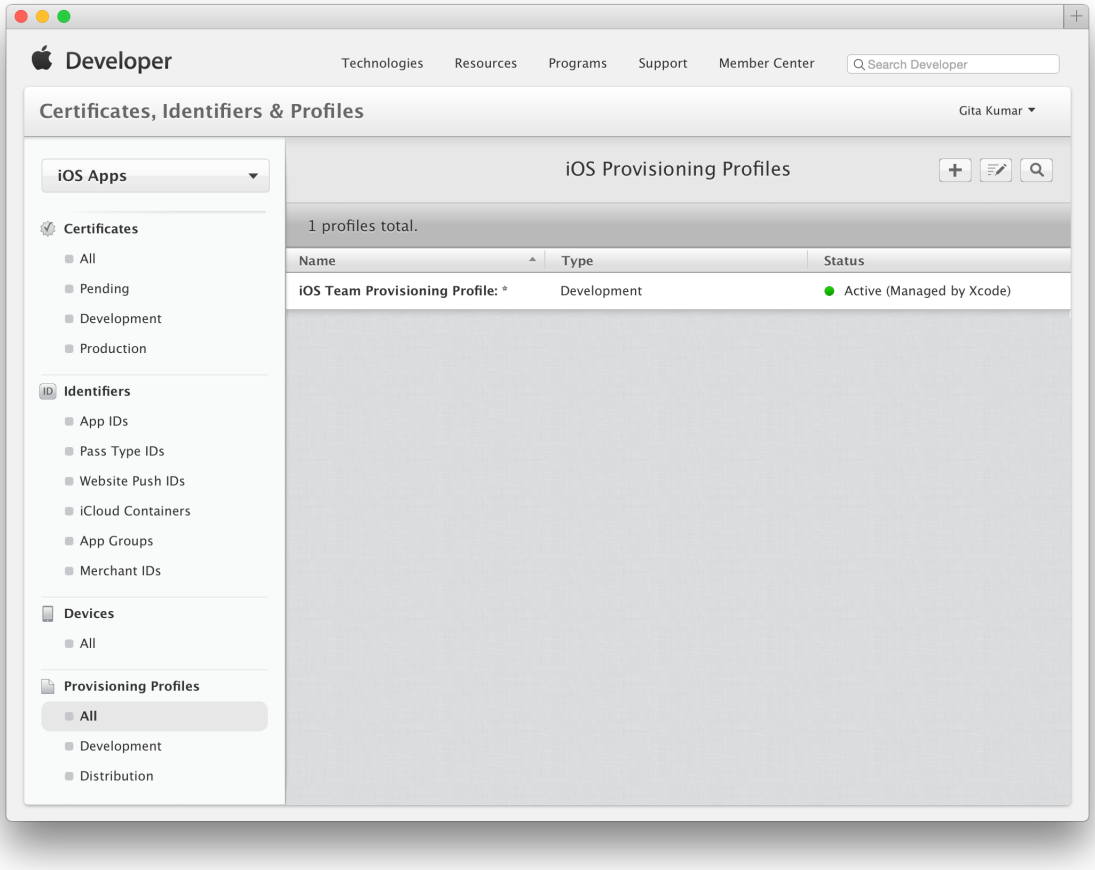
1. In Certificates, Identifiers & Profiles, select Devices.
2. Under Devices, select All.

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.

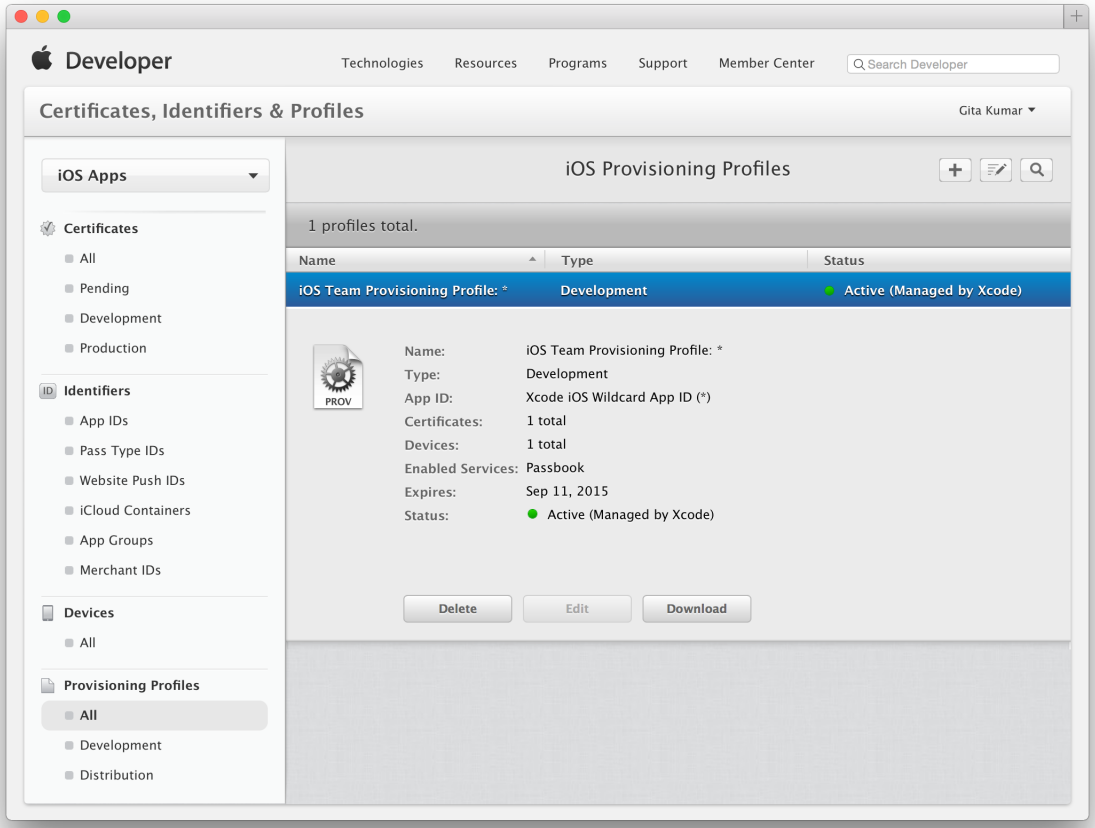
The team provisioning profile, starting with the text “iOS Team Provisioning Profile” or “Mac Team Provisioning Profile,” should appear under iOS Provisioning Profiles or Mac Provisioning Profiles.



You may have multiple team provisioning profiles depending on the capabilities you add and the number of apps.

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 enabled devices contained in your account. If you're an individual, you should have only one development certificate.

Troubleshooting

There are several reasons why your app may not run on a device.

- If the device appears ineligible in the Scheme toolbar menu, fix the issue before continuing.
- If a warning or error message appears below the Team pop-up menu in the General pane of the project editor, follow the steps in Creating the Team Provisioning Profile (page 32) to fix the issue.
- Read Adding Capabilities (page 48) to fix similar provisioning errors in the Capabilities pane.

Assuming that you followed and verified the steps in this chapter, check the Xcode project settings. The configuration of your project may not match the configuration of your device.

- 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 36).

Team Provisioning Profiles in Depth

A *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 app services during development. For an individual, a team provisioning profile allows all apps signed by you to run on all of your registered devices. For a company, a team provisioning profile allows *any* app developed by a team to be signed by *any* team member and installed on *any* team device.

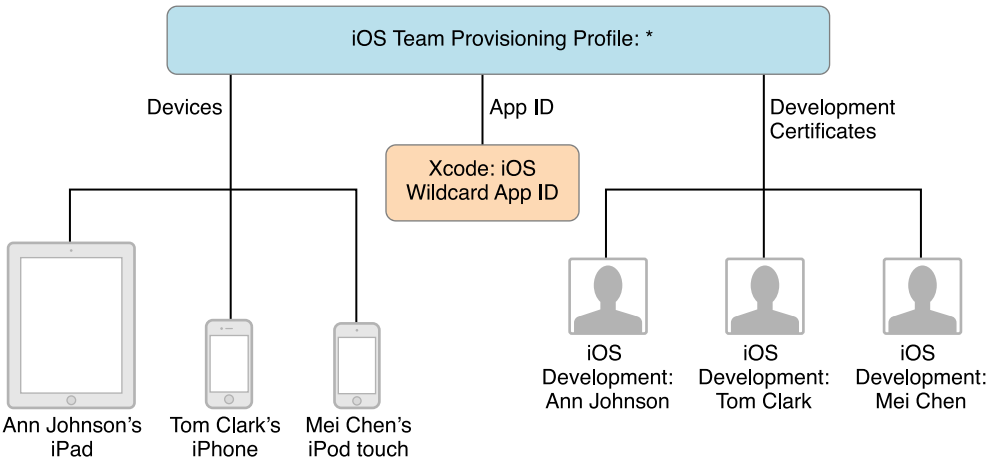
The team provisioning profile contains:

- A wildcard App ID that matches all your team's apps or an explicit App ID that matches a single app
- All devices associated with the team
- All development certificates associated with the team

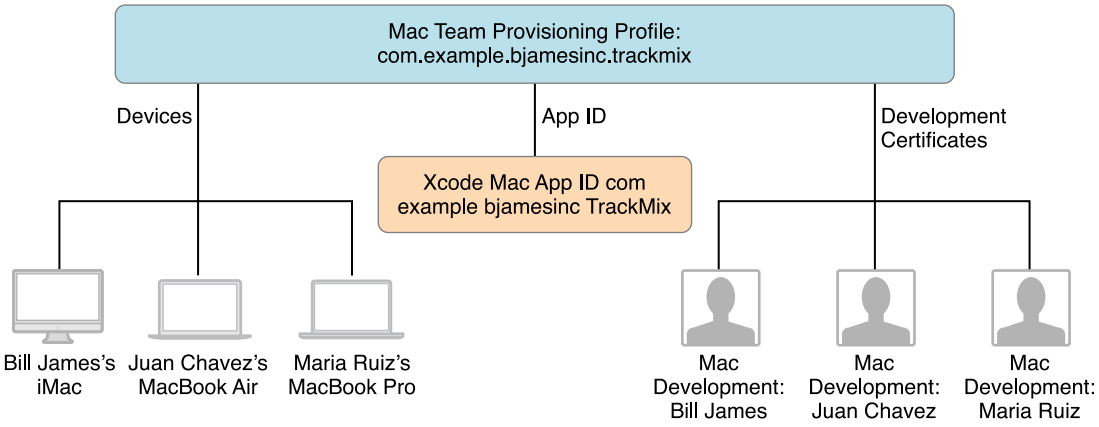
Xcode creates App IDs and team provisioning profiles as needed depending on the configuration and capabilities of your app. Xcode adds all of the devices and development certificates from all team members to the team provisioning profile. Thereafter, Xcode updates the team provisioning profile whenever you register a device, create a development certificate, or modify the App ID. (Changes you make to your team assets using Member Center don't automatically update team provisioning profiles.)

If your app can use a wildcard App ID during development, Xcode creates a team provisioning profile named *iOS Team Provisioning Profile: ** or *Mac Team Provisioning Profile: ** using a wildcard App ID it also creates. You can use a wildcard App ID with a subset of app services. However, if you add a capability that requires an explicit App ID—for example, iCloud, Game Center, or In-App Purchase—Xcode creates an explicit App ID and a corresponding team provisioning profile called *iOS Team Provisioning Profile:* or *Mac Team Provisioning Profile:* followed by the bundle ID. Because an explicit App ID exactly matches the project's bundle ID, you can register only one explicit App ID per bundle ID. Therefore, if one already exists in Member Center, Xcode uses it in the team provisioning profile instead of creating one for you.

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



This diagram shows a Mac Team Provisioning Profile using an explicit App ID for a Mac company.



Team agents need to approve team member development certificates and devices before they're added to the team provisioning profile. If you enrolled as a company, read Managing Your Team in Member Center (page 144) to learn more about your responsibilities.

Recap

In this chapter, you learned how to use the team provisioning profile, which Xcode creates and manages for you, to launch your app on devices. You learned what information Xcode needs to create the team provisioning profile, how to verify that Xcode registered your devices, and how to troubleshoot if Xcode displays errors or warnings.

Beta Testing iOS Apps

You've created your iOS app in Xcode and optionally added app services. You've tested your app on iOS Simulator and on your own devices. 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 users a preview of your next version.

There are three options to distribute your app for beta testing. Two prerelease options are managed through iTunes Connect. You upload your app to iTunes Connect using Xcode and then sign in to iTunes Connect to invite testers to download your app. Optionally, distribute your app to 25 of your iTunes Connect users (with either Technical, Admin, or Legal roles) or 1000 users per app specifying just email addresses. Distributing your app to users requires approval from App Review.

To distribute your prerelease build using iTunes Connect:

1. Create an iTunes Connect app record.
2. Update the build string.
3. Archive and validate your app.
4. Upload your app to iTunes Connect.
5. Manage testing in iTunes Connect.
6. Analyze crash reports and solicit feedback from testers.

If you include symbols when uploading your app to iTunes Connect, Apple collects and aggregates crash data on user devices for you. Later, you can view these crash reports in the Crashes organizer, described in Analyzing Crash Reports (page 114).

The third option is to distribute your app using an ad hoc provisioning profile that allows your app to run on up to 100 registered devices. The steps to use an ad hoc provisioning profile, described in Distributing Your App Using Ad Hoc Provisioning (page 104), are slightly different than the steps to upload your app to iTunes Connect. You can register up to 100 devices, but this limit includes devices you use for development and beta testing. If you are a member of the iOS Developer Enterprise Program, you don't have access to iTunes Connect, so use this method for beta testing your apps.

You should submit to the store the last archive you distribute for testing, so before you distribute your final candidate, review Apple guidelines and the Xcode project configuration, described in Prepare for Uploading (page 127).

Important: 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 on 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 that you support and have available. In addition, keep prior versions of iOS installed on devices for compatibility testing. If you don't support certain devices or iOS versions, indicate this fact in the project target settings in Xcode, as described in Setting Deployment Info (page 36).

Creating Your App Record in iTunes Connect

If you're 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're missing required app icons. You must have an iTunes Connect app record to validate and upload your app to iTunes Connect. To create your app record, read Creating an App Record (page 140) before continuing.

Updating the Build String

If you update the build string, as described in Setting the Version Number and Build String (page 35), iTunes recognizes that the build string changed and properly syncs the new version of your app to test devices.

Archiving and Validating Your App

You create an archive of your app regardless of the type of distribution method you select. Xcode archives allow you to build your app and store it, along with critical debugging information, in a bundle that's 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 later.

Important: Save the archive for every version and build of an app you distribute. If you upload your app to iTunes Connect, you'll need the archive and associated project to jump from a stack frame in the Crashes log to the source code in the debug navigator, described in Analyzing Crash Reports (page 114). You'll also need the debugging information to symbolicate crash logs you receive from other sources.

With archives, you can upload the same build you distribute for testing that you later submit to the store. It's important to test the exact build that's 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're testing the *exact* same build of your app that you submit to the store.

Follow these steps to archive and validate your app:

1. Review the Archive scheme settings.
2. Create an archive of your app.
3. Validate the archive.

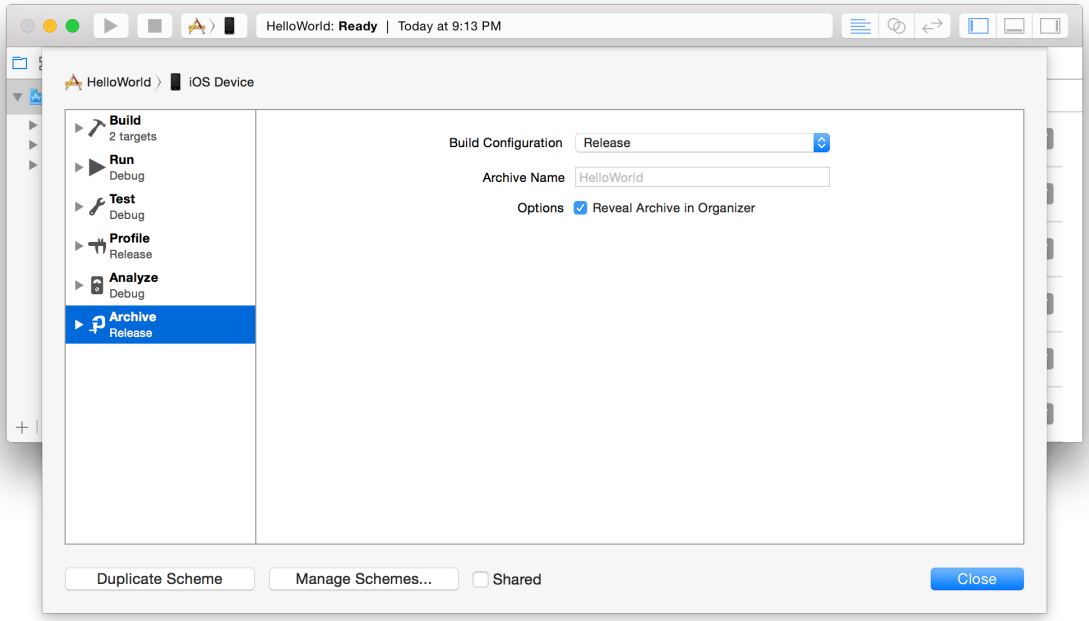
Reviewing the Archive Scheme Settings

The first time 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. From the Build Configuration pop-up menu, choose Release, and click Close.

Creating an Archive

Next, create an archive of your app. Xcode stores this archive in the Archives organizer.

To create an archive

1. In the Xcode project editor, choose iOS Device or your device name from the Scheme toolbar menu.

   You can't 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.

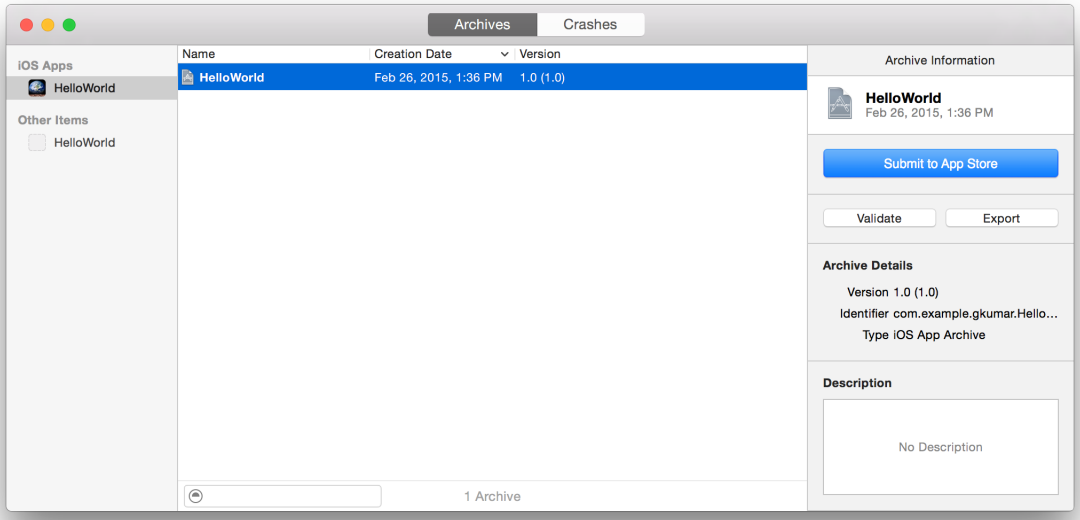
Xcode runs preliminary validation tests on the archive and may display a validate warning in the project editor. For example, if you are not using asset catalogs and don't set required app icons, as described in Setting Individual App Icon and Launch Image Files (page 42), an `Info.plist` warning message appears. If you see this warning, fix the issue and create the archive again.

Running iTunes Connect Validation Tests

Because iTunes Connect validates your archive when you upload it to iTunes Connect, it's best to validate your beta version now to discover issues early.

To validate an archive

1. In the Archives organizer, select the archive you want to validate, and click Validate.

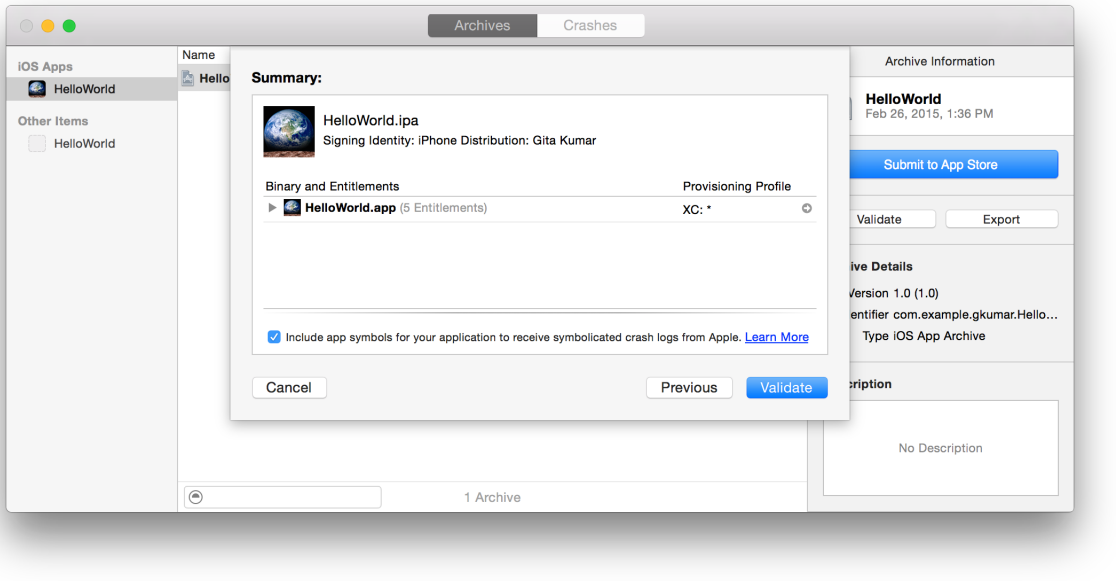


2. In the dialog that appears, choose a team from the pop-up menu and click Choose.

   If necessary, Xcode creates a distribution certificate and distribution provisioning profile for you. The distribution provisioning profile name begins with the text `XC:`.

3. In the dialog that appears, review the app, its entitlements, and provisioning profile, and click Validate.

Xcode uploads the archive to iTunes Connect, and iTunes Connect runs validation tests. If a dialog appears stating that no application record can be found, click Done, create an app record in iTunes Connect, and repeat these steps.



4. Review validation issues that are found, if any, and click Done.

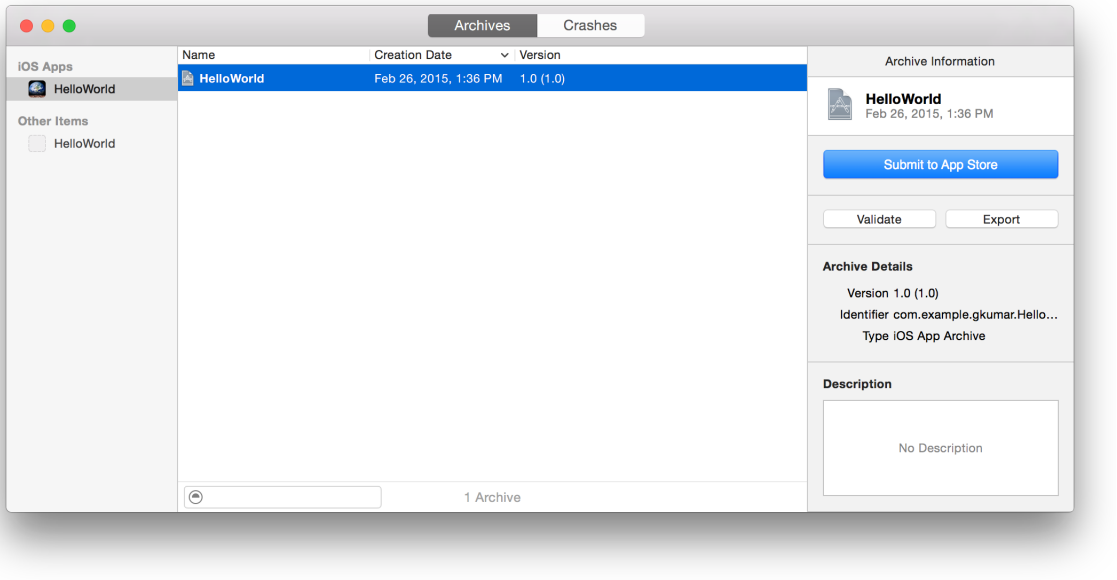
Fix any validation issues, create a new archive, and validate it again. Validation errors don't prevent you from uploading your app for beta testing.

Distributing Your Prerelease Build Using TestFlight

The TestFlight app allows testers to install your prerelease app on their iOS device. TestFlight also installs new versions as they become available and allows testers to send feedback directly to you. After you upload your app to iTunes Connect using Xcode, you use iTunes Connect to set up and invite both internal and external testers, as described in TestFlight Beta Testing (Optional).

To upload your app to iTunes Connect

1. In the Archives organizer, select the archive you want to upload, and click “Submit to App Store.”



2. In the dialog that appears, choose a team from the pop-up menu and click Choose.

   If necessary, Xcode creates a distribution certificate and distribution provisioning profile for you. The name of the distribution provisioning profile begins with the text `XC:`.

3. In the dialog that appears, review the app, its entitlements, and provisioning profile.

4. To receive crash reports from Apple, check “Include app symbols for your application…”.

   Later, you can view symbolicated crash logs in the Crashes organizer, described in Analyzing Crash Reports (page 114).

5. Click Submit.

   Xcode uploads the archive to iTunes Connect and iTunes Connect runs validation tests. If a dialog appears stating that no application record can be found, click Done, create an app record in iTunes Connect, and repeat these steps.

6. If issues are found, click Done and fix them before continuing.

   For a complete list of issues you should fix before distributing your app, read *App Store Review Guidelines for iOS Apps*. For more information on the app review process, go to App Review.

7. If no issues are found, click Submit to upload your app.

After you upload the app, use iTunes Connect to select beta distribution options, described in TestFlight Beta Testing (Optional). You specify whether you want to distribute your app to internal or external testers.

Distributing Your App Using Ad Hoc Provisioning

Alternatively, distribute the app for testing on registered devices using an ad hoc provisioning profile. Using this method, testers don't need to be team members or iTunes Connect users to run the app, but their devices need to be registered in Member Center. You can register up to 100 devices per year that your team can use for development and testing. Therefore, choose this method if you can use a portion of these devices for testing and can collect device IDs from testers. Also choose this method if you're not ready to create an app record in iTunes Connect. You don't need to validate or upload your app to iTunes Connect to distribute it using an ad hoc provisioning profile.

The steps are similar to TestFlight beta testing, except that you register test devices, and instead of uploading your app to iTunes Connect, you export an archive as an *iOS App* (a file with a `.ipa` filename extension) and distribute it for testing:

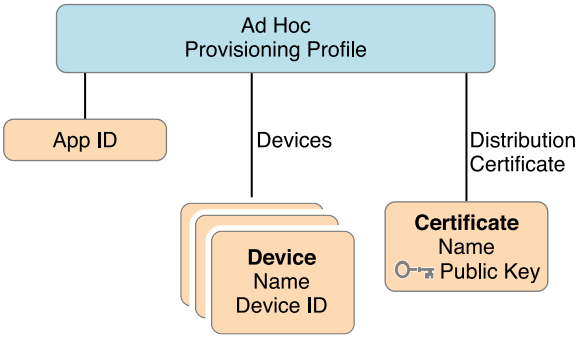
1. Optionally, create an iTunes Connect app record.
2. Register all test devices.
3. Update the build string.
4. Archive and validate your app.
5. Export the archive as an iOS App.
6. Install the app on test devices.
7. Solicit crash reports from testers.

To update the build string, read Updating the Build String (page 98), and to archive and optionally validate your app, read Archiving and Validating Your App (page 98). This section contains the steps that are different for ad hoc distribution.

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 to use app services without the assistance of Xcode. It's one of the two types of distribution provisioning profiles that 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 aren't copied and distributed without your knowledge.

When you're 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 per membership year for development and testing.

Sign the iOS App file using the distribution certificate specified in the ad hoc provisioning profile, and distribute it to testers.

Registering Test Devices

Register one or more test devices before you create an ad hoc provisioning profile. To register test devices, collect device IDs from testers and add them to Member Center.

Testers can get their device ID using iTunes. (They don't need to install Xcode to do this.) Send the instructions in Locating iOS Device IDs Using iTunes (page 192) to testers and ask them to send their device IDs to you, or follow these steps to collect your own device IDs.

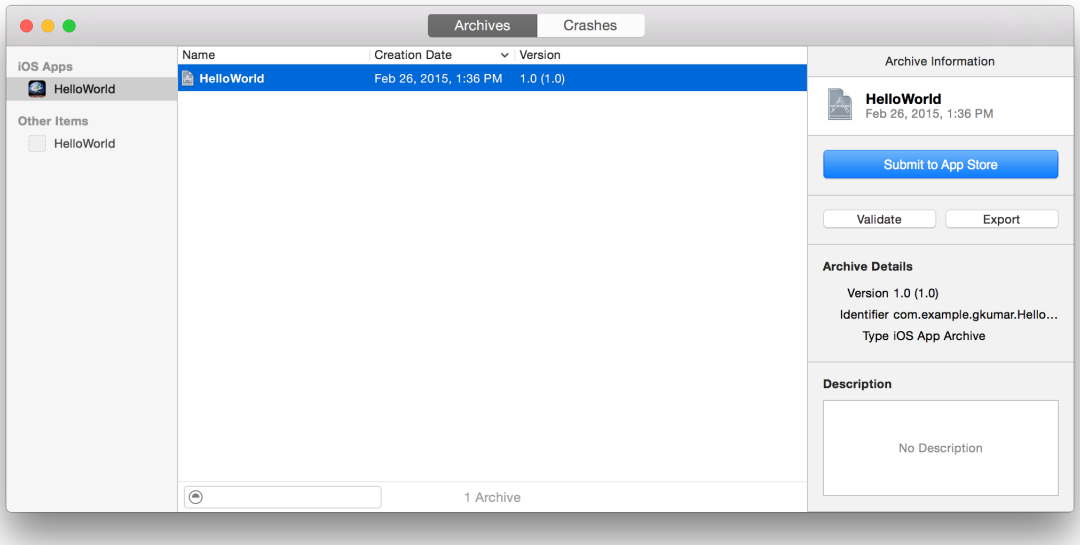
In Member Center, register one or more devices, as described in Registering Devices Using Member Center (page 191).

Creating an iOS App File for Ad Hoc Deployment

Because testers don't have Xcode to run your app, you create an iOS App file that they can then use to install your app on their device. Use Xcode to create an archive, and from the Archive organizer export an iOS App file.

To create an iOS App file for ad hoc deployment

1. In the Archives organizer, select the archive.

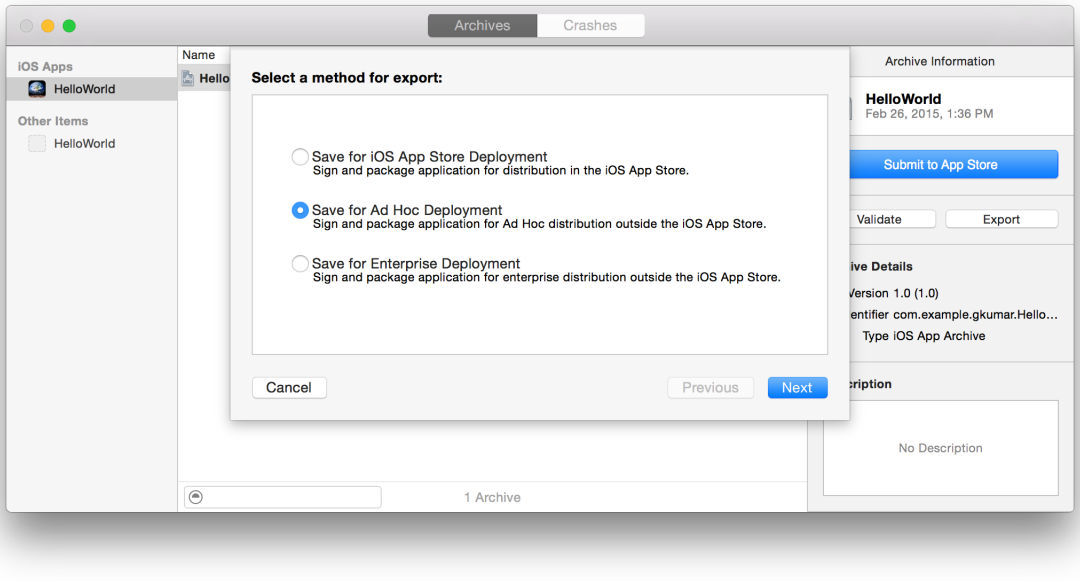


2. Optionally, click the Validate button.

   If you're close to submitting your app to the store, validate it now and fix any problems before distributing it for final testing. For a complete list of issues that should be fixed before you distribute your app, read *App Store Review Guidelines for iOS Apps*. For more information on the app review process, go to App Review. Click the Next button to continue.

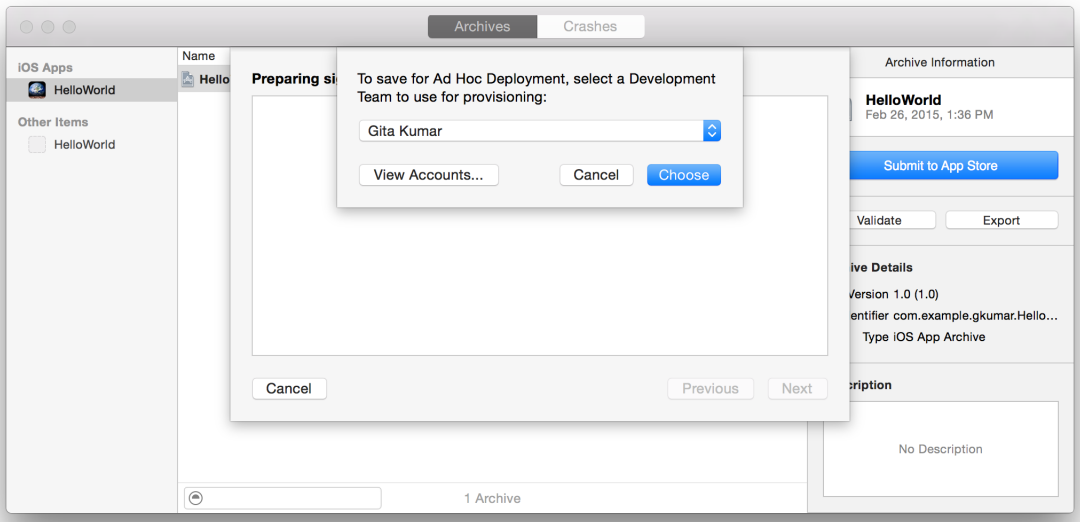
   Note: If you belong to the iOS Developer Enterprise Program, skip this step.

3. Otherwise, click the Export button, select “Save for Ad Hoc Deployment,” and click Next.



4. In the dialog that appears, choose a team from the pop-up menu and click Choose.

   If necessary, Xcode creates a distribution certificate and an ad hoc provisioning profile for you.



5. In the dialog that appears, review the app, its entitlements, and the provisioning profile, and click Export.

   The name of the ad hoc provisioning profile begins with the text `XC Ad Hoc:`.

6. Enter a filename and location for the iOS App file, and click Export.

   The file has an `.ipa` extension.

Installing Your App on Test Devices

Before you distribute your app to testers, follow the same 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.

To install the app on a device

1. Connect the testing device to a Mac running iTunes.

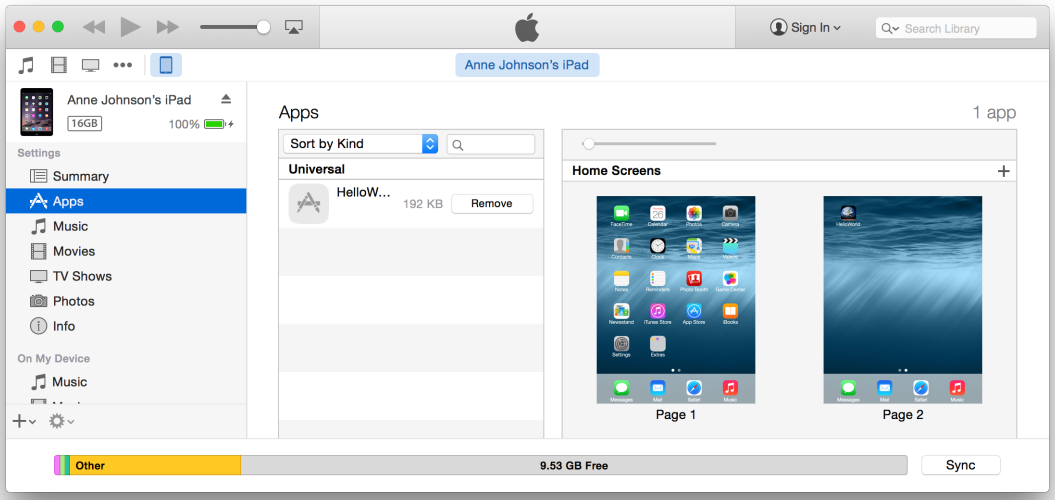
   If possible, don't use a Mac that you use for development.

2. Double-click the iOS App file that you created earlier (the file with the `.ipa` extension).

3. In iTunes, click the device in the upper-left corner of the window.

4. Click the Apps button.

   The app appears in the iTunes Apps list.



5. Under Apps, choose "Sort by Name" or "Sort by Kind" from the pop-up menu.

   An Install or Remove button appears adjacent to the app.

6. If an Install button appears, click it.

   The button text changes to Will Install.

7. Click the Apply button or the Sync button in the lower-right corner to sync the device.

   The app is uploaded to the device so that the user can start testing.

Finally, send the iOS App file to testers, along with the app installation instructions and the crash report instructions, as described in Soliciting Crash Reports from Testers.

Distributing Your App Using the Xcode Service

After you perform the steps in this chapter to create an ad hoc provisioning profile and export your app for beta testing, you can optionally set up an Xcode service to distribute your app. The Xcode service, available in OS X Server, automates the integration process of building, analyzing, testing, and archiving your app. The Xcode service also hosts a website that facilitates the distribution of product builds and archives to testers and other team members. See *Xcode Server and Continuous Integration Guide* for more information about using the Xcode service to distribute versions of your app to testers.

Copying App Sandbox Data

To further diagnose an issue with your app, you can view, download, and replace the contents of an app container on an iOS device, as described in Viewing, Downloading, and Replacing App Containers on iOS Devices (page 198).

Soliciting Crash Reports from Testers

When an app crashes on a device, iOS creates a report 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've sent.

To send crash reports from Windows

1. In the Windows search field, enter the crash log directory for your operating system, 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:

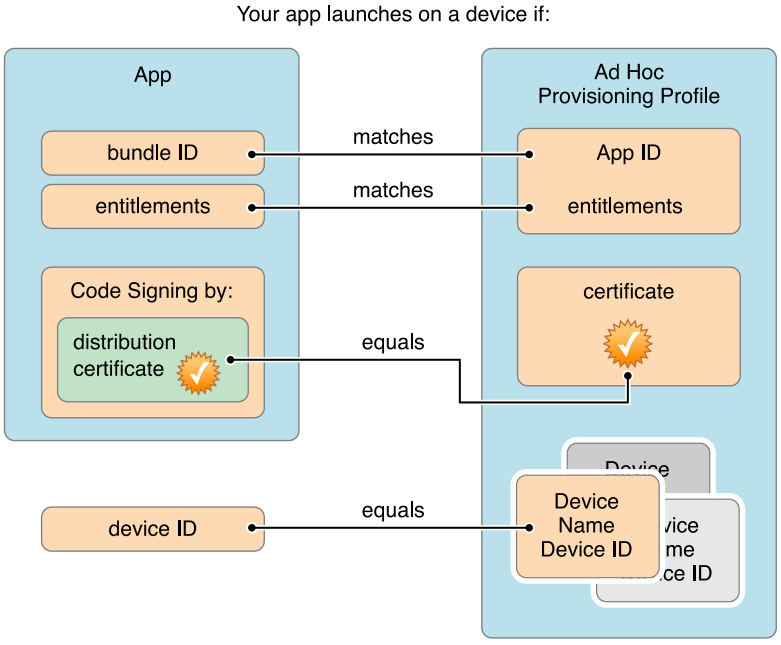
     ```
     C:\Documents and Settings\<user_name>\Application Data\Apple computer\Logs\CrashReporter
     ```

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 that you receive from testers, read Viewing and Importing Crashes in the Devices Window (page 123).

Ad Hoc Provisioning Profiles in Depth

You signed the iOS App file using the distribution certificate specified in the ad hoc provisioning profile. The ad hoc provisioning profile was included in the app bundle when you built and archived the app. Then you installed the app on a 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 using three different methods. You learned how to upload an archive of your app to iTunes Connect for distribution using TestFlight to internal or external users. You also learned how to distribute your app on designated test devices using ad hoc provisioning profiles. You also received instructions to send to ad hoc testers to install the beta version of your app and send crash reports to you. If you use TestFlight to distribute your prerelease app, Apple distributes the app and collects crash logs for you.

Beta Testing Mac Apps

Mac apps can be configured to launch outside Xcode on designated test devices using the team provisioning profile. The team provisioning profile uses the same pool of devices that you register for development, so you are limited to 100 combined development and test Mac computers per year.

To distribute your app to beta testers using the team provisioning profile:

1. Register all test devices.
2. Archive and export the app using the team provisioning profile.
3. Install the app on test devices.

Registering Test Devices

Register all the devices you want to use for testing with Member Center before exporting your app using Xcode. Xcode automatically registers your development Mac before it creates a team provisioning profile. To register additional test Mac computers, collect device IDs from testers and add them to Member Center.

To add multiple test Mac computers to the team provisioning profile, read Registering Devices Using Member Center (page 191). After you add devices, regenerate the team provisioning profile by refreshing profiles in Xcode, described in Refreshing Provisioning Profiles in Xcode (page 80).

Distributing Your App Using the Team Provisioning Profile

When you export a Mac app from Xcode, it embeds the team provisioning profile in the bundle and code signs the app with your development certificate. This allows the exported app to launch on all registered devices.

> **Important:** If you configured your Xcode project to use a custom provisioning profile, configure the project to use the team provisioning profile before performing these steps, as described in Using Xcode-Managed Provisioning Profiles (page 209). Otherwise, your app may not launch on test devices.

To export a development certificate signed app using the team provisioning profile

1. Choose a target from the Scheme toolbar menu, and click Run.

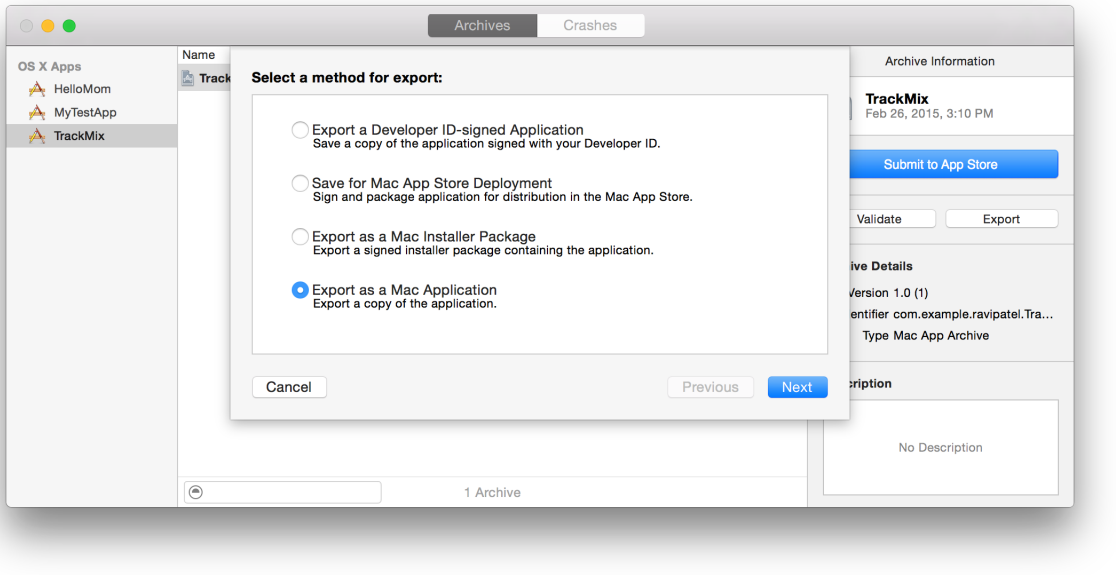
Always test the app before you export it.

2. Choose Product > Archive.

   The Archives organizer appears and displays the new archive.

3. In the Archives organizer, select the archive and click Export.

4. Select “Export as a Mac Application” and click Next.



5. Enter a filename in the Export As text field, and click Export.

Installing Your App on Test Devices

Before you distribute your app to testers, launch the app on another registered Mac that you don’t use for development. The app will launch only on devices specified in the team provisioning profile. If you can’t launch the app on a designated device because it’s from an unidentified developer, bypass the security settings in OS X to launch the app.

To open an app from an unidentified developer

1. In the Finder, Control-click the app icon, and choose Open from the shortcut menu.

2. In the Gatekeeper dialog that appears, click Open.

Analyzing Crash Reports

After you distribute your iOS app for testing by using TestFlight or after you make it available on the App Store, routinely download and analyze crash reports. When an app crashes, the system creates a crash log that describes the conditions under which the app terminated, in most cases including a complete stack trace for each executing thread. Apple provides a crash report service for iOS apps that collects and aggregates user crash logs.

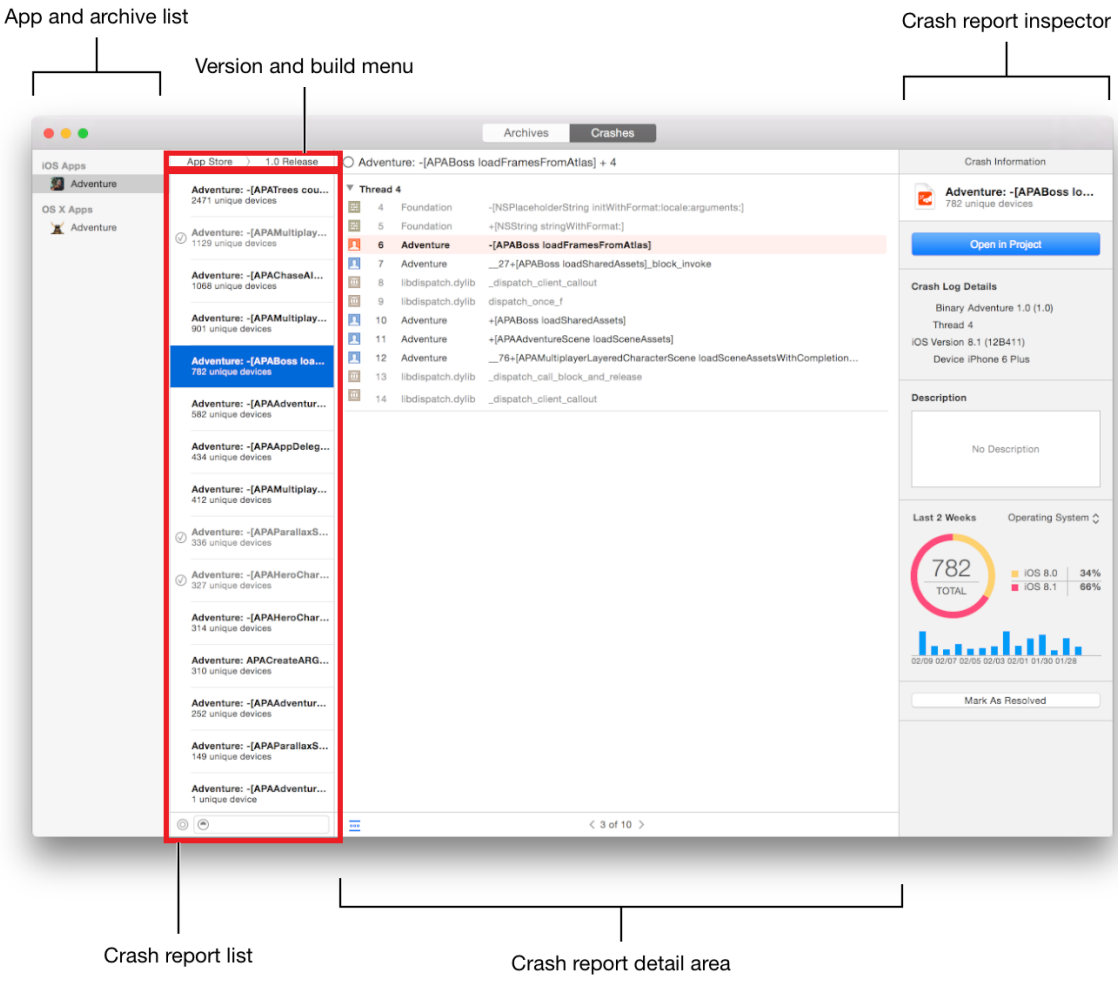
Alternatively, view crash logs directly on iOS devices or import crash reports, as described in Viewing and Importing Crashes in the Devices Window (page 123). For example, if you distribute your iOS and Mac app outside of the store, you can solicit crash reports directly from users and import them into Xcode.

About the Crash Report Service

Apple provides a service that allows you to download crash reports for the most recent iOS app versions and builds that you distribute using TestFlight or the App Store. The service collects and groups similar crash logs into crash reports. Each build of an app may have multiple crash reports and each crash report may contain multiple crash logs.However, crash logs are not sent to Apple unless the user agrees to share crash data with app developers. TestFlight users automatically agree to share crash data. The service does the following to generate crash reports:

- Collects crash logs from apps running on user devices
- Symbolicates the crash logs (replaces memory addresses with human-readable names)
- Compares stack traces to identify and group similar crash logs into crash reports
- Removes all personal user data from crash logs
- Provides the total number of unique devices where the crash occurred
- Provides a sample set of crash logs for each crash report
- Updates crash reports daily

Xcode displays information about crash reports in the Crashes organizer:



Before Viewing Crash Reports

Perform all these steps to fully enable the crash report service. If you previously submitted an app with symbols to iTunes Connect and distributed it using TestFlight or the App Store, you may already see crash reports in the Crashes organizer. If you do not see crash reports in the the Crashes organizer, verify that you have performed these steps.

	Step
☑	In Xcode, to upload your app to iTunes Connect, enter an Apple ID in Accounts preferences that belongs to the iOS Developer Program, as described in Adding Your Apple ID Account in Xcode (page 20).

	Step
☑	Set the version and build number, as described in Setting the Version Number and Build String (page 35). If you are distributing another build of your app, increment the build string.
☑	Create a single app archive and upload it to iTunes Connect with symbols, as described in Distributing Your Prerelease Build Using TestFlight (page 102) and Submitting Your App to the Store (page 126). (Ensure that the "Include app symbols for your application…" box is checked before you click Submit.) Do not delete archives that you upload to iTunes Connect.
☑	In iTunes Connect, distribute the app to testers using TestFlight, as described in TestFlight Beta Testing (Optional), or submit the app for review, as described in Submitting the App to App Review.
☑	On devices, users who download your app from the store need to agree to share crash data with app developers, as described in Sharing Crash Data with App Developers (page 122). (TestFlight users automatically agree to share crash data.)
☑	In Xcode, to view crash reports, enter an Apple ID that is a team agent or admin, or an iTunes Connect user with Admin or Technical role, as described in Adding Your Apple ID Account in Xcode (page 20).
☑	In Xcode, to go from a stack frame in the Crashes organizer to the source code in debug navigator, open the Xcode project that you archived and uploaded to iTunes Connect.

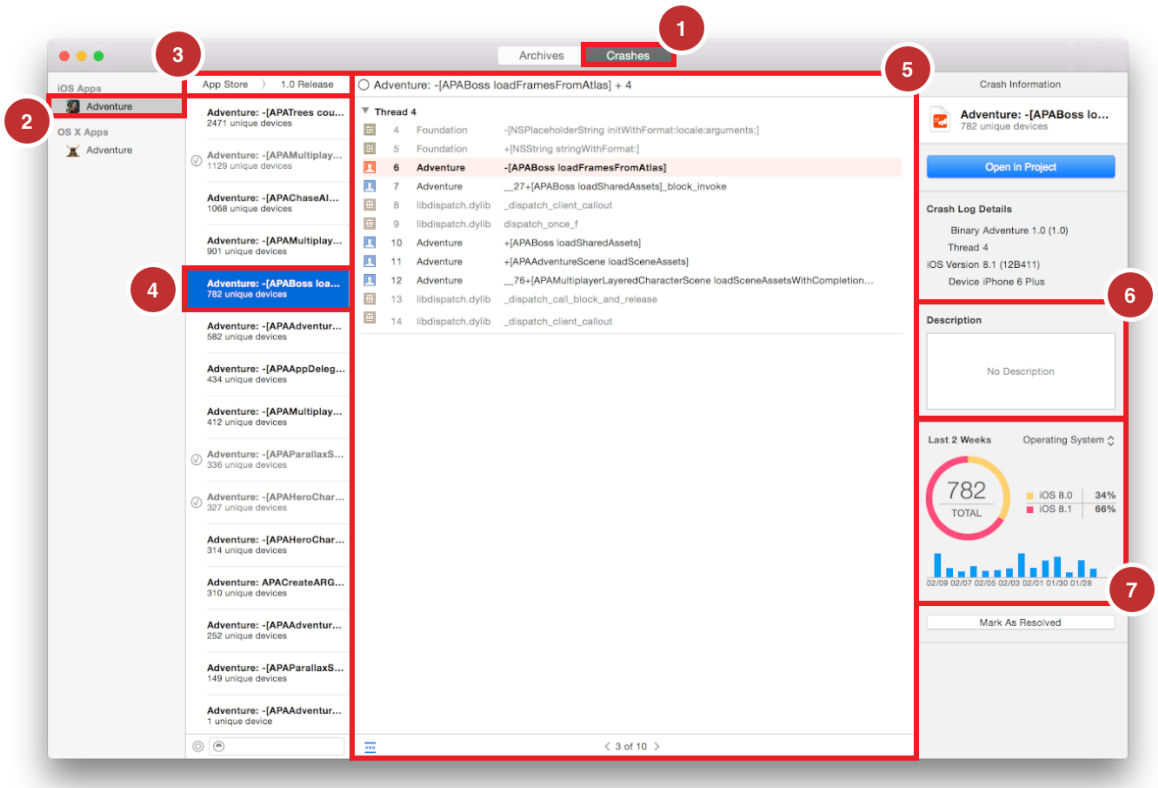
Also, allow up to three days between when you first distribute your app and when crash reports appear in Xcode. To maintain a good user experience, crash data is sent from user devices to Apple when the user allows.

Viewing Crash Reports in the Crashes Organizer

The Crashes organizer displays crash reports for all the apps developed by all your teams.

To view crash reports

1. In the organizer, click Crashes.



2. In the left column, select an app or local archive from the list.

   The list of apps is obtained from iTunes Connect and includes information about every version and build you uploaded to iTunes Connect.

3. At the top of the second column, choose a version and build from the pop-up menu.

   Xcode begins refreshing the crash reports for the version and build you select. Xcode downloads the top 25 crash reports—crash reports with the most number of occurrences on unique devices—that occurred during the past two weeks.

4. In the second column, select a crash report.

   The title of the crash report defaults to the stack frame where the crash likely occurred. The number of unique device occurrences of the crash appears below the crash title.

5. In the detail area, view the crash logs.

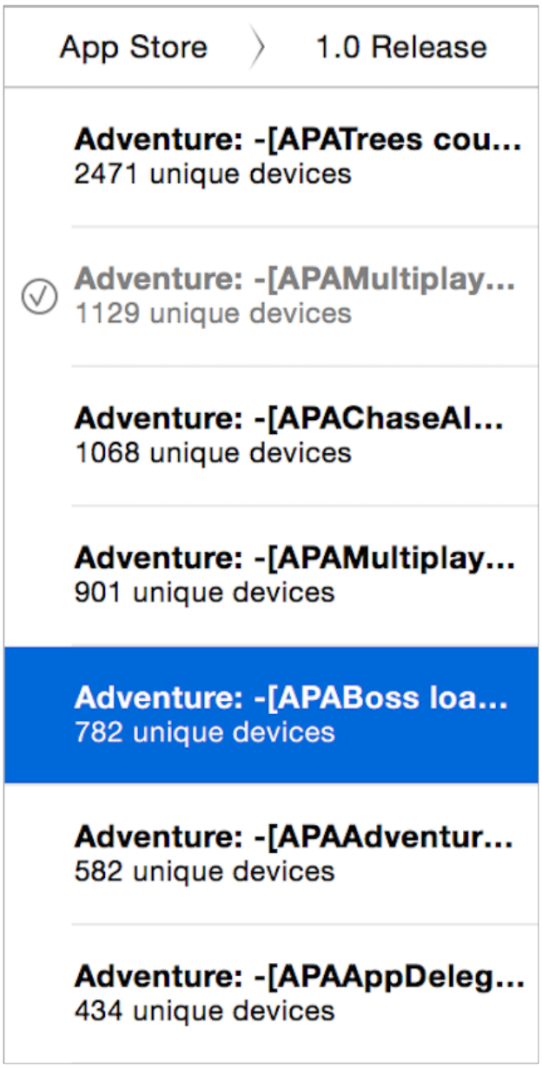
   The detail area displays the stack trace for each thread. The stack frame where the crash occurred is highlighted in orange.

6. In the inspector, add information and view statistics about a crash report.

Viewing and Finding Crash Reports

In the crash report list, perform these actions:

- To search for crash reports, enter a string in the search field at the bottom.
- To view the crash logs of a crash report in the Finder, Control-click a crash report and choose Show in Finder.



In the detail area, perform these actions:

- To view the previous or next crash log of a crash report, click the page arrows (< or >) in the footer, or click the page count (for example, `11 of 20`) and choose a page from the pop-up menu.

- To collapse or expand the crash log, click the filter icon ( ) in the lower-left corner in the footer. Collapse the crash log to view only the stack frames in your app. Expand the crash log to view all the stack frames including framework and system stack frames.

Editing Information About Crash Reports

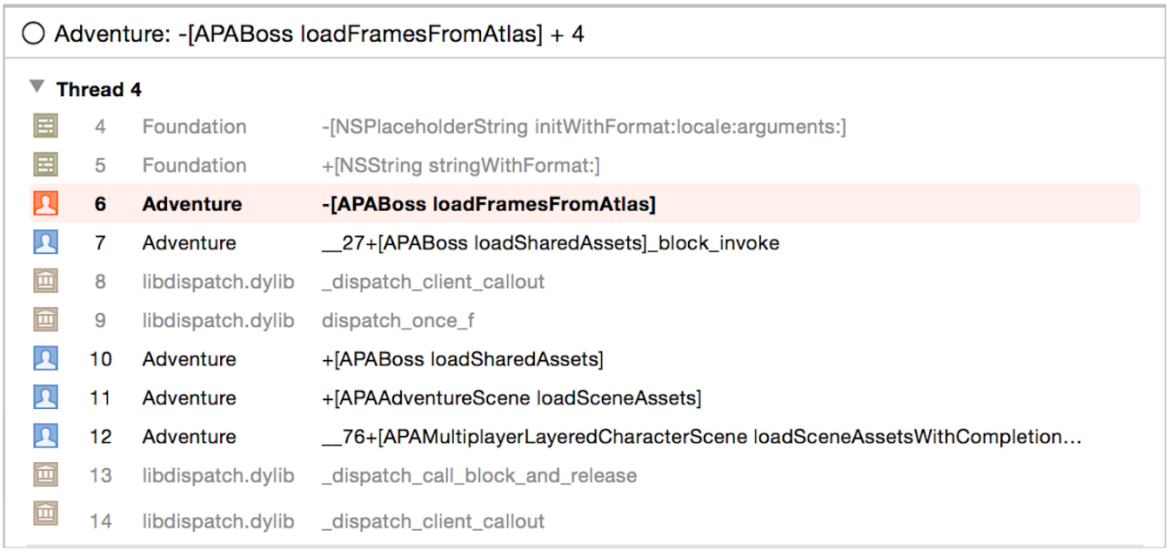
As you resolve issues, you can store information about crash reports locally.

In the crash report list, perform these actions:

- To mark a crash report as resolved, select the circle next to the crash report or at the bottom of the inspector, and click “Mark as Resolved.”
- To show or hide resolved crash reports, select the circle ( ) in the footer next to the search field.

In the detail area, perform this action:

- To change the name of a crash report, place the insertion point in the header and edit the text.



In the inspector, perform this action:

- To add notes about a crash report, enter the notes in the Description text field.

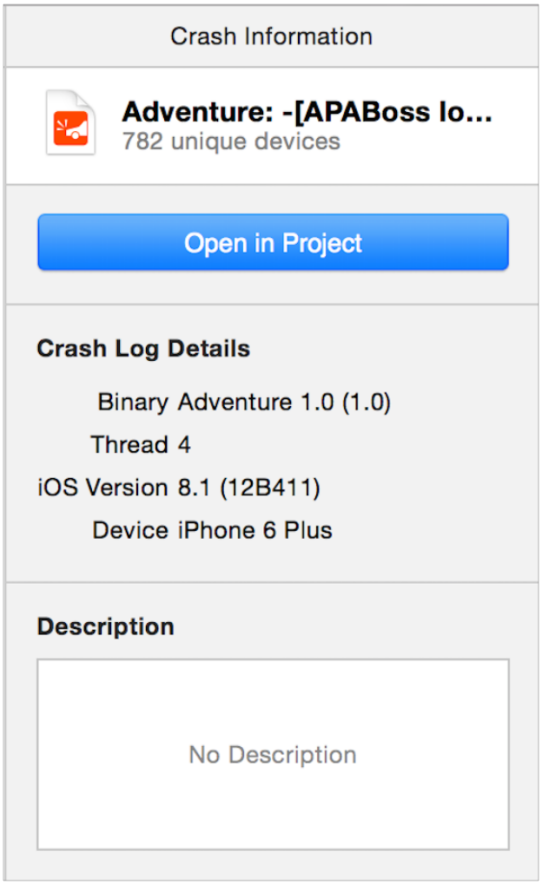
Opening the Source Code in the Debug Navigator

In the detail area, perform these actions:

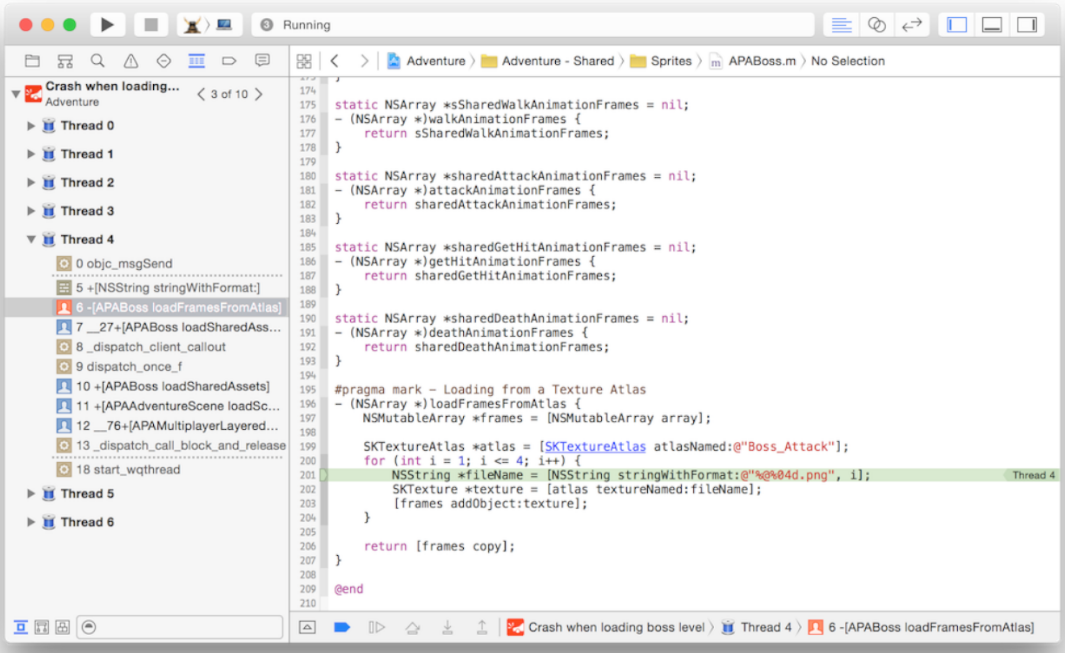
- To go to the source code for a stack frame, hover over the stack frame and click the arrow that appears to the right.
- To go to the source code where the crash occurred, click the arrow that appears when you hover the pointer over the stack frame that is highlighted.

In the inspector, perform this action:

- To go to the source code for a stack frame, click “Open in Project.”

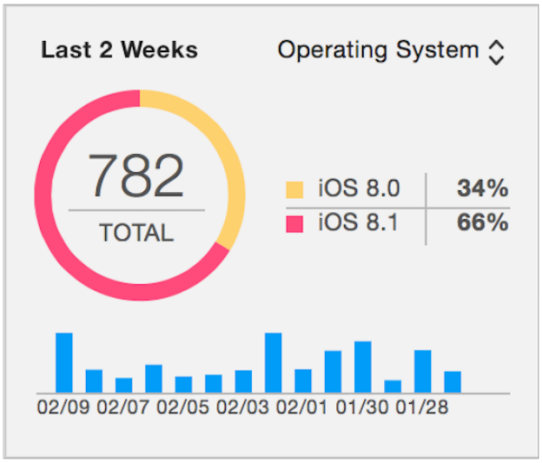


Xcode opens the associated project and displays the line of code in the debug navigator.



Viewing Statistics About Crash Reports

View the crash data over the past two weeks by device type, operating system, and date. To change the device type, choose Device Type from the pop-up menu. To change the operating system, choose Operating System from the pop-up menu.



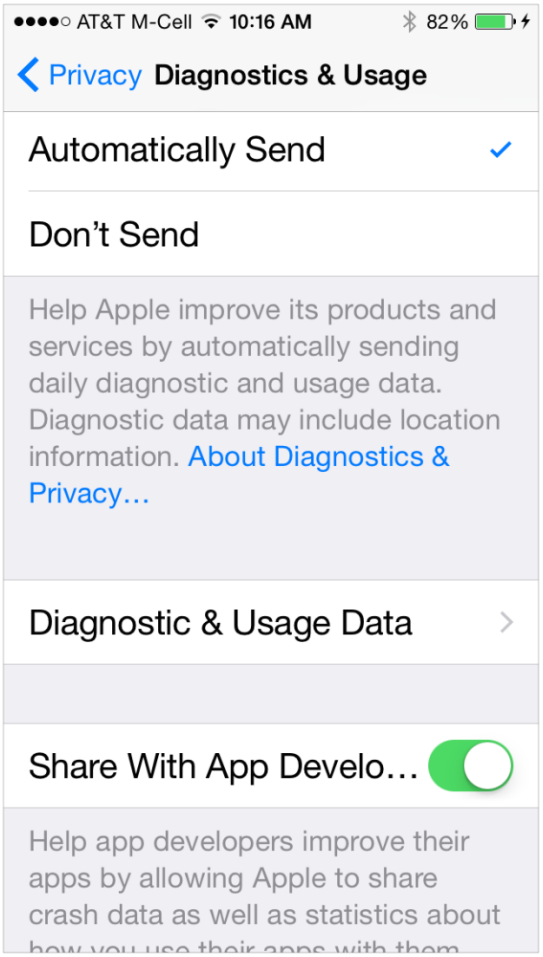
Sharing Crash Data with App Developers

Apple can't provide crash reports if users don't share the crash data with app developers. If a user reports a crash and you don't have a corresponding crash report, ask the user to share the crash data with app developers. Crash data is automatically sent to Apple for TestFlight apps but not apps the user downloads from the App Store.

Give these instruction to users. On both iOS and OS X, the crash data option is in the Diagnostics & Usage section of the Privacy settings.

To share crash data with iOS developers

1. In Settings, tap Privacy.
2. Scroll to the bottom and tap Diagnostics & Usage.

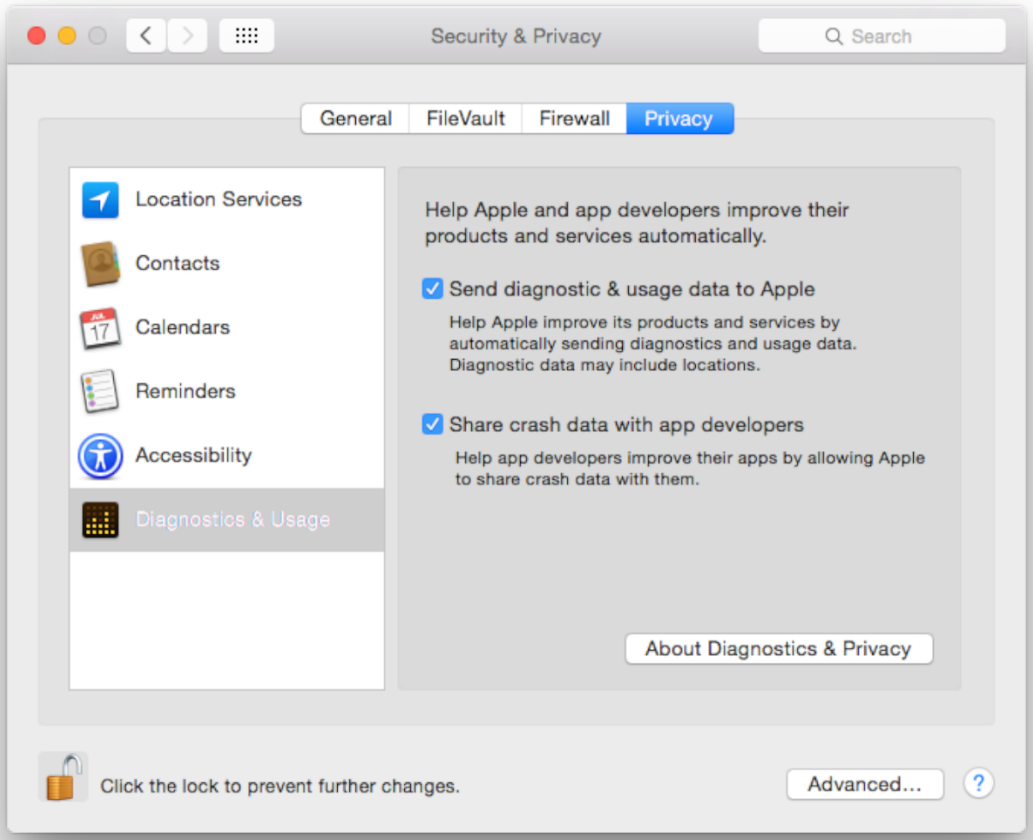


3. If necessary, tap the Share With App Developer switch to enable it.

Similarly, ask Mac users to share their crash data with app developers.

To share crash data with Mac developers

1. In System Preferences, click Security & Privacy.
2. Click Privacy and in the left column, click Diagnostics & Usage.



3. Select "Share crash data with app developers."

   To enable the Privacy options, click the lock icon. In the dialog that appears, enter the credentials for a system account with admin privileges and click Unlock.

Viewing and Importing Crashes in the Devices Window

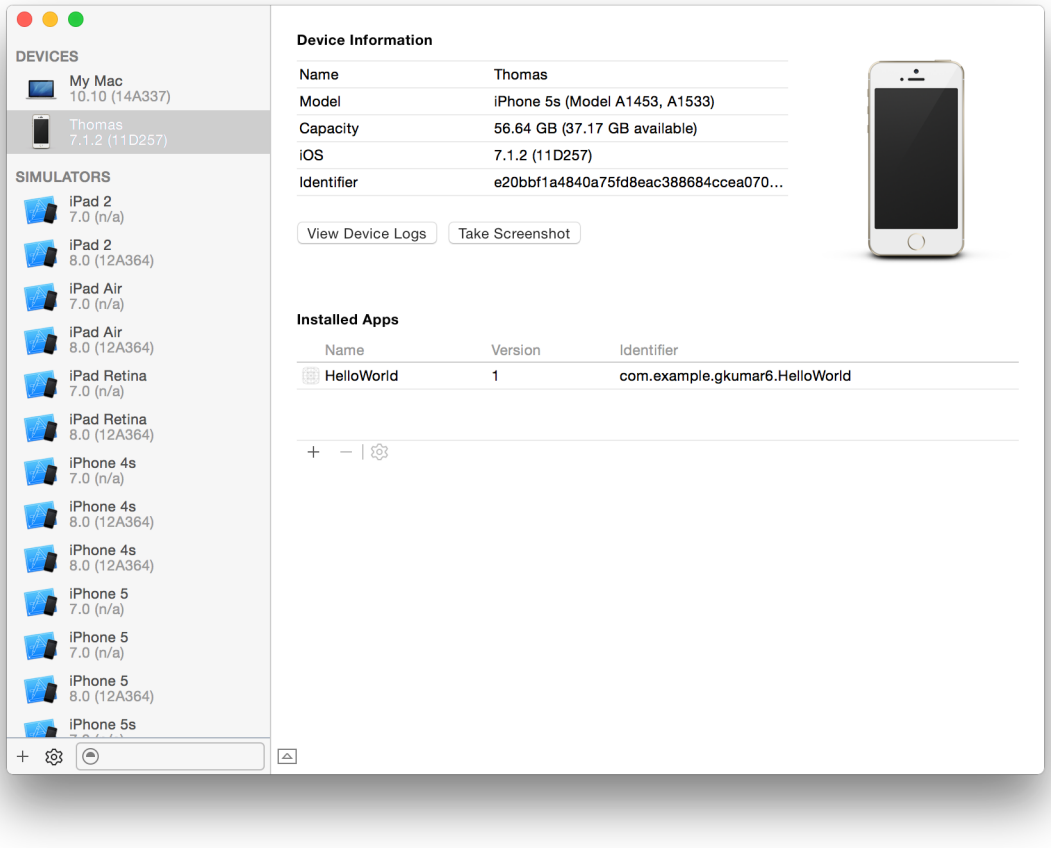
In the Devices Window, you can view details about a crash log directly on a device or import crash reports you receive from other sources. If you keep the archive of your app (that includes the matching binary and `.dSYM` file), Xcode automatically *symbolicates* imported crash reports. Xcode replaces the memory addresses in the crash log with human-readable function names and line numbers. Then you can view the stack trace for each execution thread.

For this reason, do not delete archives that you export and distribute for testing. For example, if you distribute your iOS app using ad hoc provisioning (Distributing Your App Using Ad Hoc Provisioning (page 104)) or Mac app using a team provisioning profile (Beta Testing Mac Apps (page 112)), you can import unsymbolicated crash reports you receive from testers. You can also download unsymbolicated crash reports from iTunes Connect for apps released through the App Store or Mac App Store.

> **Note:** The crash reports you download from iTunes Connect may be a different set of crash reports that you view in the Crashes organizer. If you uploaded your iOS app to iTunes Connect with symbols, view crash reports in the Crashes organizer instead.

To view device crash logs or import a crash report

1. Choose Window > Devices.
2. If necessary, connect your iOS device to your Mac.
3. Select the device under Devices, and click the View Device Logs button.



A sheet appears displaying the crash logs on the device.

4. To view a crash log, select it in the left column.

View the crash log in the detail view on the right.

5. To import a crash report, drag the crash report from the Finder to the left column of the sheet.
6. Click Done.

Reproducing Crashes in Xcode

Make sure to test the *exact* same build that crashed. Save all the archives that you distribute for testing or upload them to iTunes Connect. Verify that the archive in Xcode matches the crash report, as described in *How to Match a Crash Report to a Build*. Follow these same steps to determine if you're testing the same build that you submitted to the store.

For Mac apps, to reproduce a crash, use a guest account with a fresh install of the version of OS X that matches the crash report. Don't attempt to recreate a crash using a developer or admin system account, because the problems you want to analyze may not occur.

Submitting Your App to the Store

The next distribution step is to submit your app to the App Store or Mac App Store. Read Prepare for Uploading (page 127) to ensure that you perform all the steps to streamline the app review process. It's recommended that you submit the last archive you distribute for testing. The last build you test should be product quality and pass iTunes Connect validation tests. Depending on the testing method you choose, follow the steps in this chapter to submit your app.

If you use TestFlight to distribute your iOS app to testers (described in Beta Testing iOS Apps (page 97)), submit the last build directly to App Review using iTunes Connect, as described in Submitting the App.

For iOS apps you distribute for testing using ad hoc provisioning (described in Distributing Your App Using Ad Hoc Provisioning (page 104)), and Mac apps you distribute for testing using the team provisioning profile (described in Beta Testing Mac Apps (page 112)), follow these steps:

1. If necessary, validate the last archive you distributed for testing, described in Running iTunes Connect Validation Tests (page 130).
2. For Mac apps, test the Mac Installer Package, as described in Testing the Mac Installer Package (page 131).
3. Upload the archive, as described in Uploading Your App Using Xcode (page 133).
4. Use iTunes Connect to submit your app to App Review, as described in Submitting the App.

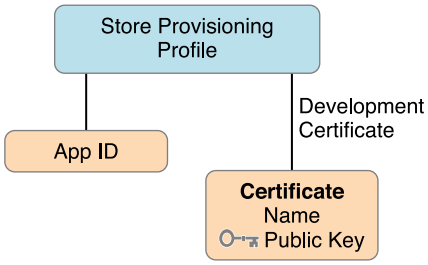
If you want to create an archive just before uploading your app (for example, if you changed the build number), follow the steps in Creating an Archive (page 129) before you validate the archive.

The first time you upload your app to iTunes Connect, Xcode creates the necessary distribution certificate and store provisioning profile for you.

About Store Provisioning Profiles

A *store provisioning profile* is a distribution provisioning profile that authorizes your app to use certain app services and ensures that you are the one submitting your app. A store provisioning profile contains a single App ID that matches one or more of your apps and a distribution certificate. You configure the App ID indirectly through Xcode to use certain app services. You enable and configure app services by setting entitlements and

performing other configuration steps. 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, Xcode signs the app bundle with the distribution certificate referenced in the store provisioning profile.



Prepare for Uploading

Before uploading your app to iTunes Connect, review the human interface and store guidelines, enter information in iTunes Connect, and verify your Xcode project settings.

Review Human Interface and Store Guidelines

To streamline the approval process, review the following guidelines and fix any problems and retest your app 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*.

For more information on the app review process, go to App Review.

Enter Information in iTunes Connect

iTunes Connect is a web tool you use to enter information about your app for sale in the App Store or the Mac App Store. iTunes Connect stores all the metadata about your app including the versions and builds that you upload using Xcode. Before you upload your app, create an app record in iTunes Connect, as described in Creating an iTunes Connect Record for an App. Then enter all the required information, described in Viewing and Changing Your App's Metadata.

Verify Your Xcode Project

Verify that your Xcode project is configured correctly:

- Review your Xcode project configuration. Read Configuring Your Xcode Project for Distribution (page 23).

- The bundle ID in the Xcode project, described in Setting the Bundle ID (page 27), should match the bundle ID you enter in iTunes Connect.
- Use the same App ID to code sign your archive as you used for development and testing.

  If you don't use any app services that require an explicit App ID, you can use the Xcode wildcard App ID. If you want to create a new App ID, read Registering App IDs (page 184). However, if you change your App ID, retest your app using the new App ID before uploading it to iTunes Connect.
- Review the version number and build number settings, described in Setting the Version Number and Build String (page 35). iTunes Connect extracts its prerelease version number and build number from the binary.
- To ensure that your app enables the key app services you want to use, review your App ID settings. Read Adding Capabilities (page 48).

For Mac apps, review these additional tasks:

- Select Mac App Store as the signing method during development, as described in Choosing a Signing Identity (Mac Only) (page 29).
- All apps need to have App Sandbox enabled, as described in Configuring App Sandbox (Mac Only) (page 50).
- 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 the helper app.

Archiving and Validating Your App

Before you upload your app to iTunes Connect, you create a signed archive of your app and validate it. Test your final build before uploading it, 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. Review the Archive scheme settings.
2. Create an archive of your app.
3. Validate the archive.
4. If necessary, test your archive before uploading it.

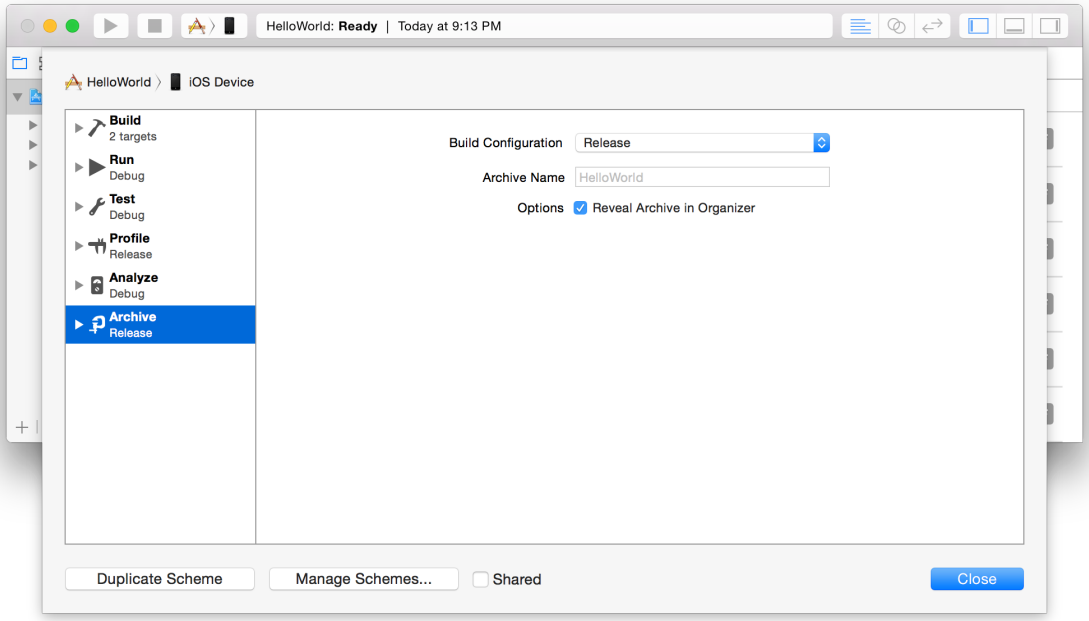
Important: Archives allow you to build your app and store it, along with critical debugging information, in a bundle that's managed by Xcode. Save an archive for any version of an app you distribute to users. You'll use the debugging information stored in the archive to decipher crash reports later.

Reviewing the Archive Scheme Settings

If necessary, 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 Close.

Creating an Archive

No matter what method you choose to distribute your app, create an archive first. Before creating the archive, build and run your app one more time to ensure that it's 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 can't 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.

Xcode runs preliminary validation tests when you create the archive. If a validate warning message appears in the project editor, fix the issue and create the archive again.

Running iTunes Connect Validation Tests

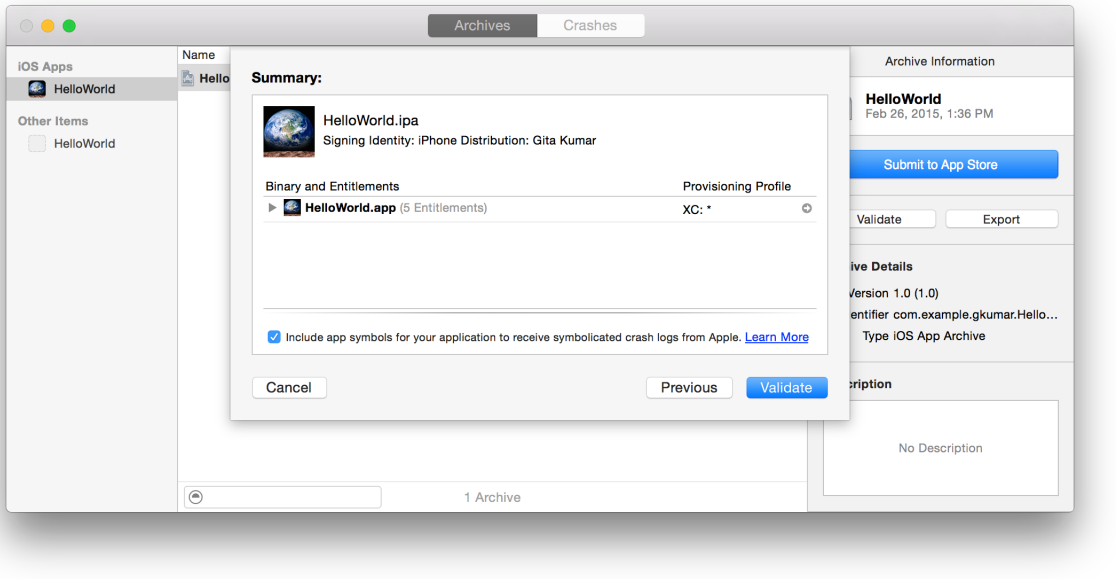
Immediately after creating the archive, upload your archive to iTunes Connect and run validation tests.

To validate an archive

1. In the Archives organizer, select the archive.
2. Click the Validate button.
3. For Mac apps, select "Validate for the Mac App Store" as the validation method and click Next.
4. In the dialog that appears, choose a team from the pop-up menu and click Choose.

   If necessary, Xcode creates a distribution certificate and distribution provisioning profile for you. The name of the distribution provisioning profile begins with the text `XC:`.
5. In the dialog that appears, review the app, its entitlements, and provisioning profile, and click Validate.

Xcode uploads the archive to iTunes Connect and iTunes Connect runs validation tests. If a dialog appears stating that no application record can be found, click Done, create an app record in iTunes Connect, and repeat these steps.



6. Review validation issues that are found, if any, and click Done.

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

Testing the Mac Installer Package

Before you submit your app 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.

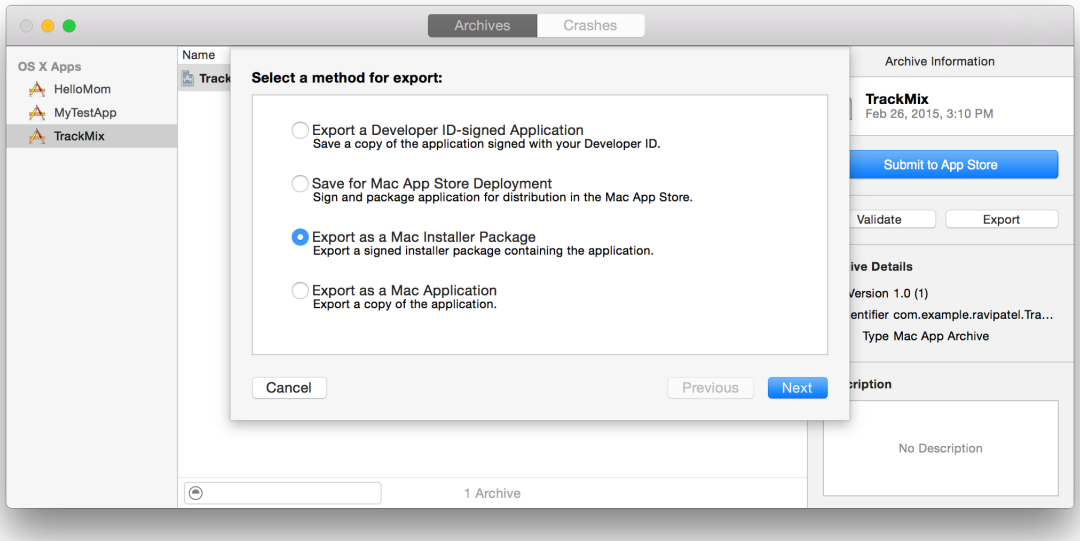
Creating an Installer Package

You save an installer package to your disk by following similar steps as for validating and distributing your Mac app.

To create an installer package

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

3. Select "Export as a Mac Installer Package," and click Next.



4. In the dialog that appears, choose a team from the pop-up menu and click Choose.

   If necessary, Xcode creates a distribution certificate and distribution provisioning profile for you. The name of the distribution provisioning profile begins with the text `XC:`.

5. In the dialog that appears, review the app, its entitlements, and provisioning profile, and click Export.
6. In the dialog that appears, enter a filename and choose a location, and click Export.

Testing the Installer Package

Don't 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's 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 /
```

The output of the `installer` command should be similar to:

```
rpatel$ sudo installer -store -pkg ../Documents/TrackMix.pkg -target /

WARNING: Improper use of the sudo command could lead to data loss
or the deletion of important system files. Please double-check your
typing when using sudo. Type "man sudo" for more information.
```

```
To proceed, enter your password, or type Ctrl-C to abort.


Password:
installer: Note: running installer as an admin user (instead of root) gives better
 Mac App Store fidelity
installer: TrackMix.pkg has valid signature for submission: 3rd Party Mac Developer
 Installer: Ravi Patel (7U3X8B3P5Z)
installer: Installation Check: Passed
installer: Volume Check: Passed
installer: Bundle com.example.rpatel.TrackMix will be installed to
/Applications/TrackMix.app
installer: Starting install
installer: Install 0.0% complete
installer: Install 90.8% complete
installer: Install 100.0% complete
installer: Finished install
rpatel$
```

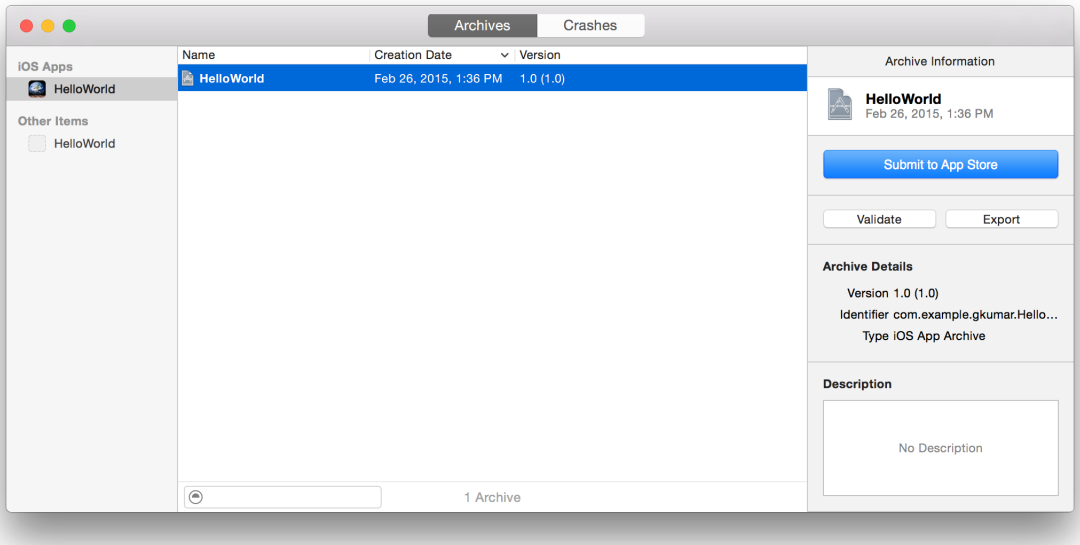
If the installer finds a bundle with the same bundle ID as the one it's installing, it upgrades the existing app. Users can then install upgrades even if they have moved your app. If you have a copy of your app installed (for example, in your build products directory), you may want to remove it so that 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.

Uploading Your App Using Xcode

Upload your code signed and provisioned app to iTunes Connect. This important step ensures that the upload comes directly from you and that only you grant permission for your app to use certain app services. Code signing your app or installer package prevents an attacker from uploading a modified version of your app—only someone with the private key for your distribution certificate can upload your app to iTunes Connect.

To upload an app to iTunes Connect

1. In the Archives organizer, select the archive and click the “Submit to the App Store” button.



2. In the dialog that appears, choose a team from the pop-up menu and click Choose.

   If necessary, Xcode creates a distribution certificate and distribution provisioning profile for you. The name of the distribution provisioning profile begins with the text `XC:`.

3. In the dialog that appears, review the app, its entitlements, and provisioning profile, and click Submit.

   Xcode uploads the archive to iTunes Connect and iTunes Connect runs validation tests. If a dialog appears stating that no application record can be found, click Done, create an app record in iTunes Connect, and repeat these steps.

4. If issues are found, click Done and fix them before continuing.

5. If no issues are found, click Submit to upload your app.

Xcode transmits the archive to Apple, where the binary is examined to determine whether it conforms to Apple guidelines. If the binary is rejected, correct the problems that were identified and upload a new binary. If you successfully upload your app, view the version and build of your app in iTunes Connect, as described in Viewing Binary Details. Later, use iTunes Connect to submit your app to App Review, as described in Submitting the App.

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

Recap

In this chapter, you learned how to submit your app to the store using Xcode. This chapter doesn't cover the steps to distribute your app using iTunes Connect. To learn how to view the status of your app, read Viewing the Status of Your App (page 140).

Releasing and Updating Your App on the Store

After your app is approved, you release and maintain your app throughout the lifetime of the app on the App Store or Mac App Store. For example, you view crash reports, respond to customer reviews, and fix problems as needed. You use iTunes Connect to release, manage, and update versions of your app. This chapter provides an overview of a few common tasks you'll perform.

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 140).

Enter sales and marketing information. Use iTunes Connect to prepare your app for purchase on the store, as described in *iTunes Connect Developer Guide*.

Release your app. Use iTunes Connect to set the availability date, as described in Changing the Availability Date of Your App (page 141).

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, correct the bug and submit a new version of the app through the approval process. To view customer reviews, read Viewing Customer Reviews (page 142).

View crash reports. Use iTunes Connect to download crash reports submitted to Apple by users. Crash reports represent significant problems that users find in the app. To access and analyze these crash reports, read Viewing Crash Reports (page 142) and Analyzing Crash Reports (page 114).

Update your app. You follow the same distribution process to upload 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 143).

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. It's important to make investigating these reports a high priority.

Managing Your App in iTunes Connect

iTunes Connect is a marketing and business web tool that iOS Developer Program and Mac Developer Program members 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, app services that it uses, and any version information in iTunes Connect. This chapter teaches you additional tasks you perform in iTunes Connect during development and distribution to the store.

Initially, only the individual who joins the iOS Developer Program and Mac Developer Program has access to iTunes Connect. 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, you can tightly control access to iTunes Connect separately from your Member Center account. For example, you can add nondeveloper iTunes Connect users and control access to metadata by assigning roles and privileges.

iTunes Connect users with admin and technical roles perform a number of tasks, explained in this chapter, in support of the development team and related to submitting your 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 key app services, and submit your app.
3. View the status of your app when you're 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.

For complete documentation on using iTunes Connect, refer to *iTunes Connect Developer Guide*.

About iTunes Connect User Roles and Privileges

The person who enrolls in the developer program—called 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 can't 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 certain app services. The technical role also grants permission to upload a binary to iTunes Connect and submit an app to App Review.
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 isn't 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 Apps	✓	✓	✓	✗	✗
Sales and Trends	✓	✓	✗	✓	✓
Tax and Banking	✓	✓	✗	✓	✗
Contracts	✓	✗	✗	✗	✗
Payments and Financial Reports	✓	✓	✗	✓	✗

Responsibility	Legal	Admin	Technical	Finance	Sales
Catalog Reports	✓	✓	✓	✓	✓

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.

Adding iTunes Connect Users

To add an iTunes Connect user, read Setting Up User Accounts.

Creating an App Record

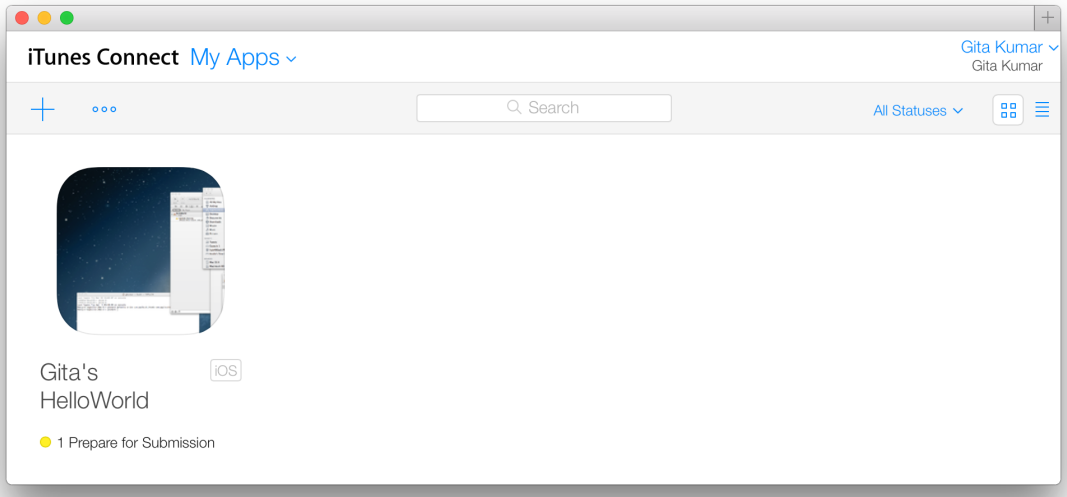
Certain app services require you to create an app record and enter the bundle ID using iTunes Connect during development. Later, you also need to create an app record in iTunes Connect to upload your app for distribution. When you're ready to create your app record, read Creating an iTunes Connect Record for an App. To learn about other information you enter in iTunes Connect, and to view the versions and builds you upload to iTunes Connect, read Viewing and Changing Your App's Metadata.

Viewing the Status of Your App

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 My Apps.
3. Locate the app you want to edit.



   The status of each version of your app appears below the icon and title.
4. Click the large icon or app name to see more details.

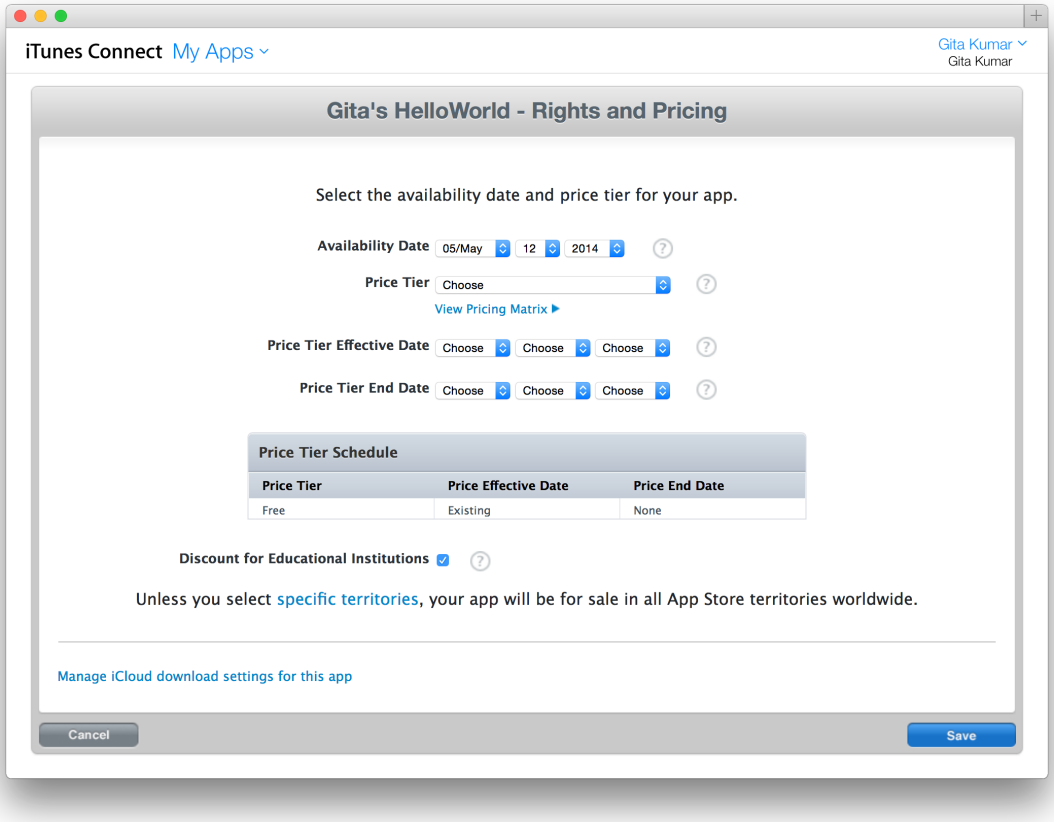
For details on each status and how to view the status history, refer to Viewing and Changing Your App's Status and Availability.

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's 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. On the iTunes Connect homepage, click My Apps.
3. Locate the app and click the large icon or app name.
4. Click Pricing.
5. Choose a date from the Availability Date pop-up menus.



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

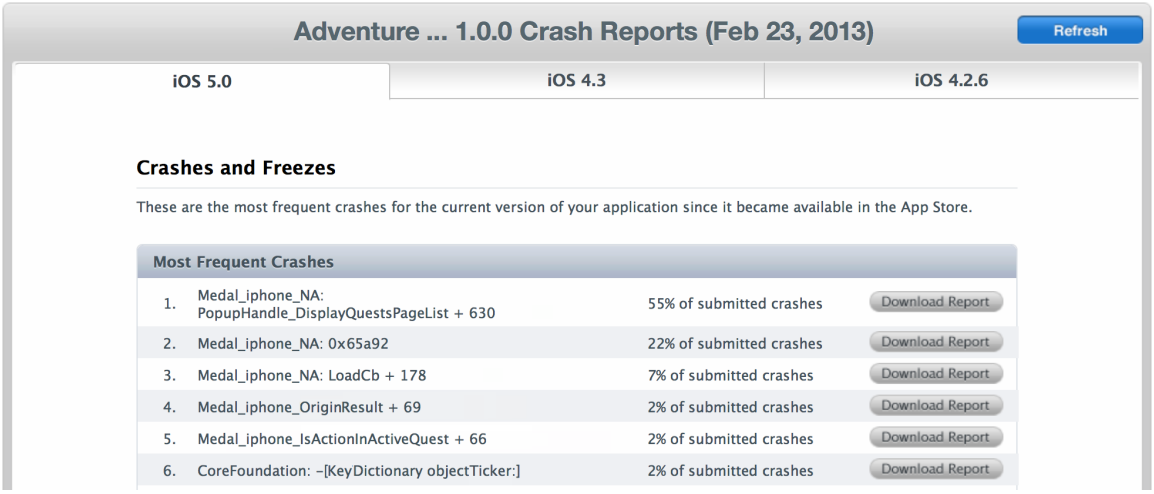
Viewing Crash Reports

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.

To download and view crash reports from iTunes Connect

1. Sign in to iTunes Connect.
2. On the iTunes Connect homepage, click My Apps.
3. Locate the app, and click the large icon or app name.
4. In the Versions pane, select the version, scroll to the bottom, and under Additional Information, click Crash Reports.

   Crash Reports appears only if iTunes Connect has crash logs available for this version of the app.
5. Click Download Report in the row of the crash report you want to download.



To view the crash reports in Xcode, follow the steps in Analyzing Crash Reports (page 114).

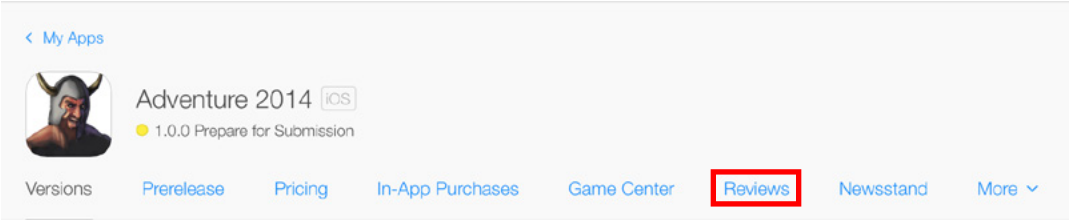
Viewing Customer Reviews

You can view the most recent feedback from your users for an app that is available on the store.

To view customer reviews

1. Sign in to iTunes Connect.
2. On the iTunes Connect homepage, click My Apps.

3. Locate the app, and click the large icon or app name.
4. Click Reviews.



   The list of customer reviews appears.
5. Select the review you want to view.
6. Click Done.

Creating New Versions of Your App

To create new versions of your app, read Replacing Your App with a New Version.

Recap

In this chapter, you learned how to grant access to iTunes Connect and perform common iTunes Connect tasks performed by technical team members.

Managing Your Team in Member Center

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 approve signing certificates and create provisioning profiles for team members. If you're an individual, you're the team agent for your one-person team and don't perform any of the tasks described in this chapter.

> **Note:** Team members aren't the same as iTunes Connect users. Only the person who joins the iOS Developer Program or Mac 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 137).

About Apple Developer Program Team Roles and Privileges

A person's role on the team defines the level of access that person 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 create signing identities and provisioning profiles. 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. By giving you control over team roles, Apple makes it easier for you to maintain good security practices for the team.

> **Note:** Some privileges—such as the ability to upload a binary to iTunes Connect and submit it to App Review—are controlled by the iTunes Connect user role, described in About iTunes Connect User Roles and Privileges (page 137).

Team Roles

Table 12-1 lists the roles a person can play and describes each. Each level of access includes all the capabilities of the levels below it.

Table 12-1 Team roles

Role	Description
Team agent	A *team agent* 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's only one team agent.
Team admin	A *team admin* 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 *team member* can 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 12-2 lists the specific privileges granted to members of the team. The privileges are listed in chronological order to help guide you through the process. Refer to Table 13-1 (page 174) for the types of certificates that each team member can revoke.

Table 12-2 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	✓	✗	✗
Enroll in additional developer programs and renew them	✓	✗	✗
Invite team admins and team members	✓	✓	✗
Request development certificates	✓	✓	✓
Approve team member requests for development certificates	✓	✓	✗

Privilege	Team agent	Team admin	Team member
Request distribution certificates	✓	✓	✗
For Mac apps, request Developer ID certificates	✓	✗	✗
Add devices for development and testing	✓	✓	✗
Create App IDs and enable certain app services	✓	✓	✗
Create development and distribution provisioning profiles	✓	✓	✗
Create SSL certificates for Apple Push Notification service	✓	✓	✗
Download development provisioning profiles	✓	✓	✓

Team Agent

To start, one person must enroll in either the iOS Developer Program or 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 has an unrestricted role; 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.

The team agent might need to sign updated or new licensing agreements, particularly when the team wants to incorporate specific services into an app. For example, an app that uses the iAd service requires that the team agent sign a separate agreement.

Inviting Team Members and Assigning Roles

If you enroll as a company, you're 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 don't have write access. However, you can assign an admin role to a

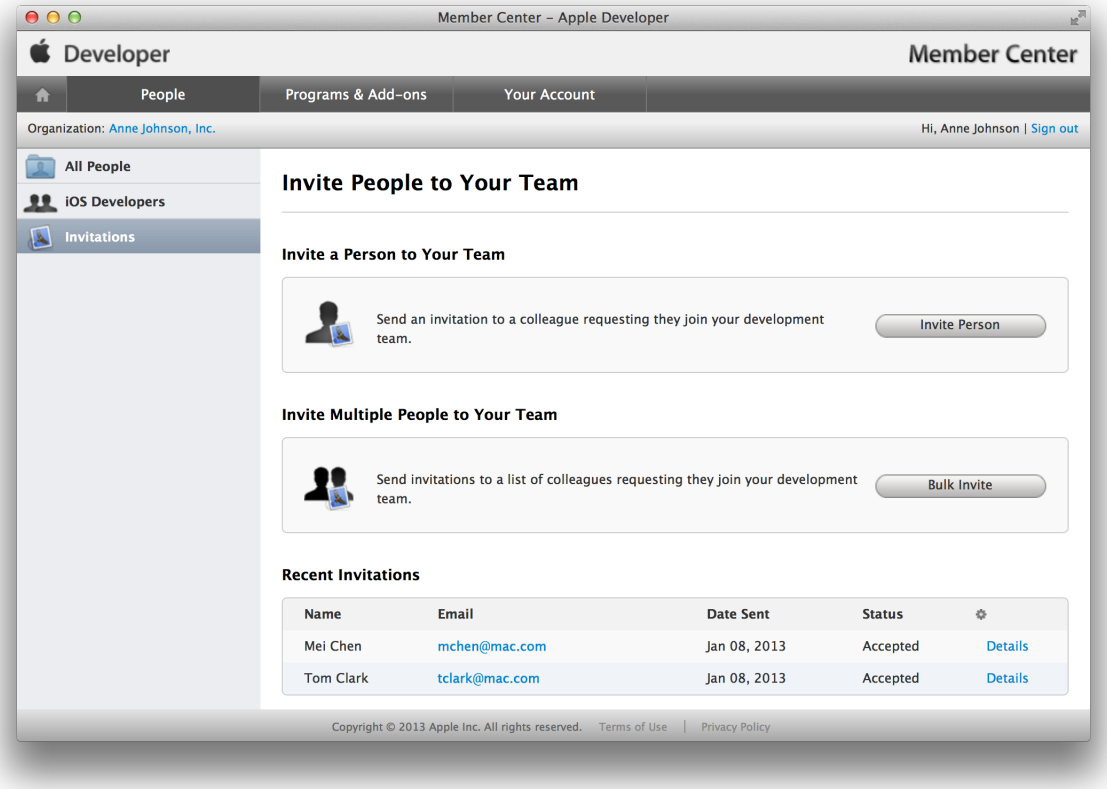
team member, which allows that person to have some of the privileges of a team agent—for example, a team admin can create signing certificates and provisioning profiles but can't sign agreements. Assigning roles helps team agents delegate some of their responsibilities.

Inviting Team Members

When you invite people to join your team, you enter information about them and set their role on the team.

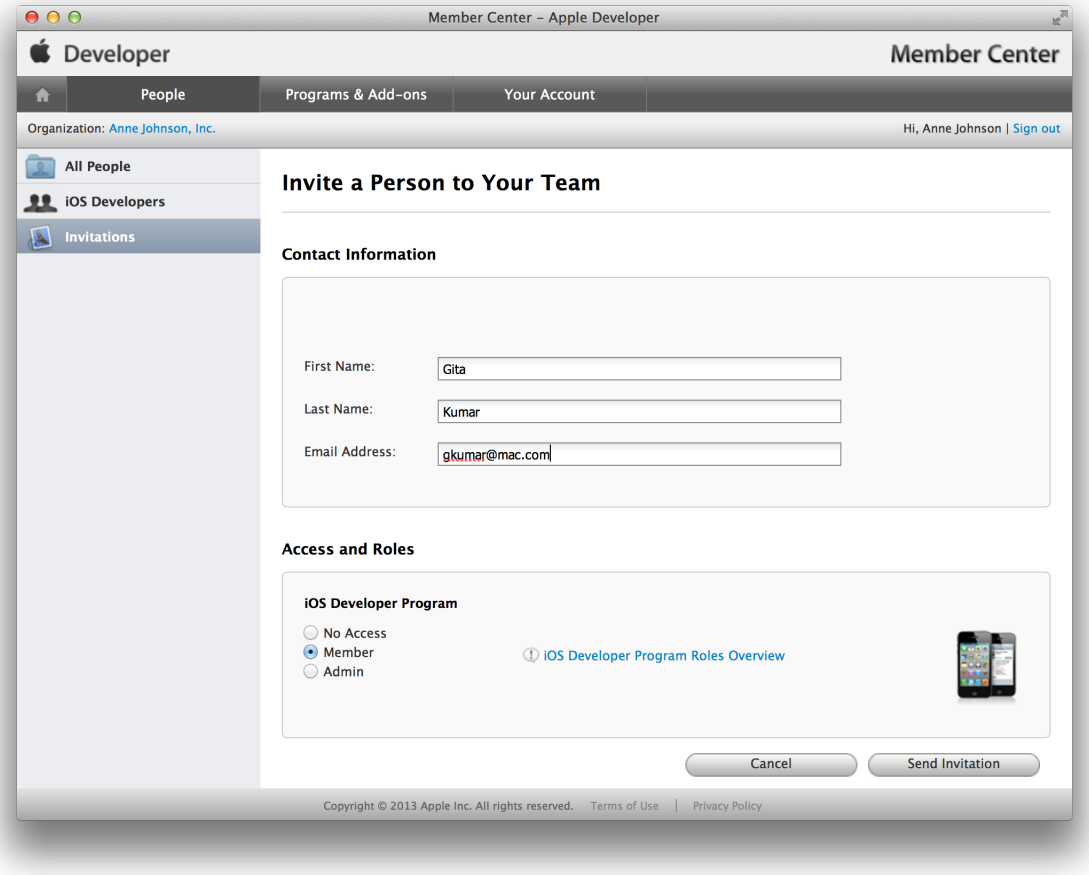
To invite team members

1. In Member Center, click People at the top of the webpage.
2. If necessary, 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 doesn't have an Apple ID, he or she is asked to create one before accepting the invitation.

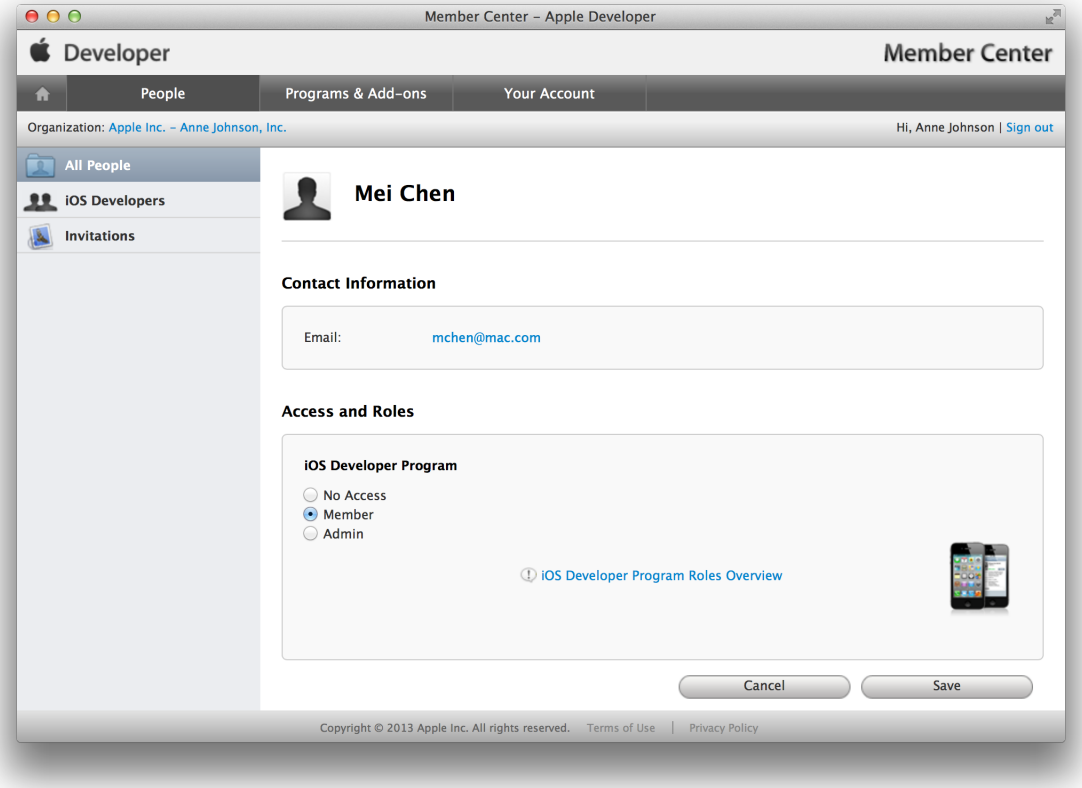
Changing Team Roles

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 at the top of the webpage.
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.

4. Specify the person's access and role for each program, and click Save.



Approving Development Certificates

If you're a team admin for a company, it's your responsibility to approve team member requests for development certificates. Team members need a development certificate 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 Identities (page 156). 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 refresh provisioning profiles in Xcode, as described in Refreshing Provisioning Profiles in Xcode (page 205), to download the approved certificate, and export his or her developer profile, as described in Exporting Your Developer Profile (page 166).

To approve a development certificate request

1. In Certificates, Identifiers & Profiles, select Certificates.
2. Under Certificates, select Pending.
3. Select the certificate.
4. Click Approve.

5. In the dialog that appears, click Approve again.

If you use the team provisioning profile, regenerate it after approving the certificate. Xcode regenerates the team provisioning profile whenever a team member refreshes provisioning profiles in Xcode, as described in Refreshing Provisioning Profiles in Xcode (page 205). Afterward, all other team members should refresh their provisioning profiles to download the latest team provisioning profile.

Registering Team Member Devices

Before a team member can launch an app on his or her device, the device needs to be registered and added to the team provisioning profile. Xcode automatically registers team agent and admin devices when needed, as described in Launching Your App on Devices (page 88). However, a team agent or admin must register team member devices on their behalf.

The team member sends the device name and device ID to his or her team admin. In Xcode, a team member can select the device in the Devices window to display the device ID, as described in Locating Device IDs Using Xcode (page 192). If you're a Mac developer, you can also get the device ID using the System Information app, as described in Locating Mac Device IDs Using System Preferences (page 193).

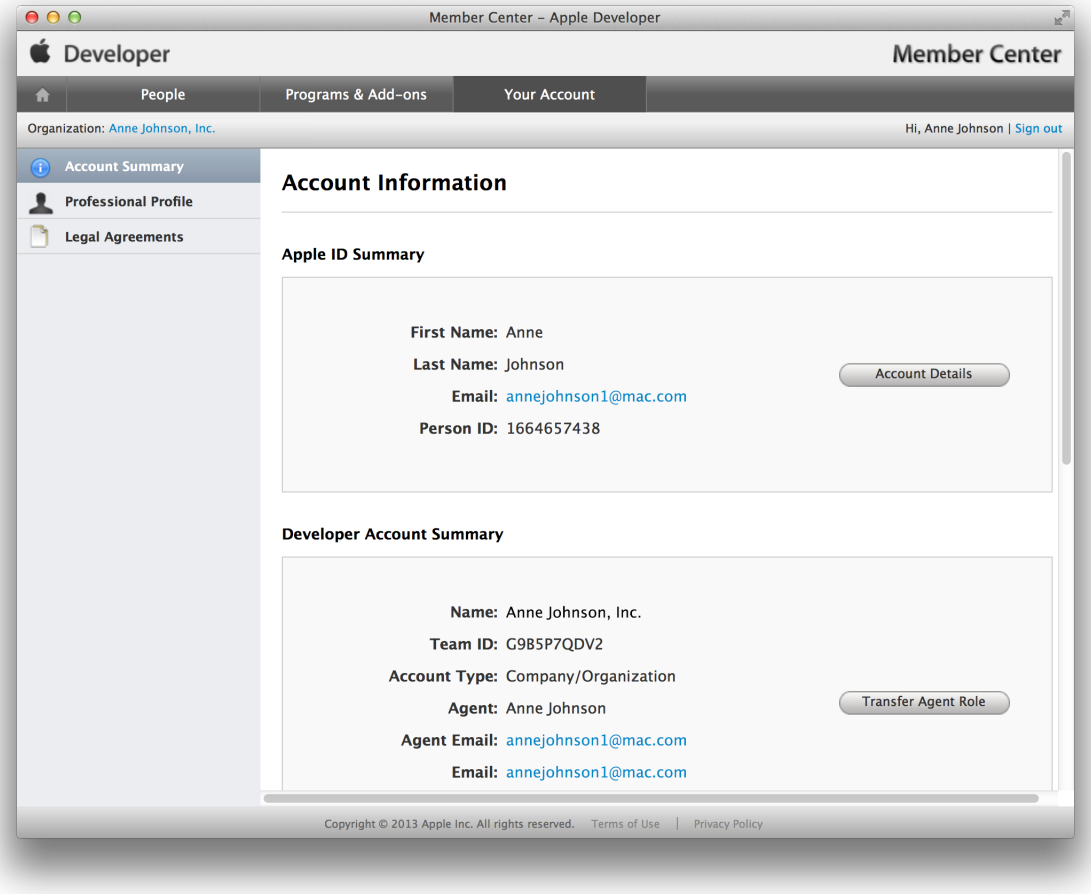
In Member Center, the team admin may register one device, as described in Registering Individual Devices (page 194), or multiple devices, as described in Registering Multiple Devices (page 195).

Transferring the Team Agent Role

Because the team agent has sole legal responsibility for the team, another team member can't demote the team agent, nor can the team agent's privileges be restricted. However, the team agent can transfer that role to another team member using Member Center.

To invite team members

1. In Member Center, sign in as the team agent and click Your Account at the top of the webpage.



2. In the Developer Account Summary section, click Transfer Agent Role next to your name.
3. Follow the instructions that appear in a series of dialogs.

   For example, you will be asked to choose a new team agent and sign an Agent Transferor Agreement.

Removing Team Members

If a team member is no longer working on your project, you can remove him or her from your team.

To remove a team member

1. In Member Center, click People at the top of the webpage.
2. Click All People in the sidebar.
3. Select the checkbox in the row of the person you want to delete.
4. In the lower-left corner, click Delete.

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.

Maintaining Your Signing Identities and Certificates

Code signing your app lets users trust that your app has been created by a source known to Apple and that it hasn't been tampered with. All iOS apps and most Mac apps must be code signed and provisioned to launch on a device, to be distributed for testing, or to be submitted to the store. Code signing uses cryptographic technology to digitally sign your app and installer package. You create signing identities—stored in your keychain—and certificates—stored in Member Center—to sign and provision your app. These assets uniquely identify you or your team, so it's important to keep them safe. This chapter covers common tasks that you perform to protect and maintain your signing identities and certificates over the lifetime of your project.

For the types of the certificates you use to develop, test, and distribute your app, refer to Your Signing Certificates in Depth (page 182).

About Signing Identities and Certificates

Code signing your app allows the operating system to identify who signed your app and to verify that your app hasn't 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 aren't signed; therefore, a change to these files doesn't invalidate the signature.

Code signing is used in combination with your App ID, provisioning profile, and entitlements 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 isn't using app services 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.

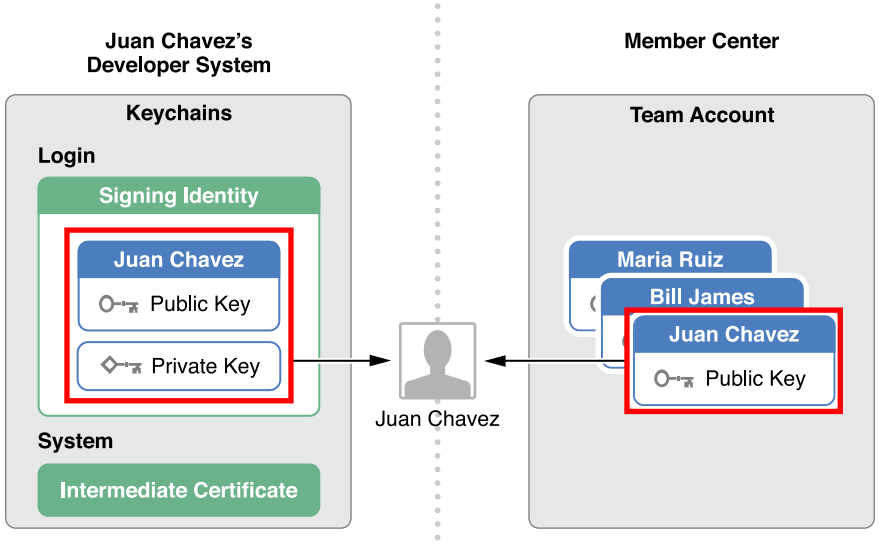
Code signing also allows your app's signature to be removed and re-signed by a trusted source. For example, you sign your app before submitting it to the store, but Apple re-signs it before distributing it to customers. Also, you can re-sign and submit a fully tested development build of your app to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

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

Signing requires that you have both the signing identity and the intermediate certificate installed in your keychain. When you install Xcode, Apple's intermediate certificates are added to your keychain for you. You use Xcode to create your signing identity and sign your app. Your signing identity is added to your keychain, and the corresponding certificate is added to your account in Member Center.

Signing identities are used to sign your app or installer package. A *development certificate* identifies you, as a team member, in a development provisioning profile that allows apps signed by you to launch on devices. A *distribution certificate* identifies your team or organization in a distribution provisioning profile and allows you to submit your app to the store. Only a team agent or an admin can create a distribution certificate. You also use different development and distribution certificates to sign iOS and Mac apps. For a complete list of certificate types, refer to Your Signing Certificates in Depth (page 182).

For a company, other team members have their own signing identities installed on their Mac computers. Member Center contains a repository for all of the combined team assets but doesn't store any of the private keys.



Because the private key is stored locally on your Mac, protect it as you would an account password. Keep a secure backup of your public-private key pair. If the private key is lost, you'll have to create an entirely new identity to sign code. Worse, if someone else has your private key, that person 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 can't be retrieved if lost.

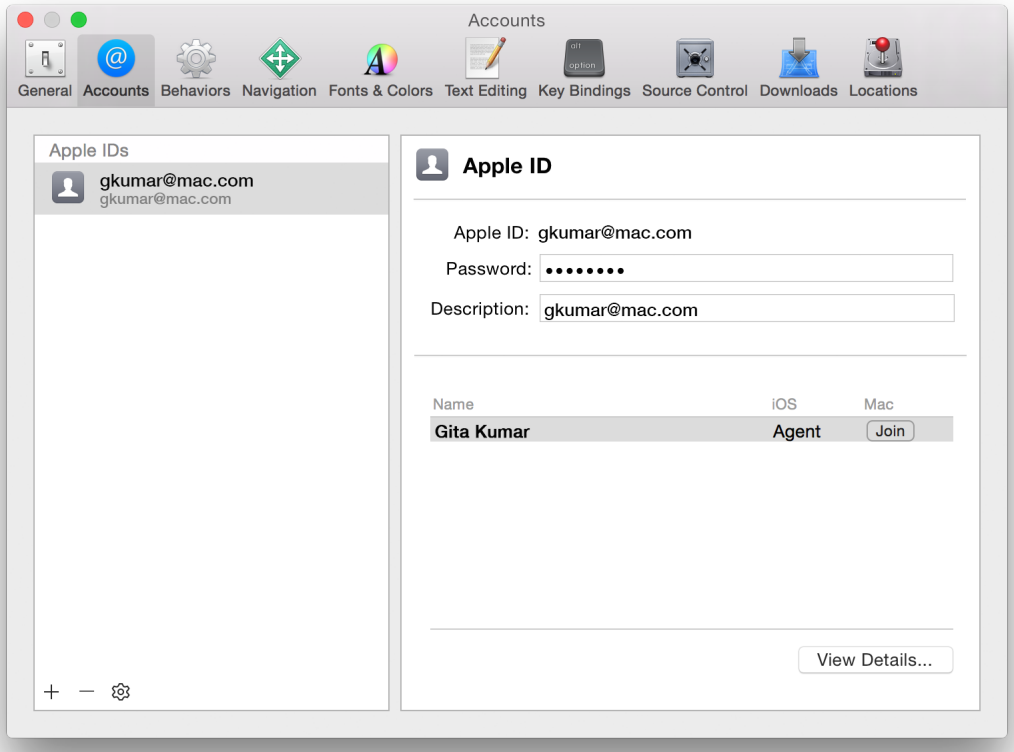
If you want to code sign your app using another Mac, you export your developer profile on the Mac you used to create your certificates and import it on the other Mac. You can also share distribution certificates among multiple team agents using this feature. (Team members should not share development certificates.)

Viewing Signing Identities and Provisioning Profiles

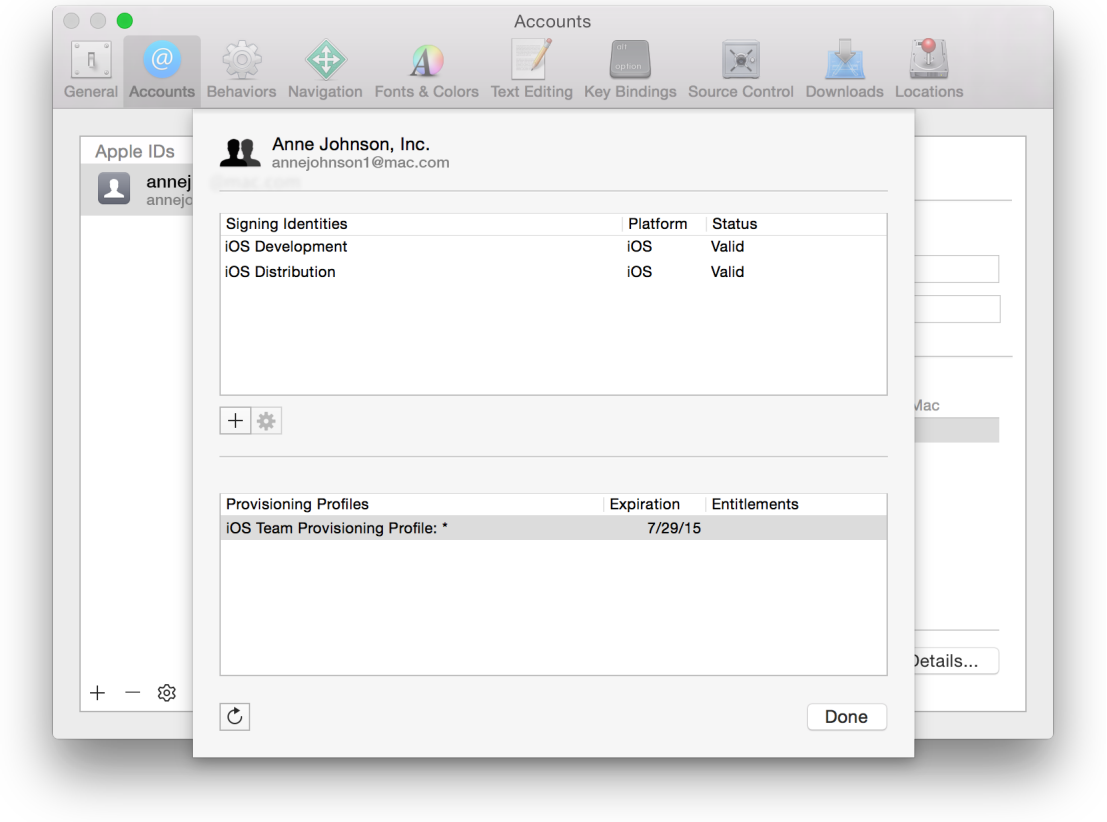
To verify or troubleshoot your certificates and profiles, view them in Xcode Accounts preferences. Although Xcode manages these assets for you, you may occasionally need to request or revoke specific certificates and to refresh your provisioning profiles.

To view account details

1. Choose Xcode > Preferences.
2. Click Accounts at the top of the window.
3. Select the team you want to view, and click View Details.



In the dialog that appears, view your signing identities and provisioning profiles.



4. Click Done.

Requesting Signing Identities

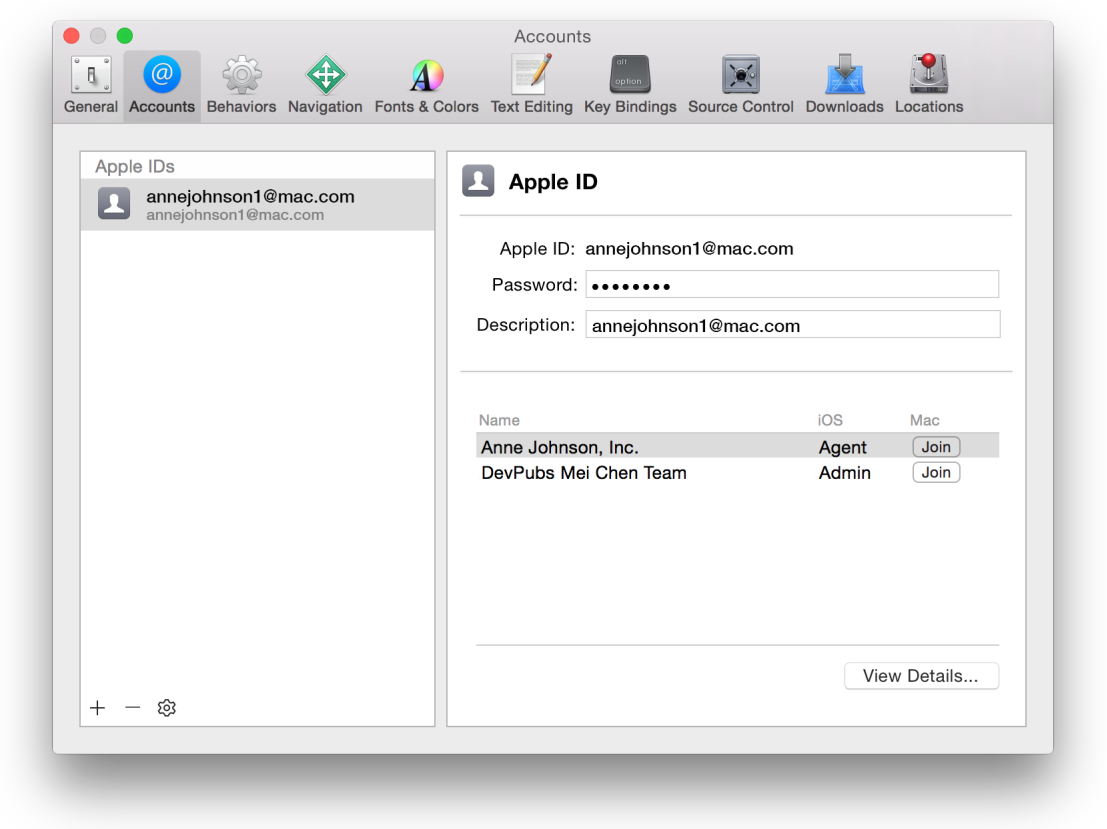
Before you can code sign your app, you create your development certificate and later, a distribution certificate to upload your app to iTunes Connect. You can create all the types of certificates and signing identities you need using Xcode. Xcode requests, downloads, and installs your signing identities for you.

For a company, a team member requests his or her development certificate using Xcode but downloads and installs it later, after it's approved, as described in Approving Development Certificates (page 149). Only a team agent or admin can create a distribution certificate. Only a team agent can create a Developer ID certificate. If you have a company membership, read Managing Your Team in Member Center (page 144) for a description of team roles and tasks that team agents perform on behalf of team members.

Xcode asks to create development certificates for you when you need them. For example, when you assign your project to a team or create the team provisioning profile, as described in Configuring Identity and Team Settings (page 26), a dialog might appear asking if Xcode should create a certificate for you. Because of this, you typically request distribution certificates using the Xcode Preferences window.

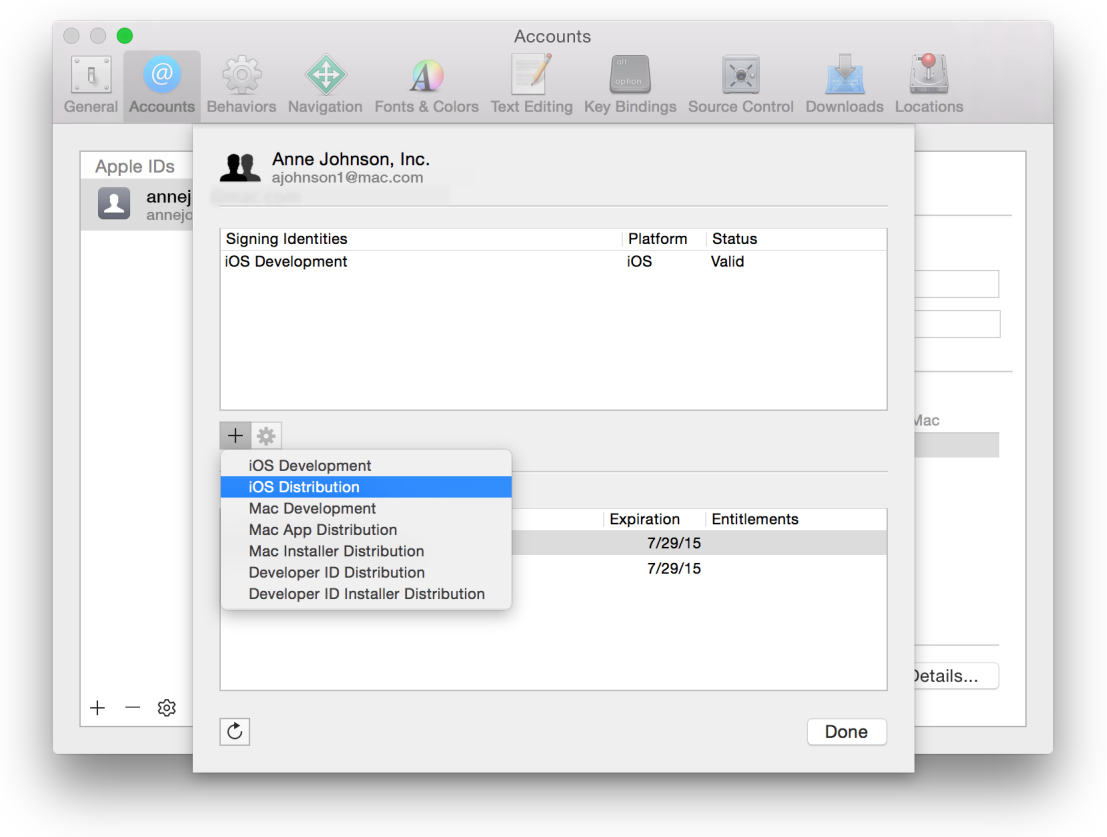
To request a signing identity

1. In the Xcode Preferences window, click Accounts.
2. Select the team you want to use, and click View Details.



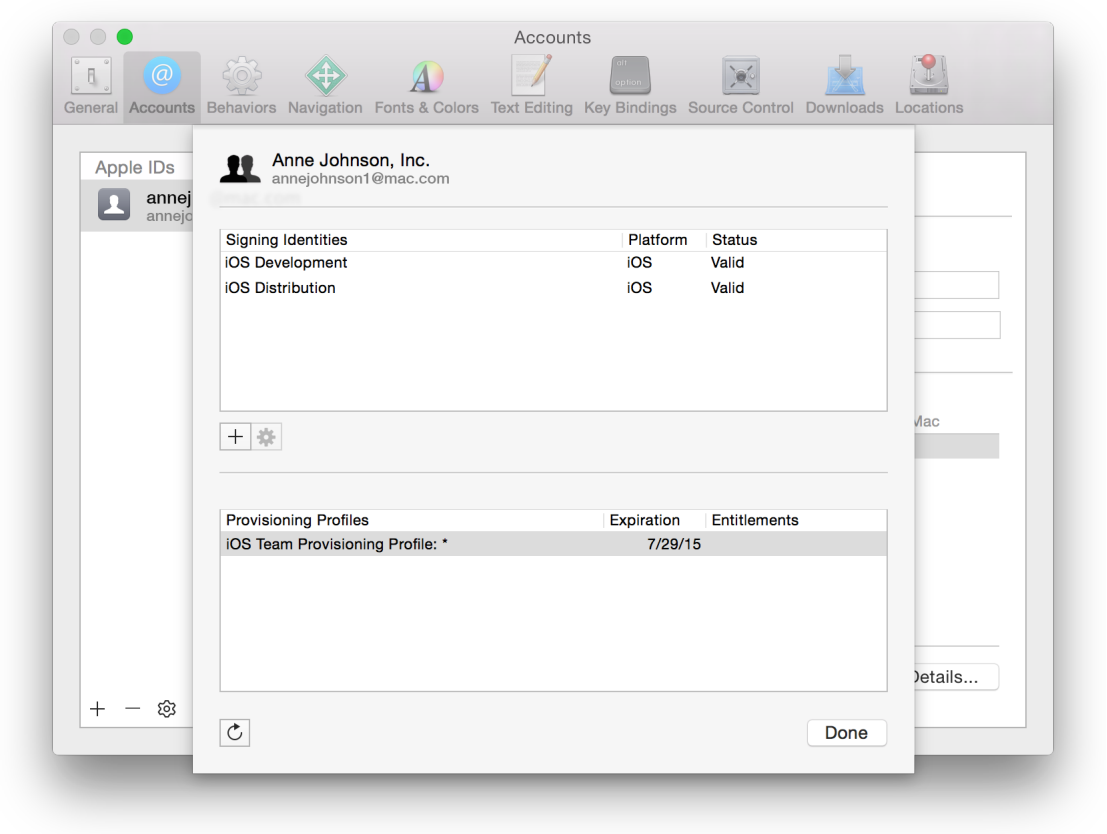
3. In the dialog that appears, choose the type of certificate you want to create by clicking the Add button (+) below the Signing Identities table.

For a description of each type of certificate, refer to Table 13-2 (page 182).



4. In the Signing Identity Generated dialog that appears, click OK.

The new signing identity appears in the Signing Identities table.



A team member may need to wait until a team agent or admin approves the request.

5. To return to Accounts preferences, click Done.

You can now export your signing identities to create a backup, as described in Exporting Your Developer Profile (page 166).

Verifying Your Steps

Verify that your certificates are correct and ready for use. Certificates must be valid in order to sign your app—and for a Mac app, to sign your installer package.

The first time you verify your certificates, verify them in Xcode, Keychain Access, and Member Center to learn where they're located and how they appear in each tool. Keychain Access and Member Center display the expiration dates of your signing identities and certificates. Later, you'll use Keychain Access for troubleshooting.

> **Note:** The name of the certificate in Keychain is different from the certificate type that appears in Xcode and Member Center. For the mapping between the certificate names and types that appear in the tools, refer to Table 13-2 (page 182).

Verifying 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 or Production depending on the type of certificate you want to verify.

   The name, type, and expiration date of the certificate should match the information that you view in Xcode.

Verifying Using Keychain Access

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

To verify signing identities 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. In the left pane, select “login” in the Keychains section and select Certificates in the Category section.

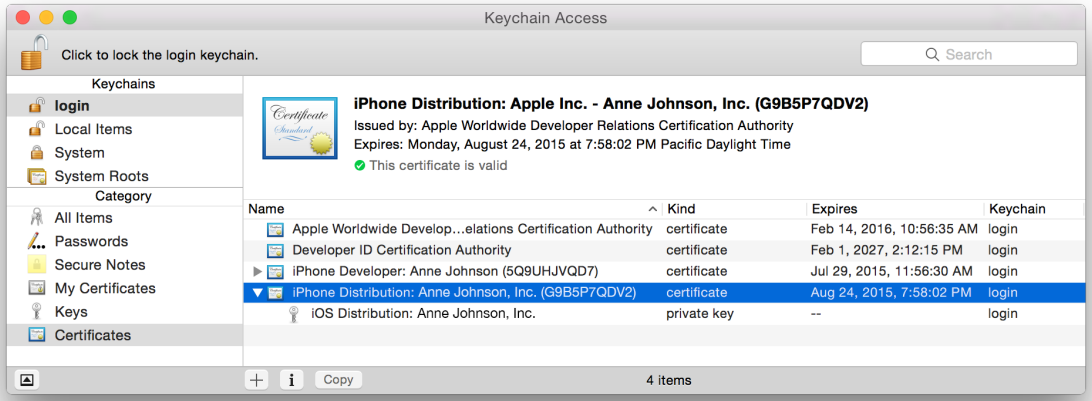
Your development and distribution certificates appear in the Certificates category in Keychain Access. The name of the development certificate begins with the text “iPhone Developer” for the iOS Developer Program and “Mac Developer” for the Mac Developer Program, followed by your name (development certificates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

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

> **Important:** 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 isn’t login, select login in the Keychains column and choose File > Make Keychain “login” Default.

3. Verify that there’s 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’re missing your private key. (Read The Private Key for Your Signing Identity Is Missing (page 242) to fix this issue.)



4. Verify that the certificates are valid.

When you select a certificate, a 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.”

Troubleshooting

If the certificates shown in Xcode and Keychain Access don’t match your certificates in Member Center, read Certificate Issues (page 241) for information about how to resolve the discrepancies.

Requesting Additional Developer ID Certificates

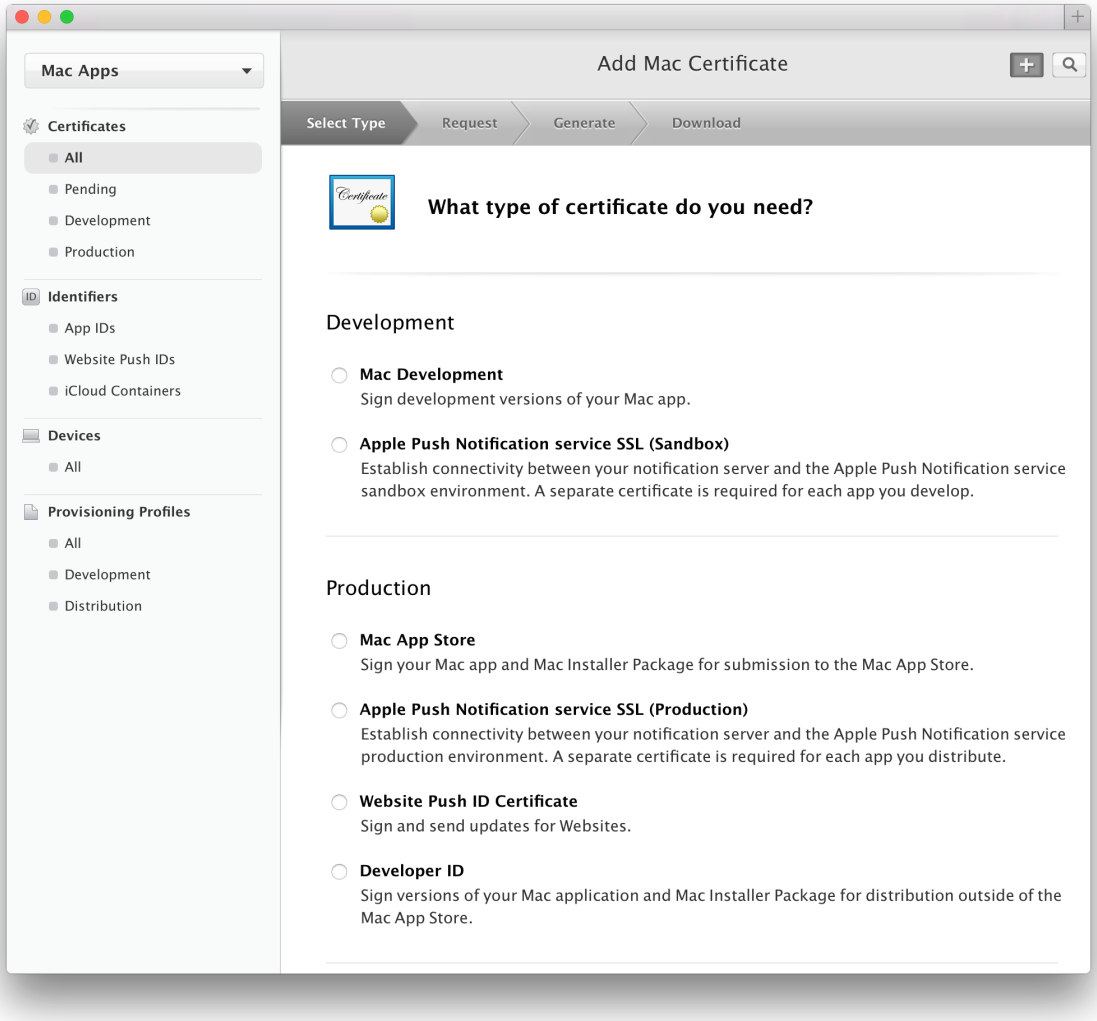
Developer ID certificates are used to distribute your application outside of the Mac App Store. Create your Developer ID certificates, along with other types of certificates, using Xcode, as described in Requesting Signing Identities (page 156). If you want more Developer ID certificates, you can create up to five of each type using Member Center.

> **Important:** Only team agents can request additional Developer ID certificates.

To create a Developer ID certificate

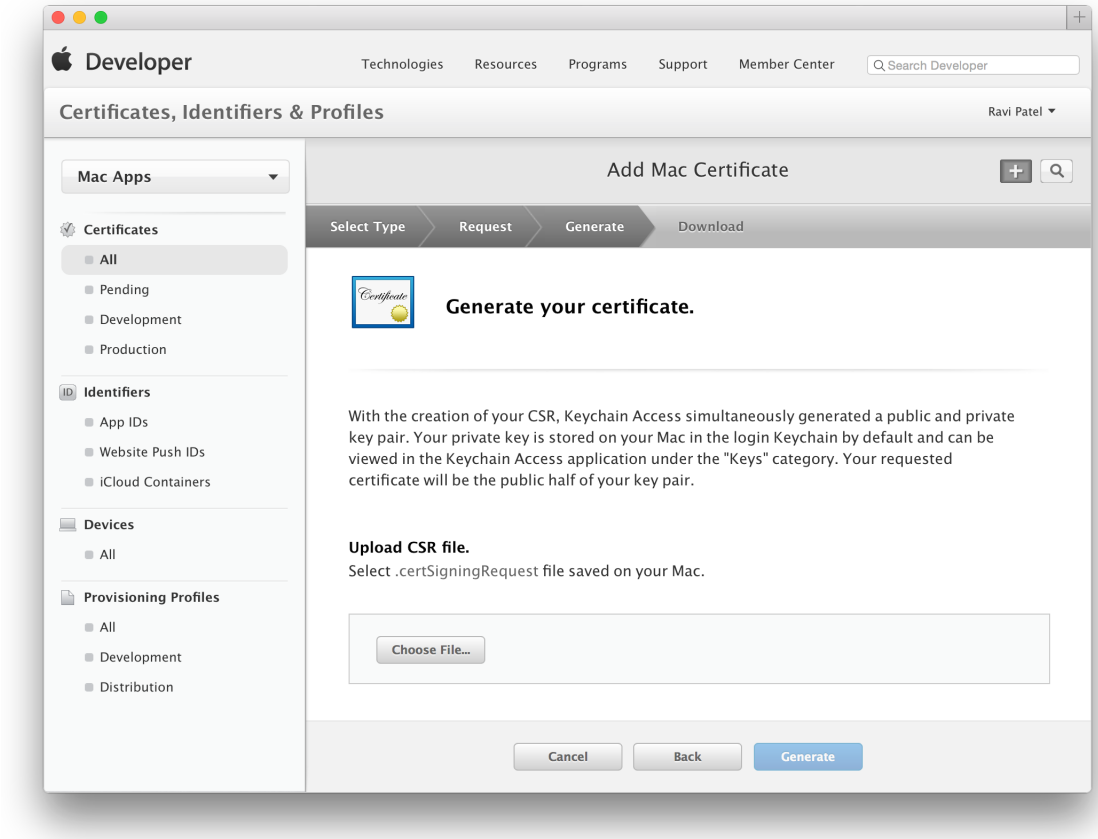
1. In Certificates, Identifiers & Profiles, select Certificates.
2. Under Certificates, select All.
3. Click the Add button (+) in the upper-right corner.

4. Select Developer ID under Production, 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 appears in the My Certificates category in Keychain Access.

Installing Missing Intermediate Certificate Authorities

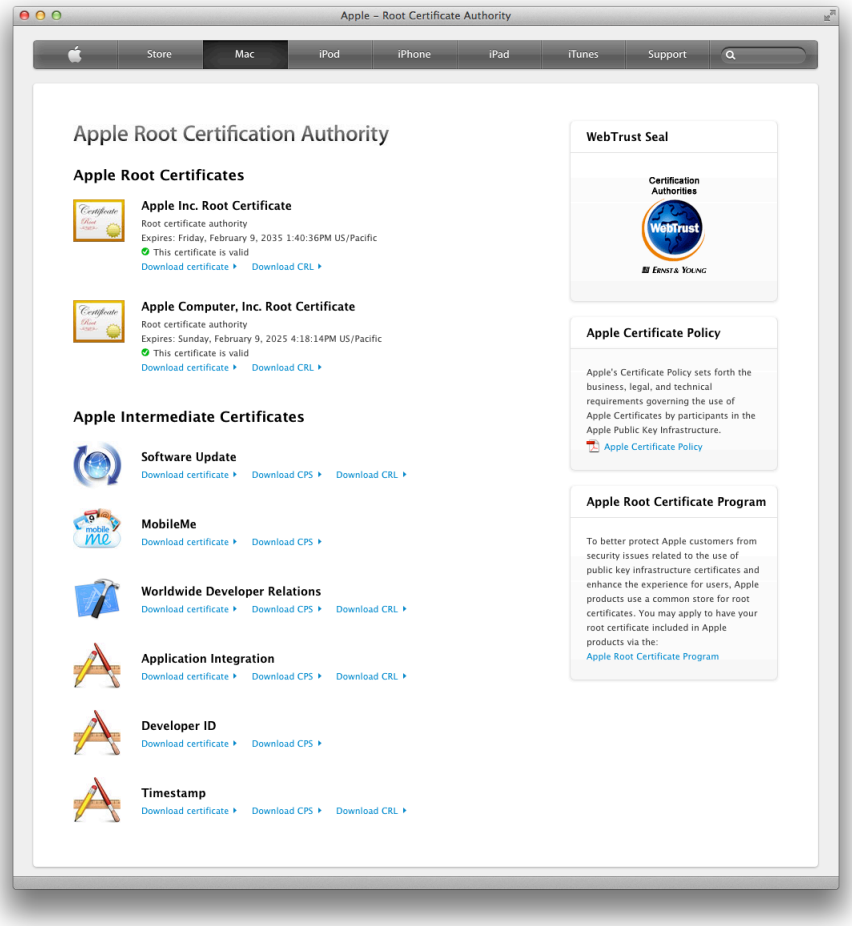
To use your certificates, the correct intermediate certificate needs to be in your keychain. An intermediate certificate ensures that your certificates were issued by a trusted source. The intermediate certificate, named Apple Worldwide Developer Relations Certification Authority, is installed in your 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.

First try refreshing your provisioning profiles in Xcode, as described in Refreshing Provisioning Profiles in Xcode (page 205), to install missing intermediate certificates. If that doesn't work for you, download and install the missing intermediate certificate.

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're 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.

Exporting and Importing Certificates and Profiles

After Xcode creates your certificates and profiles for you, export them to create a backup of all your assets. You do this to, for example, transfer your assets to another Mac you use for development or repair a certificate if the private key is missing. Refreshing your certificates and profiles in Xcode won't replace a missing private key. Instead, import your certificates and profiles from a backup.

The export file, called a *developer profile*, contains the following team assets:

- Development certificates
- Distribution certificates
- Provisioning profiles

You can also export selected certificates to share with other team members. In this case, the export file contains just the certificates you select.

Exporting Your Developer Profile

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

To export your developer account assets

1. Choose Xcode > Preferences.
2. Click Accounts at the top of the window.

3. Click the Action button (the gear icon to the right of the Delete button) in the lower-left corner, and choose Export Accounts from the pop-up menu.

4. Enter a filename in the Save As field and a password in both 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.

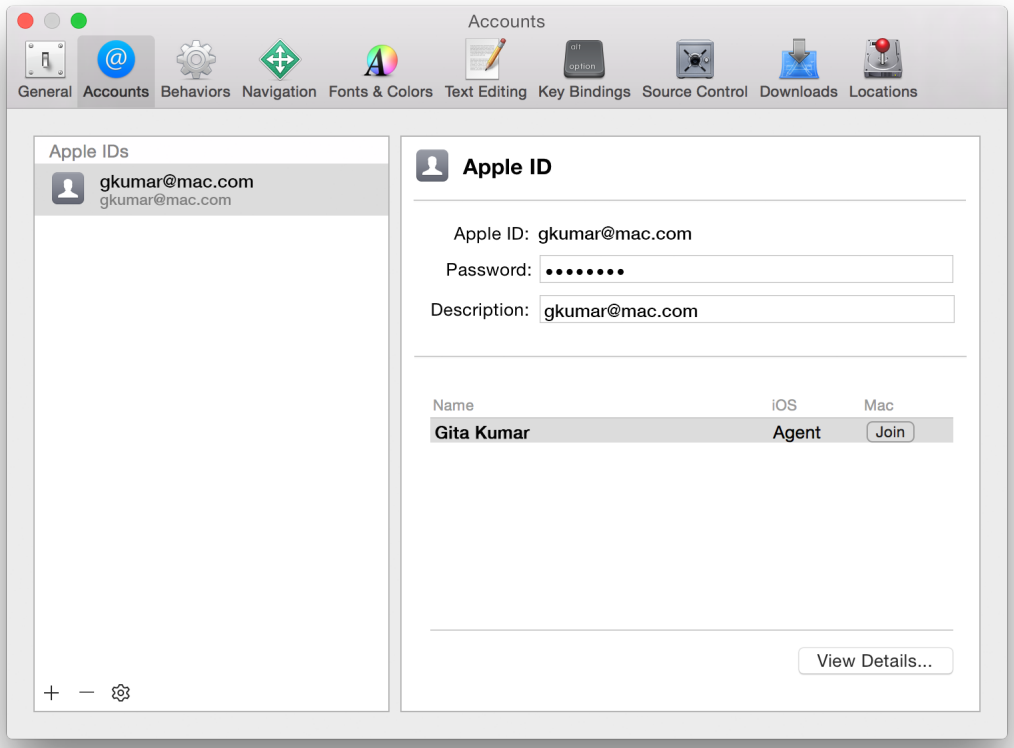
6. In the dialog that appears, click OK.

Exporting Selected Certificates

To export a few certificates and exclude the profiles, select the certificates in the details dialog.

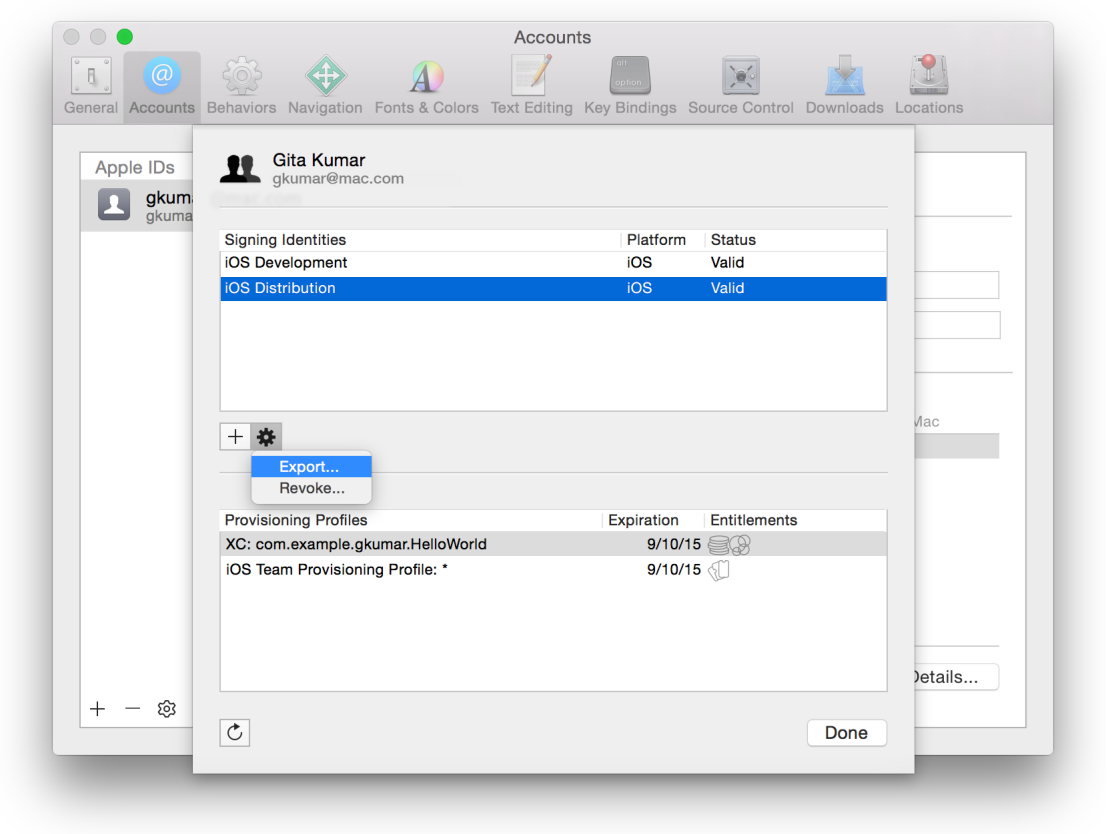
To export selected certificates

1. Choose Xcode > Preferences.
2. Click Accounts at the top of the window.
3. Select the team you want to view, and click View Details.



4. Select the certificates you want to export in the Signing Identities table.

5. Choose Export from the Action pop-up menu below the Signing Identities table.



6. Enter a filename in the Save As field and a password in both the Password and Verify fields.

   The file is encrypted and password protected.

7. Click Save.

   The file is saved to the location you specified with a `.p12` extension.

8. Click Done.

Alternatively, you can export certificates using the `security(1)` command-line utility.

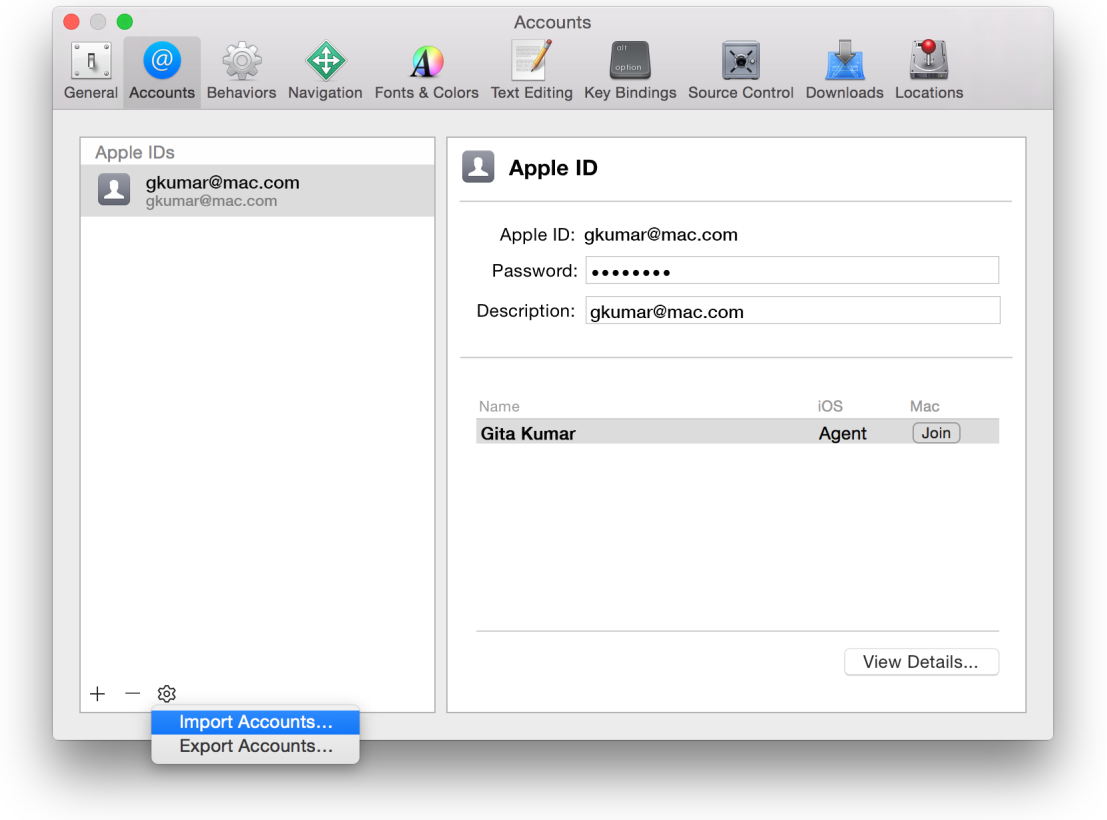
Importing Your Developer Profile

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

To import your developer account assets

1. Choose Xcode > Preferences.
2. Click Accounts at the top of the window.

3. Click the Action button (the gear icon) in the lower-left corner, and choose Import Accounts from the pop-up menu.



4. Locate and select the file containing your developer profile, and click Open.

   The file should have a `.developerprofile` extension.

5. Enter the password you used to encrypt the file, and click OK.

6. In the dialog that appears, click OK.

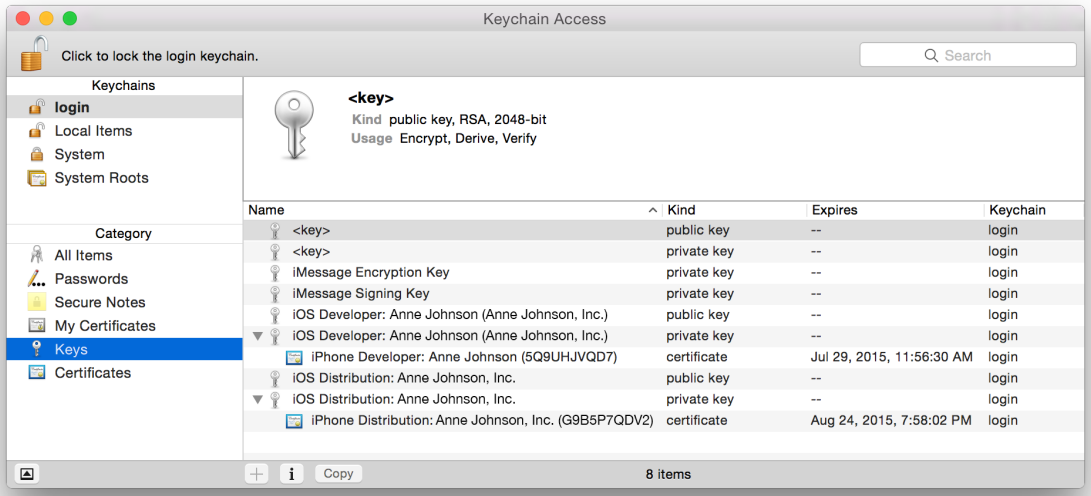
Removing Signing Identities from Your Keychain

You remove signing identities from your keychain if they're invalid, no longer used (perhaps they belong to a previous team you were a member of), or are missing the private key and consequently, aren't usable. If you're missing a private key and have a backup of your signing identities, import your developer profile, as described in Importing Your Developer Profile (page 171), immediately after removing the signing identities. If you remove signing identities for some other reason, read all the steps in Re-Creating Certificates and Updating Related Provisioning Profiles (page 178) to avoid code signing issues later.

Warning: You can't re-create a private key once you remove it from your keychain unless you import it from a developer profile file. If you're 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 can't sign apps using the certificates.

To remove signing identities 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.

   For how to recognize the type of certificate by the name as it appears in Keychain Access, refer to Table 13-2 (page 182).
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.

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 241) for the types of problems that can occur). You also revoke certificates if you suspect that they have been compromised. If you're a team admin for a company, you may want to revoke development certificates of team members who no longer work on your project. Revoking certificates may invalidate provisioning profiles, so read all the steps in Re-Creating Certificates and Updating Related Provisioning Profiles (page 178) to avoid code signing issues later.

> **Important:** Revoking development or distribution certificates doesn't affect apps that you've submitted to the store nor does it affect your ability to update them.

Revoking Privileges

Table 13-1 lists the types of certificates that 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. For a company, any team member can revoke his or her own development certificate, but a team member can only revoke distribution certificates if he or she is a team agent or admin.

Table 13-1 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	✓	✗	✗

Type of certificate	Team agent	Team admin	Team member
Store distribution certificates: iOS Distribution Mac App Distribution Mac Installer Distribution	✓	✓	✗
Developer ID certificates: Developer ID Application Developer ID Installer	✗	✗	✗
Push notification certificates: APNs Development iOS APNs Production iOS APNs Development Mac APNs Production Mac	✓	✓	✗
Pass certificate: Pass Type ID	✗	✗	✗

You can't revoke Developer ID or Passbook certificates using Member Center. Instead, send a request to Apple at `product-security@apple.com` to revoke these types of certificates.

If Apple revokes your Developer ID certificate, users can no longer 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 162).

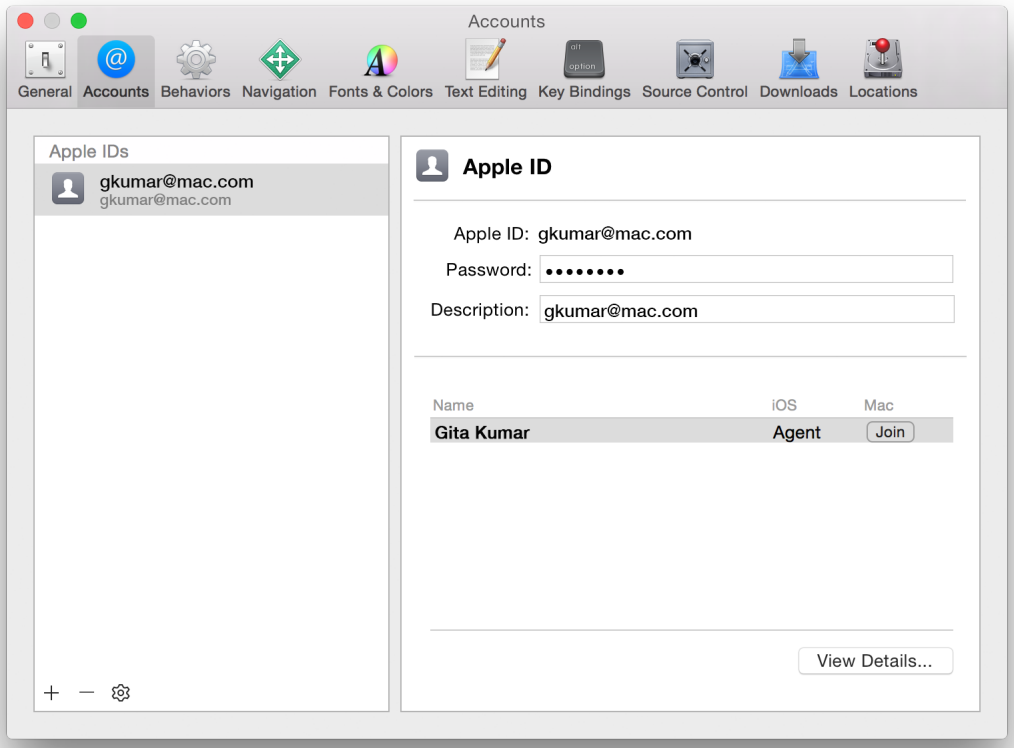
Revoking Development Certificates Using Xcode

If the development certificate you want to revoke appears in Accounts preferences in Xcode—it's a certificate you created on your Mac and is in your keychain—you can revoke it using Xcode. Otherwise, use Member Center to revoke the certificate, as described in Revoking Certificates Using Member Center. (If you attempt to revoke a distribution certificate using Xcode, you are redirected to Member Center.)

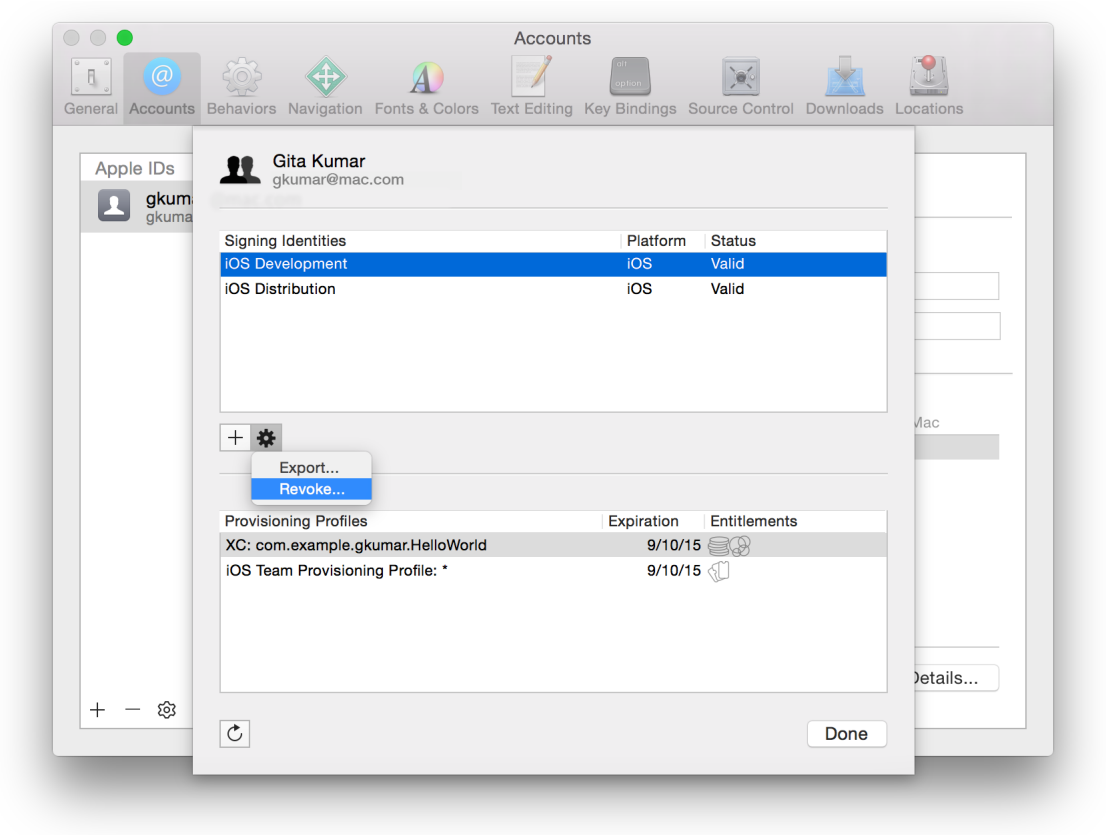
To revoke a development certificate using Xcode

1. Choose Xcode > Preferences, and click Accounts at the top of the window.

2. Select your team, and click View Details.



3. In the dialog that appears, select the development certificate you want to revoke, click the Action button (the gear icon to the right of the Add button) below the Signing Identities table, and choose Revoke from the pop-up menu.



4. In the dialog that appears, click Done.

Revoking Certificates Using Member Center

Use Member Center to revoke all types of certificates belonging to your team. For example, revoke development certificates for other team members or distribution certificates that you no longer need or want to re-create.

To revoke a certificate using Member Center

1. In Certificates, Identifiers & Profiles, select Certificates.
2. Under Certificates, select All.

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

4. In the dialog that appears, click Revoke.

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 178). Use Keychain Access or Member Center to view the expiration dates of your signing identities and certificates, as described in Verifying Using Member Center (page 160) and Verifying Using Keychain Access (page 160).

Re-Creating Certificates and Updating Related Provisioning Profiles

Re-creating certificates and updating related provisioning profiles isn't a simple task, because these assets are related and reside on both your Mac and in Member Center. If you revoke a certificate, any provisioning profile that contains that certificate becomes invalid. Xcode automatically regenerates team provisioning profiles for you, but you manage other types of provisioning profiles yourself. This section covers all the steps you perform to fully restore your code signing assets.

> **Note:** Xcode shows you the signing identities that are in your keychain, not all the certificates that you may have in Member Center.

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 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
- You use multiple Mac computers for development or belong to multiple teams, and you want to set up your Mac for a new project

However, if you're experiencing certificate, provisioning, or build issues, review Certificate Issues (page 241) first before performing these steps because removing your certificates is irreversible.

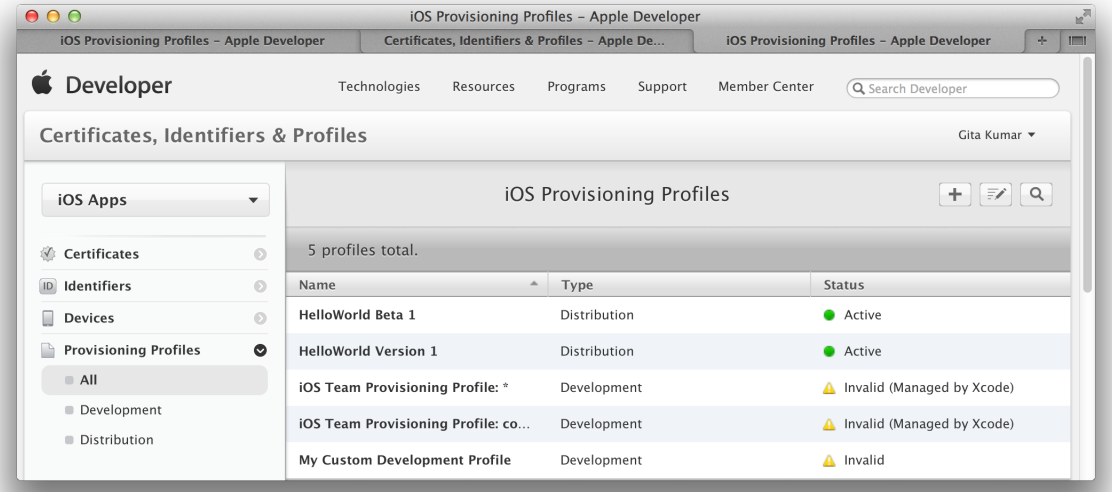
Choose the certificates you want to re-create. For example, if you experience problems running your app on a device, you may only need to re-create your development certificate. Keep in mind that re-creating a distribution certificate doesn't affect your development certificates or development provisioning 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 doesn't affect apps that you've submitted to the store nor does it affect your ability to update them.

To re-create your certificates and update related provisioning profiles

1. Revoke the certificates using Member Center, as described in Revoking Certificates (page 174).

Provisioning profiles containing a revoked certificate become invalid. For example, if you revoke your development certificate, all the development provisioning profiles containing that certificate become invalid:



2. If necessary, remove the signing identities for these certificates from your keychain, as described in Removing Signing Identities from Your Keychain (page 172).

   If you revoke your own development or distribution certificate, remove the corresponding signing identity from your keychain. Otherwise, the owner of the certificate should remove the signing identity on his or her Mac.

3. Optionally, request new certificates using Xcode, as described in Requesting Signing Identities (page 156).

If you revoke a development certificate, requesting a new development certificate or refreshing provisioning profiles in Xcode, as described in Refreshing Provisioning Profiles in Xcode (page 205), regenerates the team provisioning profiles and they no longer appear invalid in Member Center:

4. Remove or regenerate other types of provisioning profiles that contain the revoked certificates, as described in Editing Provisioning Profiles in Member Center (page 209).

   Xcode doesn't automatically regenerate distribution provisioning profiles or custom development provisioning profiles you create using Member Center.

5. If you changed provisioning profiles in Member Center, refresh your provisioning profiles in Xcode, as described in Refreshing Provisioning Profiles in Xcode (page 205).

6. If necessary, install the modified provisioning profiles on your devices, as described in Verifying and Removing Provisioning Profiles on Devices (page 212).

7. Once the certificates are repaired on the primary Mac, export your developer profile, as described in Exporting Your Developer Profile (page 166).

If you're repairing multiple Mac computers, perform these additional steps on the other Mac computers:

1. Remove the signing identities from your keychain, as described in Removing Signing Identities from Your Keychain (page 172).

2. Import your developer profile, as described in Importing Your Developer Profile (page 171), that you created on the original Mac.

Your Signing Certificates in Depth

Your code signing identities, stored in your keychain, represent your iOS and Mac program development and distribution credentials. Become familiar with the names of these certificates, because they appear in menus, and with 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're required to sign all iOS apps that run on devices and Mac apps that use certain app services 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're 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's 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 13-2 for the mapping between the type of the certificate, the name of the certificate as it appears in Keychain Access, and the purpose of each.

Member Center displays the team name (or person's name) and type for each certificate. Xcode Accounts preferences displays the type of certificate in the Signing Identities column. 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 so contain the team name. Individual developers are a one-person team, and so your name and the team name are the same.

Table 13-2 Certificate types and names

Certificate type	Certificate name	Description
iOS Development	iPhone Developer: *Team Member Name*	Used to run an iOS app on devices and use certain app services during development.

Certificate type	Certificate name	Description
iOS Distribution	iPhone Distribution: *Team Name*	Used to distribute your iOS app on designated devices for testing or to submit it to the App Store.
Mac Development	Mac Developer: *Team Member Name*	Used to enable certain app services during development and testing.
Mac App Distribution	3rd Party Mac Developer Application: *Team Name*	Used to sign a Mac app before submitting it to the Mac App Store.
Mac Installer Distribution	3rd Party Mac Developer Installer: *Team Name*	Used to sign and submit a Mac Installer Package, containing your signed app, to the Mac App Store.
Developer ID Application	Developer ID Application: *Team Name*	Used to sign a Mac app before distributing it outside the Mac App Store.
Developer ID Installer	Developer ID Installer: *Team Name*	Used to sign and distribute a Mac Installer Package, containing your signed app, outside the Mac App Store.

Recap

In this chapter, you learned how to maintain your development and distribution signing identities that you use throughout the lifetime of your app. You also learned how to identify the different types of certificates in Xcode, Keychain Access, and Member Center.

Maintaining Identifiers, Devices, and Profiles

You use Member Center to manage the identifiers, devices, and profiles used to code sign your app, enable it to launch on devices, and use certain app services. Xcode creates and manages these assets for you during development. When you're ready to distribute your app for testing, you use Member Center to manage some of these assets yourself. For example, you use Member Center to create your ad hoc and store provisioning profiles. Also, you might want to upload a list of devices used for testing, not register them individually. This chapter covers miscellaneous tasks you perform to maintain identifiers, devices, and profiles.

For how to use Website Push IDs, read *Notification Programming Guide for Websites*.

Registering App IDs

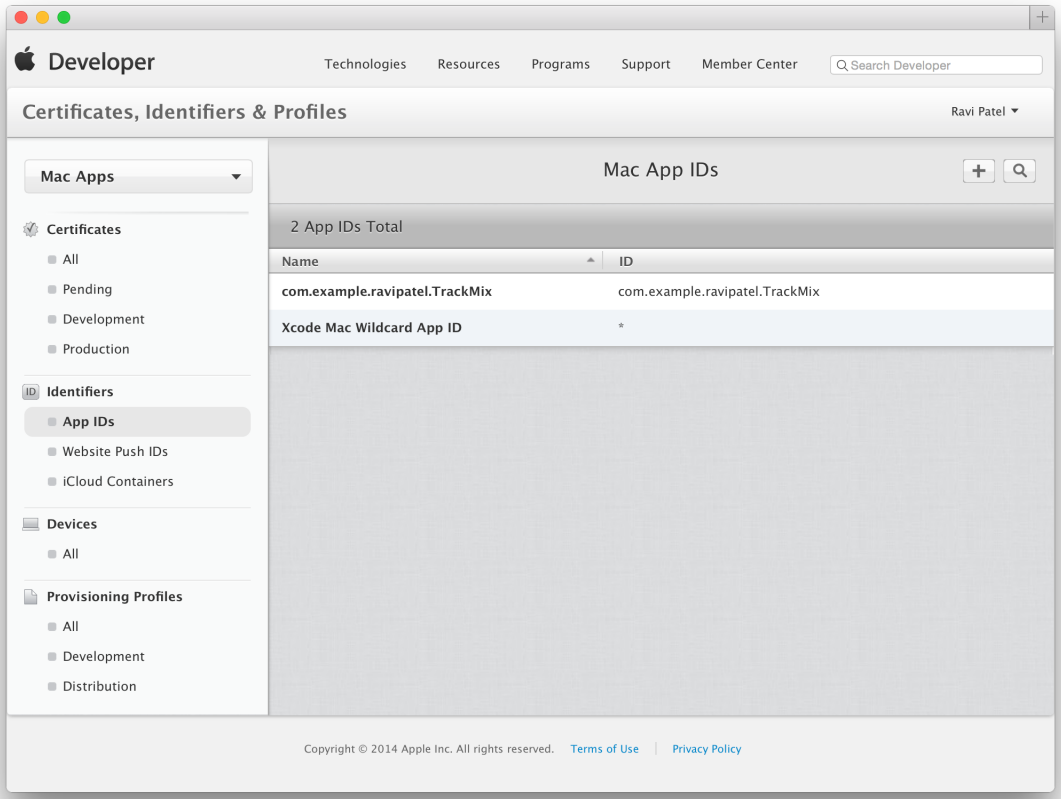
You can create a wildcard App ID that matches one or more apps or an explicit App ID that exactly matches your bundle ID. The app services enabled for an App ID serve as a whitelist of the services one or more apps may use. What services an app actually uses is configured in the Xcode project. You can enable app services when you create an App ID or modify these settings later. Game Center and In-App Purchase are enabled by default for an explicit iOS App ID. In-App Purchase is enabled by default for an explicit Mac App ID.

In most cases, the wildcard and explicit App ID that Xcode creates for you when you add capabilities to your app, as described in Adding Capabilities (page 48), are sufficient to develop, test, and distribute your app. There can be only one explicit App ID per app, so if one already exists, Xcode uses it to create team provisioning profiles on your behalf.

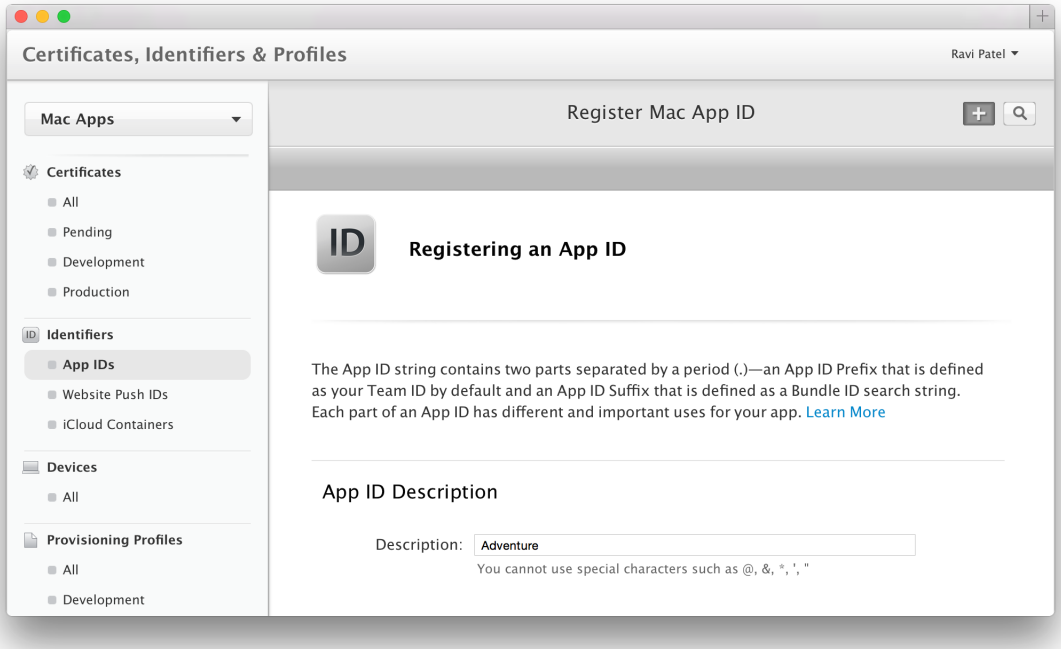
To register an App ID

1. In Member Center, select Certificates, Identifiers & Profiles.
2. Under Identifiers, select App IDs.

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

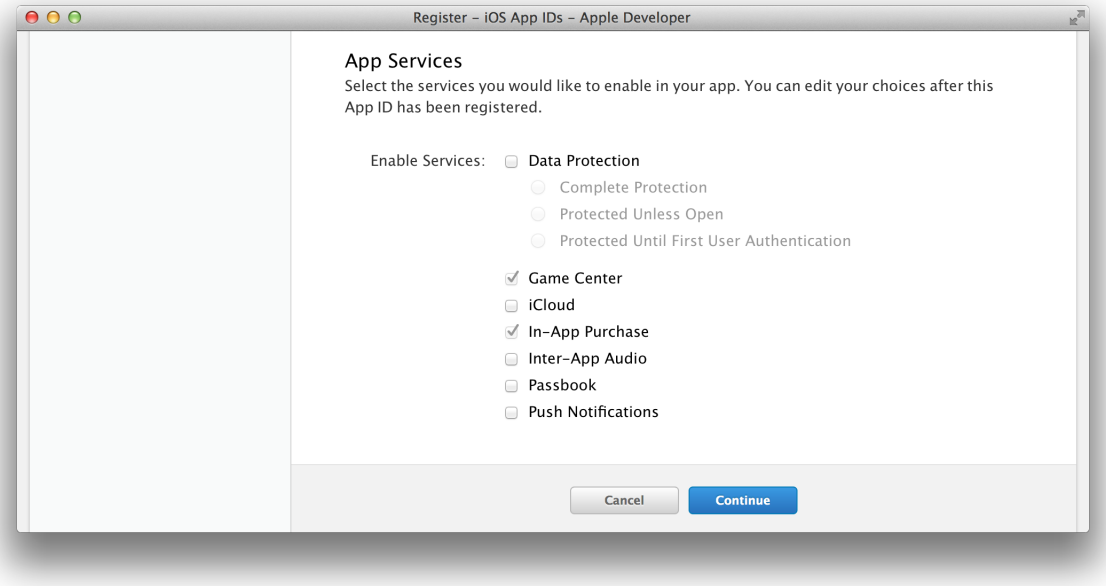


4. Enter a name or description for the App ID in the Description field.



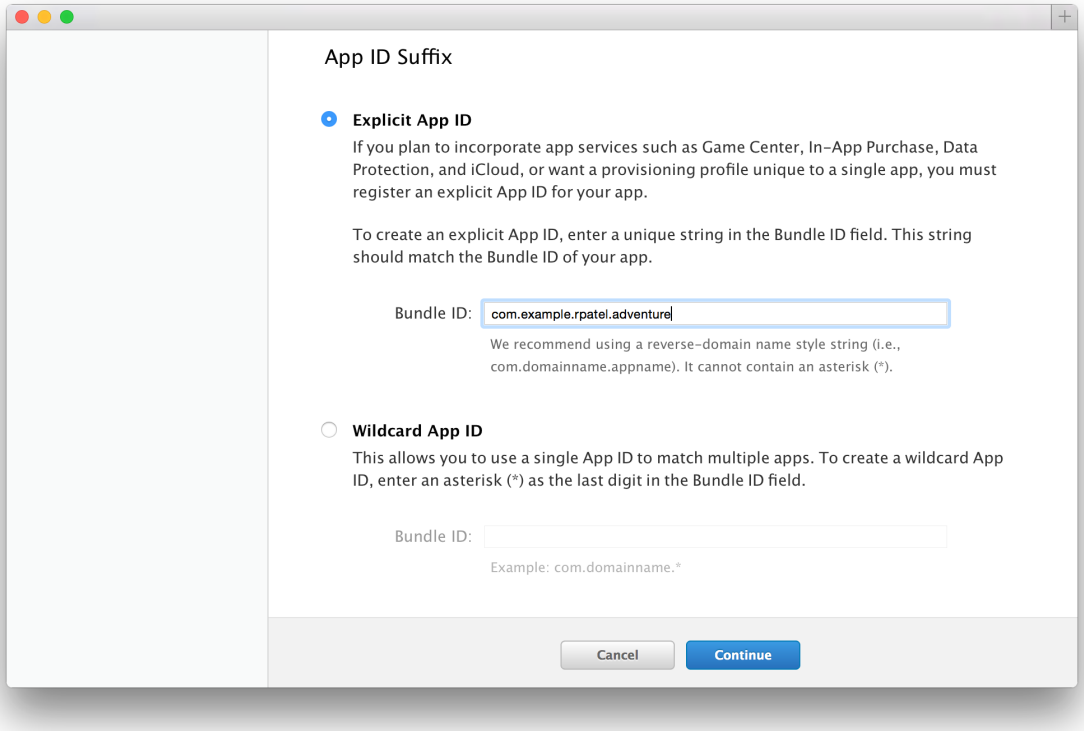
5. Select the corresponding checkboxes to enable the app services you want to use.

   A checkbox is disabled if the technology requires an explicit App ID and you're creating a wildcard App ID, or the technology is enabled by default.



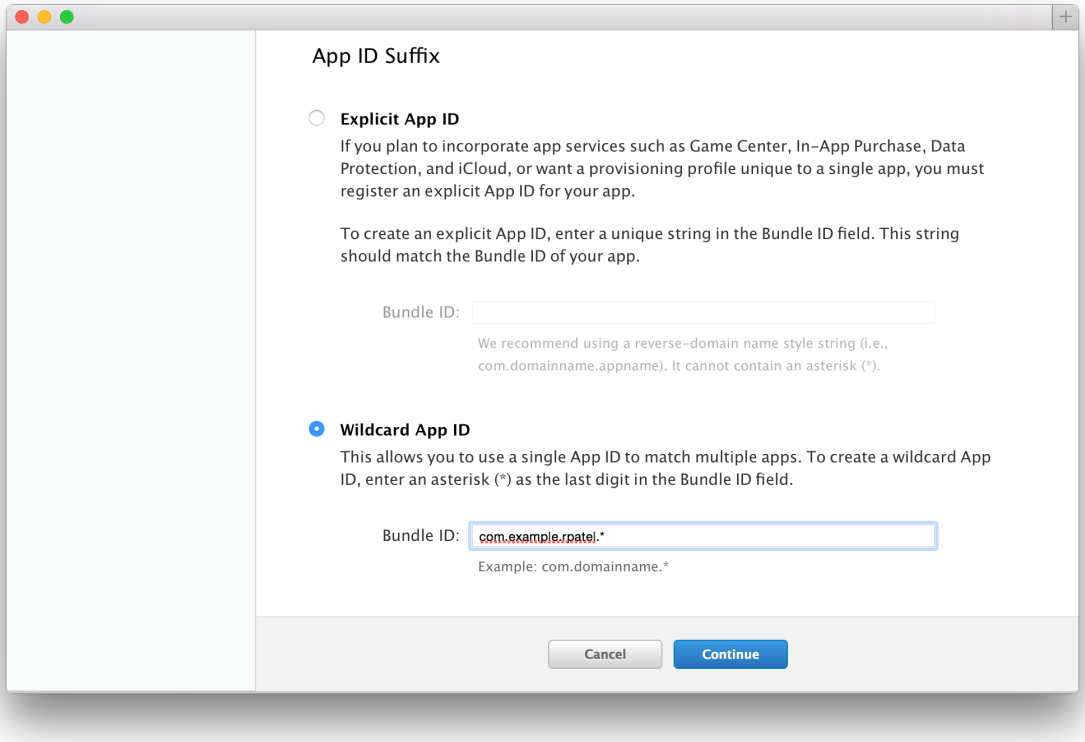
6. To create an explicit App ID, select Explicit App ID and enter the app's bundle ID in the Bundle ID field.

An explicit App ID exactly matches the bundle ID of an app you're building—for example, `com.example.gitakumar.adventure`. An explicit App ID can't contain an asterisk (*).



If you entered a bundle ID in the target's Summary pane in Xcode, the App ID you enter here should match your bundle ID.

7. To create a wildcard App ID, select Wildcard App ID and enter a bundle ID suffix in the Bundle ID field.



8. Click Continue.
9. Review the registration information, and click Submit.
10. Click Done.

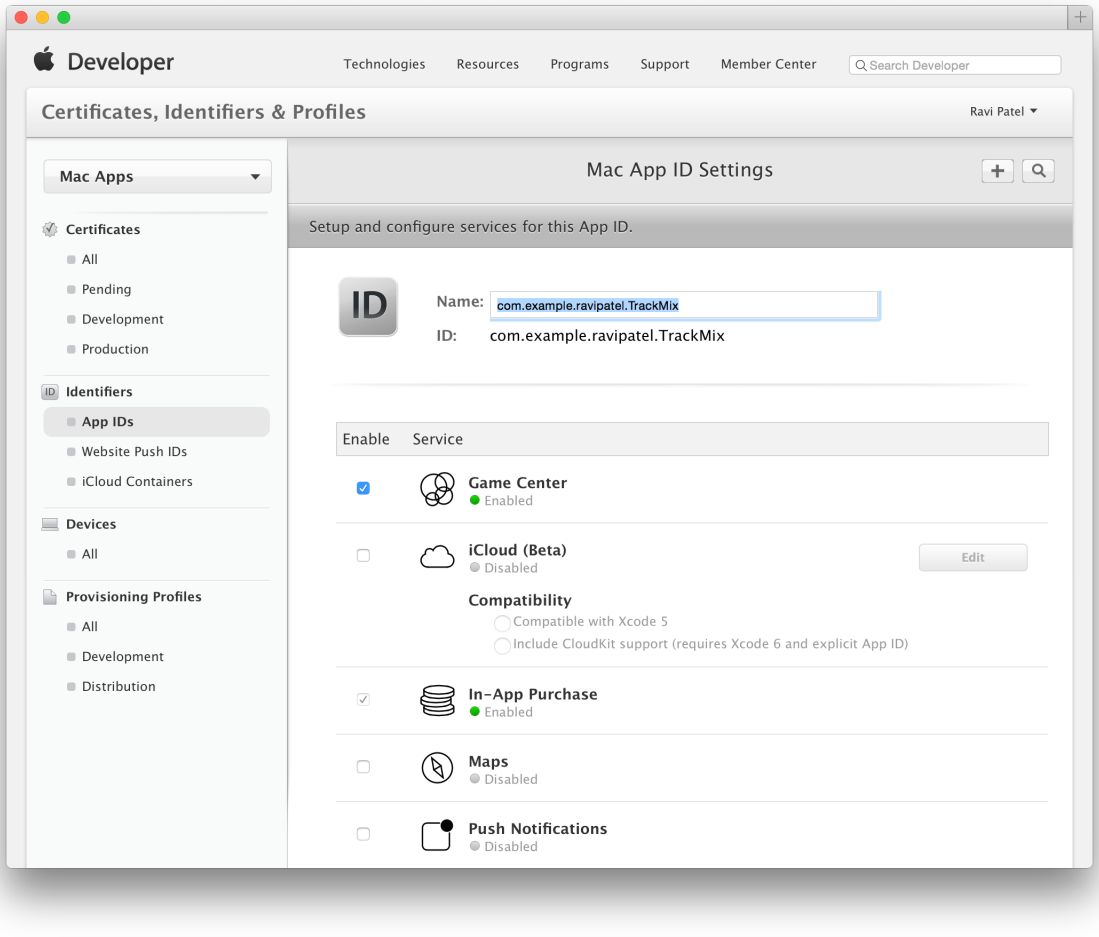
Editing App IDs

For app services you can't enable using Xcode, you can edit an App ID directly using Member Center. To enable app services for one or more apps, edit the corresponding App ID in Member Center.

To enable app services for an existing App ID

1. In Member Center, select Certificates, Identifiers & Profiles.
2. Under Identifiers, select App IDs.
3. Select the App ID you want to change, and click Edit.

4. Select the corresponding checkboxes to enable the app services you want to allow.



5. If a warning dialog appears, click OK.

   Later, you'll regenerate the provisioning profiles that use the App ID.

6. Click Done.

To fully enable push notifications, read Configuring Push Notifications (page 76).

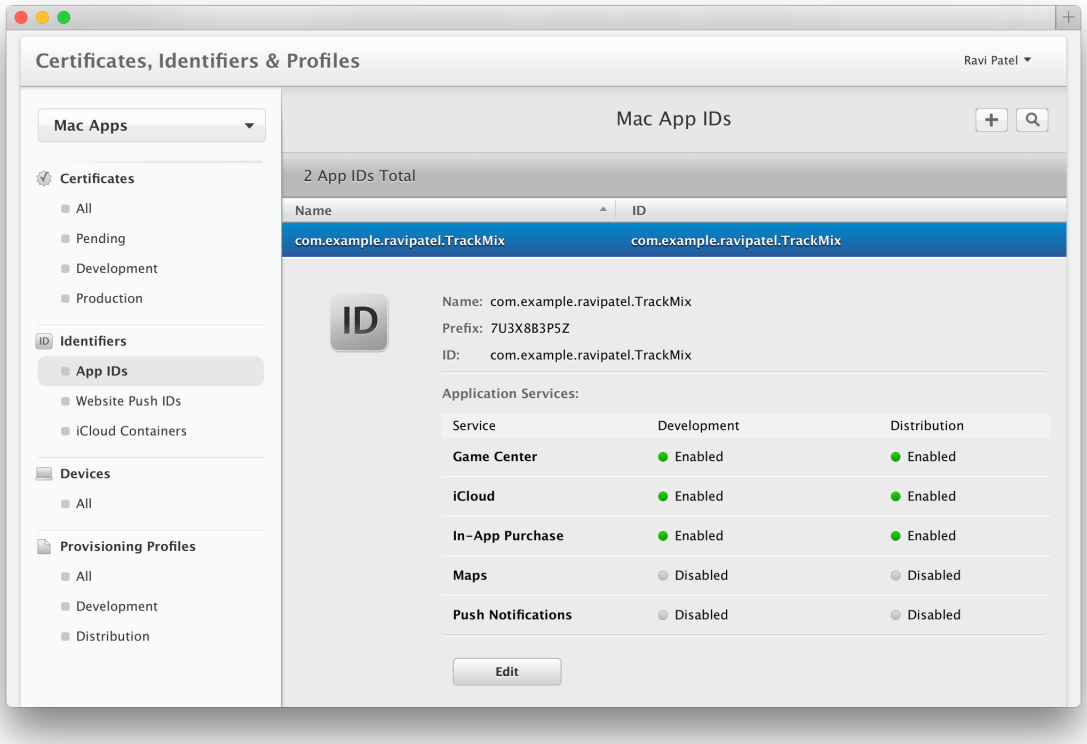
Deleting App IDs

You can also remove App IDs when you no longer need them. However, you cannot delete an explicit App ID for an app you submitted to the store.

To remove an App ID

1. In Member Center, select Certificates, Identifiers & Profiles.
2. Under Identifiers, select App IDs.

3. Select the App ID you want to delete, and click Edit.



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 in Member Center (page 209), or delete them.

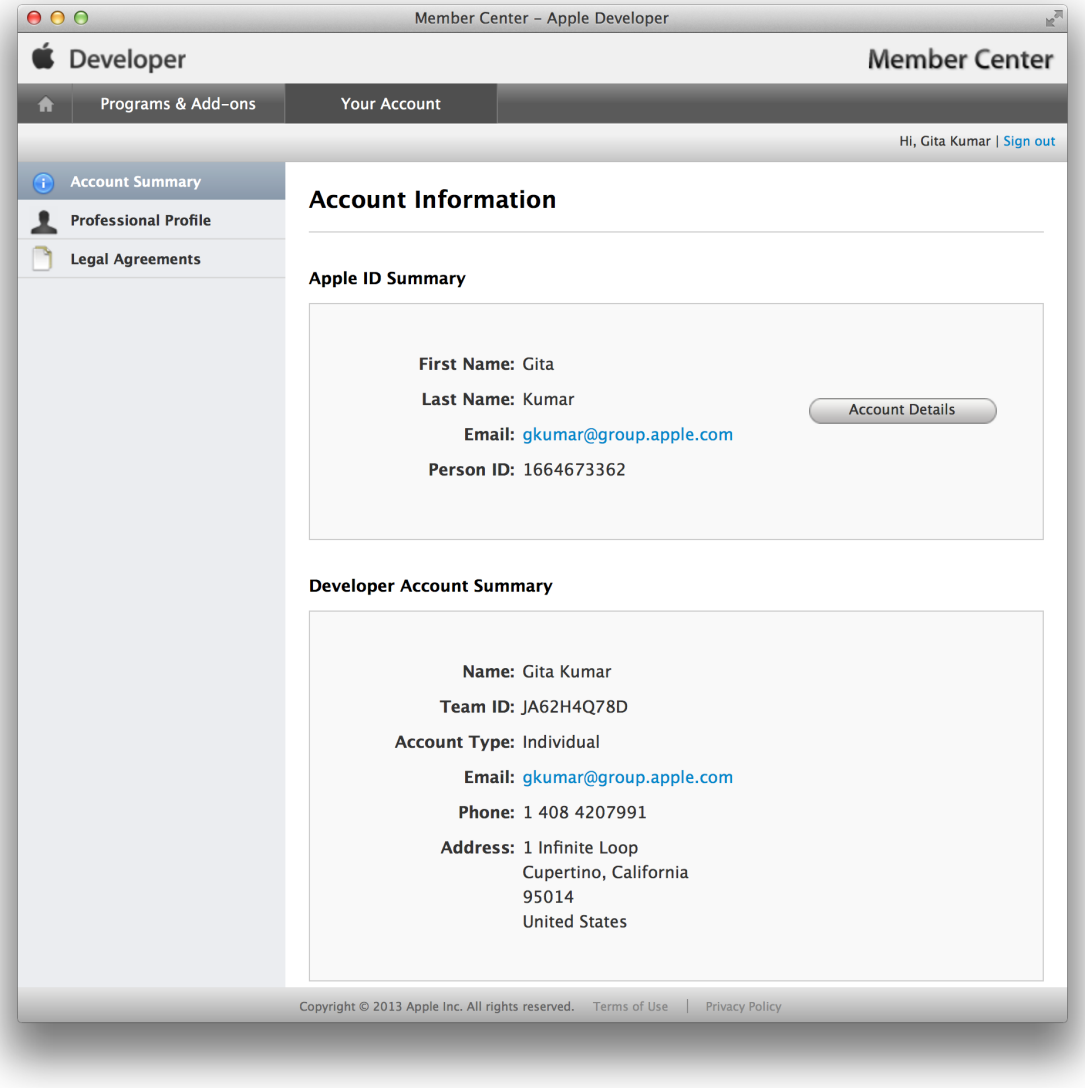
Locating Your Team ID

Occasionally, you may need to know your Team ID. For example, if you want to receive ownership of an app from another developer, the developer needs your Team ID to initiate the app transfer in iTunes Connect. The Team ID is a unique 10-character string generated by Apple that's assigned to your team. You can find your Team ID using Member Center.

To locate your Team ID

1. In Member Center, select Your Account.

Your Team ID appears in the Developer Account Summary section under the team name.



Registering Devices Using Member Center

In Member Center, you can register devices individually as needed or upload a file with information on multiple devices. Each year, you're allowed to register a fixed number of devices. The maximum number of devices you can register is 100 per membership year. If you later disable a device, as described in Disabling and Enabling Devices Using Member Center (page 197), it doesn't decrease your count of registered devices.

Locating Device IDs

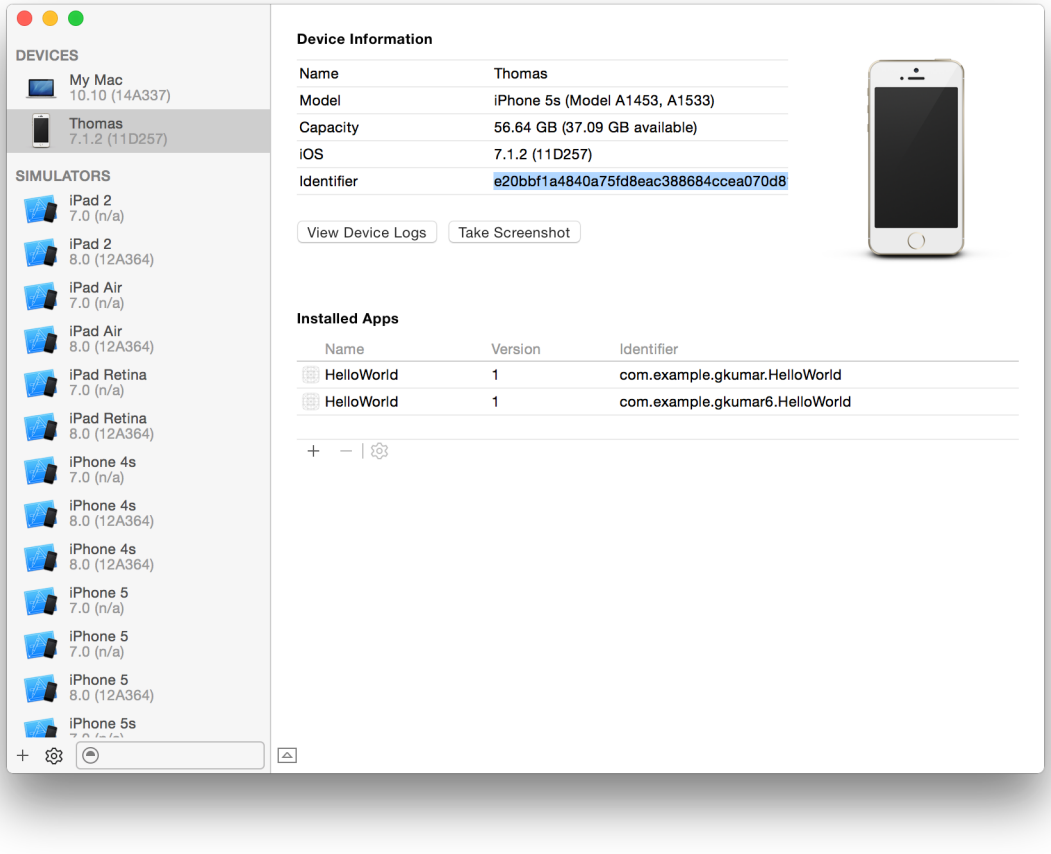
You need the name and device ID for each device you want to register. There are several ways to obtain device IDs depending on whether you have Xcode installed.

Locating Device IDs Using Xcode

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

To locate your device ID using Xcode

1. Choose Window > Devices.
2. In the Devices window under Devices, select your device (your Mac is a device too).



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

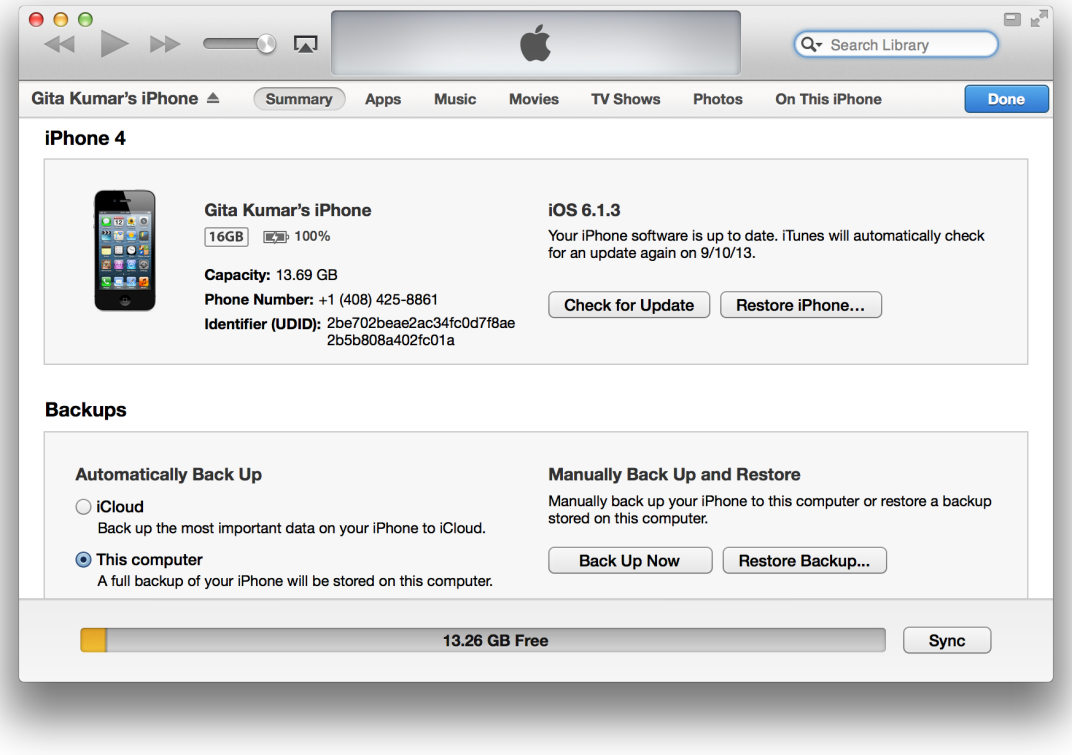
Locating iOS Device IDs Using iTunes

For iOS devices, 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 Capacity or Phone Number.

   The label Serial Number changes to UDID and displays the device ID.



5. Copy the device ID by Control-clicking the identifier and choosing Copy from the shortcut menu.
6. Paste the device ID in a document or an email message.

Locating Mac Device IDs Using System Preferences

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 isn't 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.

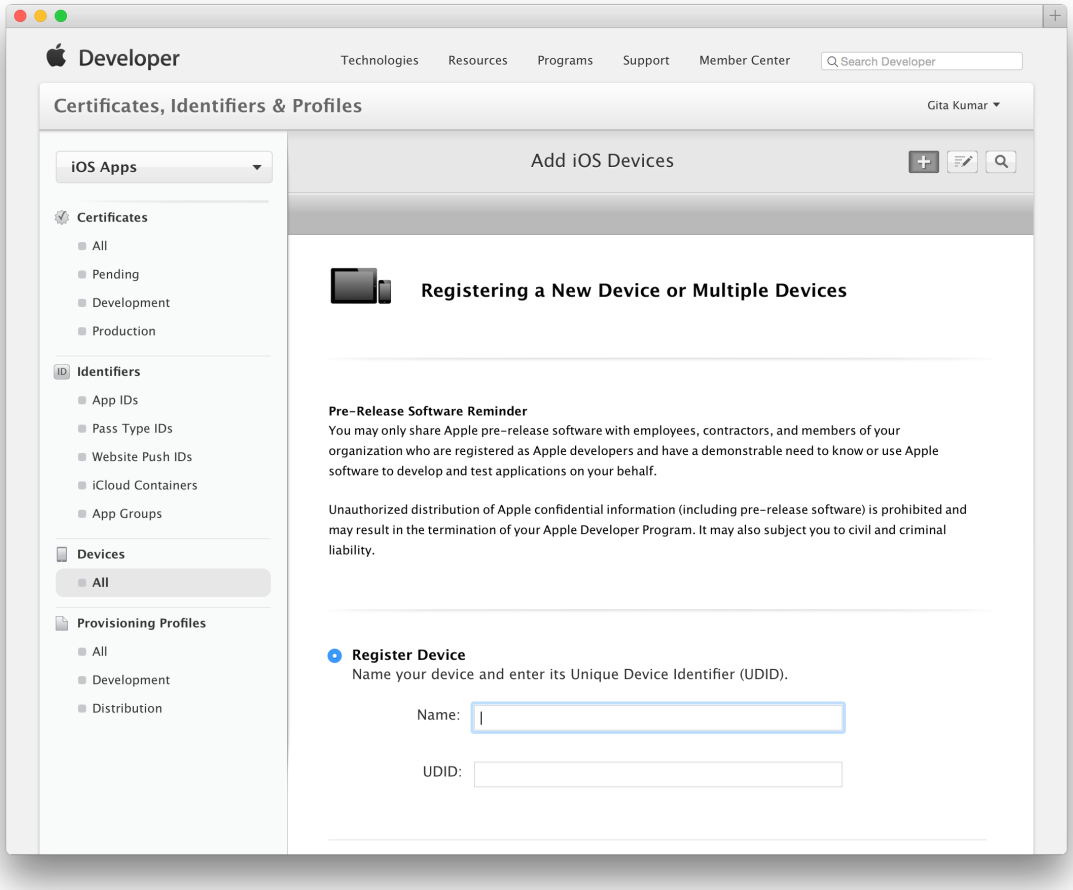
   The device ID, or hardware UUID, appears near the bottom of the Hardware Overview pane and is in the format XXXXXXXX–XXXX–XXXX–XXXX–XXXXXXXXXXXX.

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 Add button (+) in the upper-right corner.
4. Select Register Device.
5. Enter a device name and the device ID (UDID).



6. Scroll to the bottom of the page, and click Continue.
7. Review the registration information, and 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 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. Under Library, select Devices, and select the devices you want to add to the file.
4. Click Export in the toolbar.
5. Enter a filename in the Save As field.
6. Choose Device UDIDs from the “Export type” pop-up menu.
7. Click Save.

Creating a Plain Text Devices File

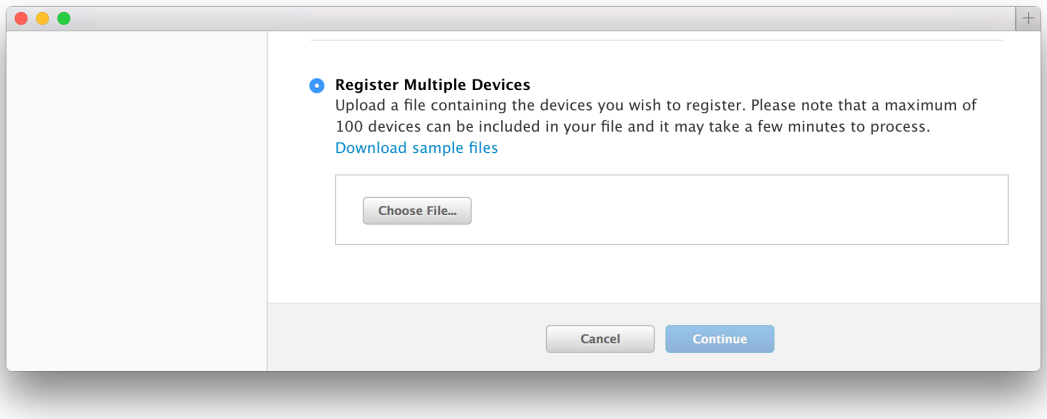
If you don’t have access to the testing devices, you can create a `.txt` file containing the device names and device IDs you collect using another method. In this case, create a tab-delimited file with one device ID and one device name in each row. You can use the first row for your headers, because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

To register multiple devices

1. In Certificates, Identifiers & Profiles, select Devices.
2. Under Devices, select All.
3. Click the Add 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, and click Submit.

Installing iOS Developer Previews on Your Device

Before you begin, download the iOS developer preview you want to install on your device from iOS Dev Center.

To install an iOS developer preview on your development device

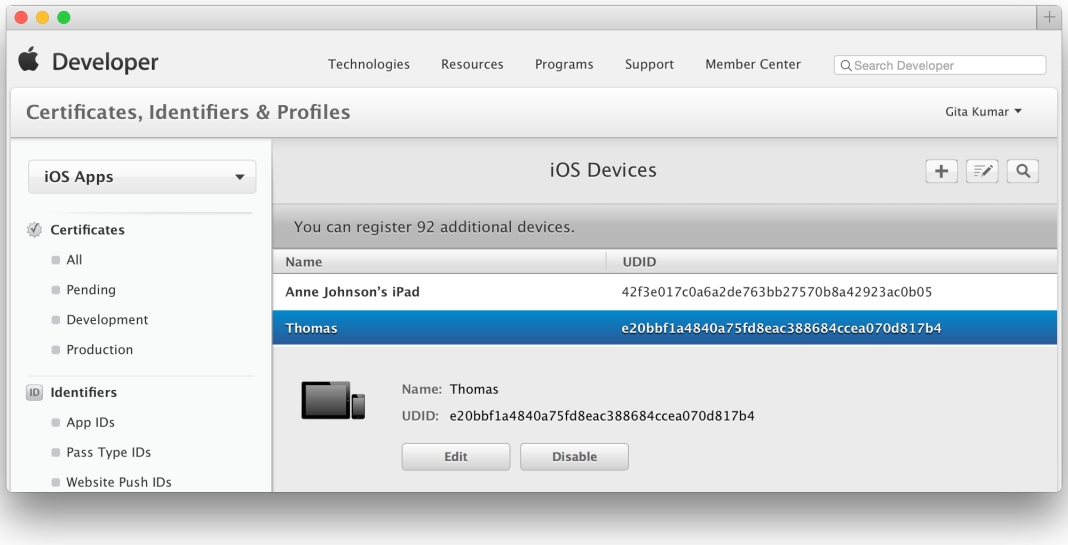
1. Connect the iOS device to your Mac.
2. In the iTunes, select the device in the upper-right corner.
3. In the Summary pane, Option-click the Restore iPhone/iPad/iPod button.
4. Select the iOS beta software restore image, and click Open to begin installation.
5. When installation is complete, activate the device and restore its contents using iTunes.

Disabling and Enabling Devices Using Member Center

You can disable and enable a device, but you can't delete it from Member Center. You can disable a device you no longer use for development or testing. However, doing so invalidates provisioning profiles that contain the device and doesn't increase your total count of devices for the year. You can also enable a device that you previously disabled.

To disable or enable a device

1. In Certificates, Identifiers & Profiles, select Devices.
2. Under Devices, select All.
3. Select the device you want to disable or enable.
4. Click either Enable or Disable.



5. In the dialog that appears, click either Enable or Disable again.

To regenerate invalid team provisioning profiles after disabling a device, read Refreshing Provisioning Profiles in Xcode (page 205). To remove a disabled device from other provisioning profiles you manage, read Editing Provisioning Profiles in Member Center (page 209).

You can also enable a device using Xcode. For iOS apps, Xcode automatically registers a connected device that's chosen from the Scheme toolbar menu in the project editor. For Mac apps, Xcode automatically registers the Mac that's running Xcode. If the device or Mac was previously registered but is disabled, Xcode enables it.

You may still run your app on a disabled device if it is in an older version of a provisioning profile that is installed on the device. To remove old versions of your provisioning profiles, read Verifying and Removing Provisioning Profiles on Devices (page 212).

Managing Apps on iOS Devices

In the Devices window, you can manage instances of your app installed on an iOS device. Before you begin, connect the iOS device to your Mac, choose Window > Devices, and select the device under Devices.

Removing Apps from iOS Devices

To remove all data files associated with an app on an iOS device, delete the app from the device. You might do this to test code that runs only when your app first launches. In Xcode, you can remove only the apps that you created. You can't remove any of the system-installed or third-party apps from the device using Xcode.

To remove an app from an iOS device

1. In the Installed Apps table, select an app and click the Delete button (–) below the table.
2. In the dialog that appears, click the Delete button.

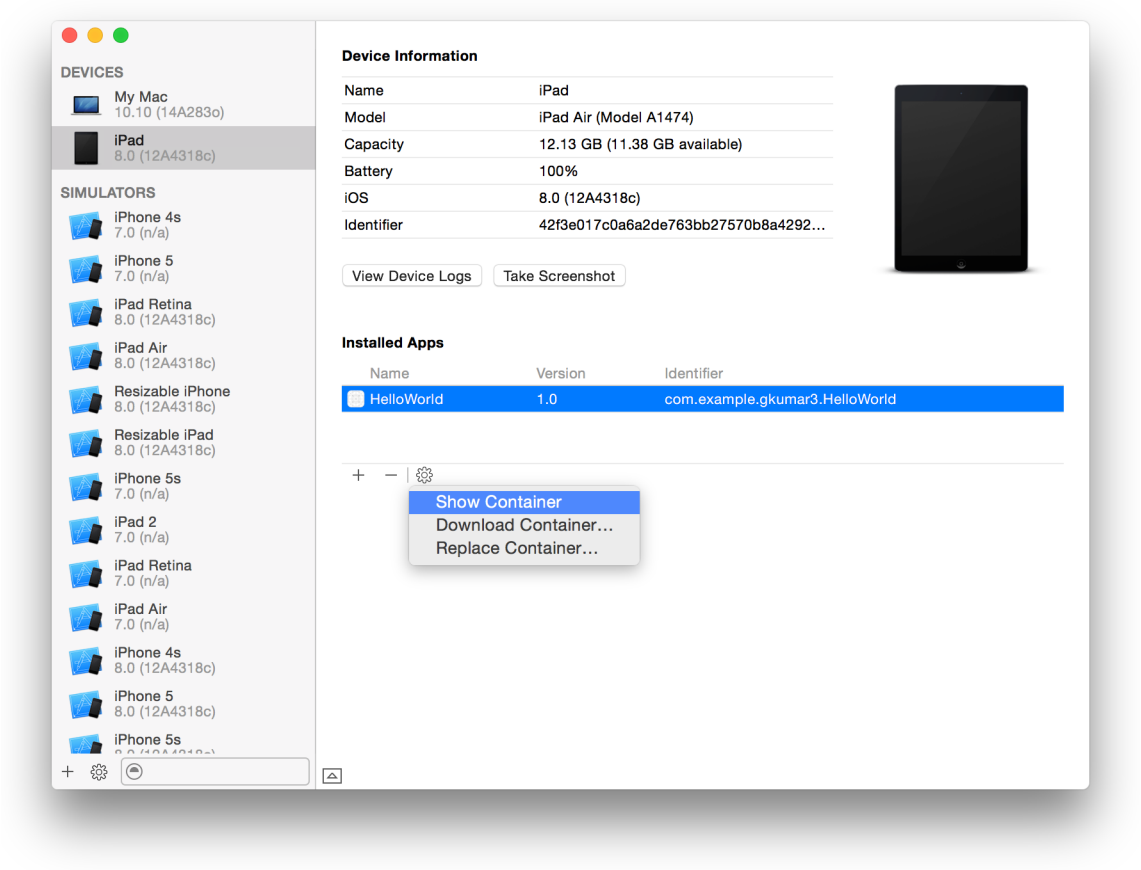
Viewing, Downloading, and Replacing App Containers on iOS Devices

You can view the file structure of an app container on an iOS device directly in Xcode but not the contents of the files.

To view the contents of an app container

1. Under Installed Apps, select the app from the list.

2. From the Action menu (the gear icon to the right of the Delete button), choose Show Container.



3. Click Done.

To examine or modify the files in an app container, download it to your file system. Analyzing the contents of an app's container may provide important information to help you diagnose problems or tune the app.

To download the contents of an app container

1. Under Installed Apps, select the app from the list.
2. From the Action pop-up menu, choose Download Container.
3. Enter a filename in the Save As text field, and click Save.

To test the app in a particular state, replace the app container on an iOS device.

To replace an app container on an iOS device

1. Under Installed Apps, select the app from the list.
2. From the Action menu, choose Replace Container.
3. Select a container from the File Picker and click Open.

Creating Provisioning Profiles Using Member Center

You can create both development and distribution provisioning profiles yourself using Member Center.

Because Xcode creates and manages team provisioning profiles for you, you only create a development provisioning profile if you want to restrict development of an app to specific team members and devices. Follow the steps in Creating Development Provisioning Profiles (page 200) if you want to create your own development provisioning profile.

Xcode creates an iOS ad hoc provisioning profile for you when you export an app archive and select the ad hoc deployment option, as described in Distributing Your App Using Ad Hoc Provisioning (page 104). To create an ad hoc provisioning profile directly in Member Center, read Creating Ad Hoc Provisioning Profiles (page 201).

Similarly, Xcode creates a store provisioning profile for you when you submit your app to the store. To create a store provisioning profile directly in Member Center, read Creating Store Provisioning Profiles (page 203).

After creating provisioning profiles, refresh provisioning profiles in Xcode, as described in Refreshing Provisioning Profiles in Xcode (page 205). The provisioning profiles should appear in the Provisioning Profiles table in the view details dialog in Accounts preferences.

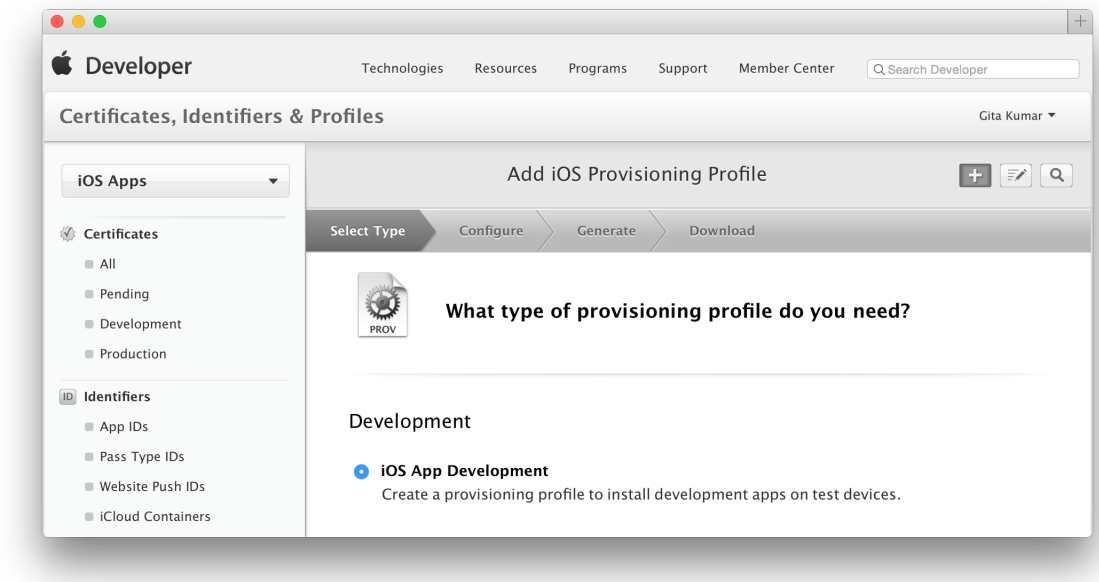
Creating Development Provisioning Profiles

Before creating a development provisioning profile, verify that you have an App ID, one or more development certificates, and one or more devices. If you want to register your own App ID, read Registering App IDs (page 184). (You can also use one of the App IDs that Xcode manages for you.) If you need to create your development certificate, read Requesting Signing Identities (page 156). If you need to register devices, read Registering Devices Using Member Center (page 191).

To create a development provisioning profile

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

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



4. Select the App ID you want to use for development, and click Continue.
5. Select one or more development certificates, and click Continue.
6. Select one or more devices, and click Continue.
7. Enter a profile name, and click Generate.
8. Click Done.

Creating Ad Hoc Provisioning Profiles

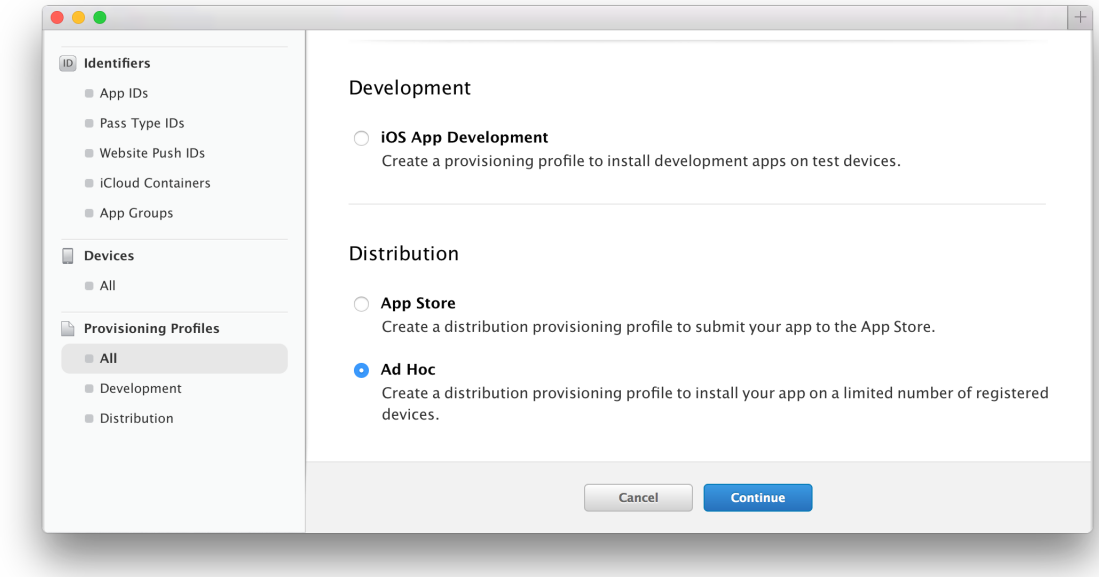
An ad hoc provisioning profile allows testers to run your app on their device without needing Xcode. To create an ad hoc provisioning profile, you select an App ID, a single distribution certificate, and multiple test devices.

> **Important:** If you don't have a distribution certificate, create one using Xcode, as described in Requesting Signing Identities (page 156), before continuing.

To create an ad hoc provisioning profile

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

3. Select Ad Hoc as the distribution method, and click Continue.



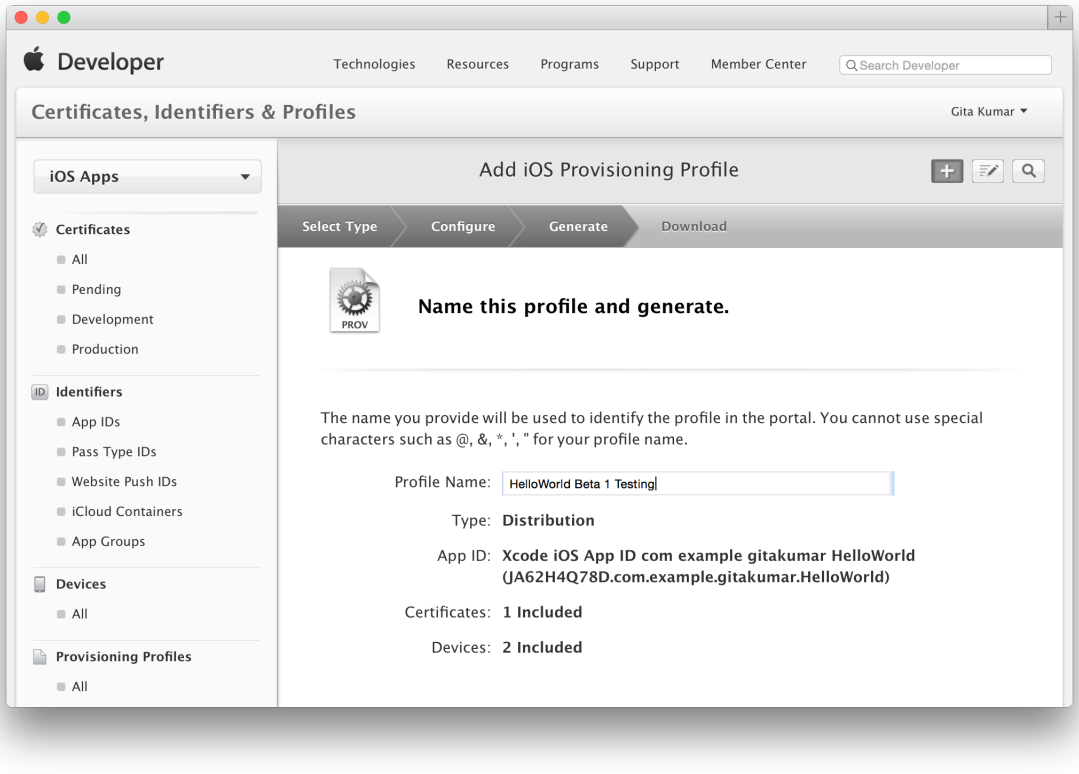
4. Choose the App ID you used for development, which matches your bundle ID, from the App ID pop-up menu, and click Continue.

   If you used a team provisioning profile during development and the menu contains only the Xcode iOS Wildcard App ID, select it. If the menu contains another Xcode-managed explicit App ID (it begins with “Xcode” and contains your bundle ID), select that App ID. If you created your own App ID, select that one.

5. Select the distribution certificate you want to use, and click Continue.

6. Select the devices you want to use for testing, and click Continue.

7. Enter a profile name, and click Generate.



Wait while Member Center generates the provisioning profile.

8. At the bottom of the page, Click Done.

In Xcode, refresh the provisioning profiles to download the ad hoc provisioning profile, as described in Refreshing Provisioning Profiles in Xcode (page 205). The ad hoc provisioning profile should now appear in the Provisioning Profiles table in the view details dialog in Accounts preferences.

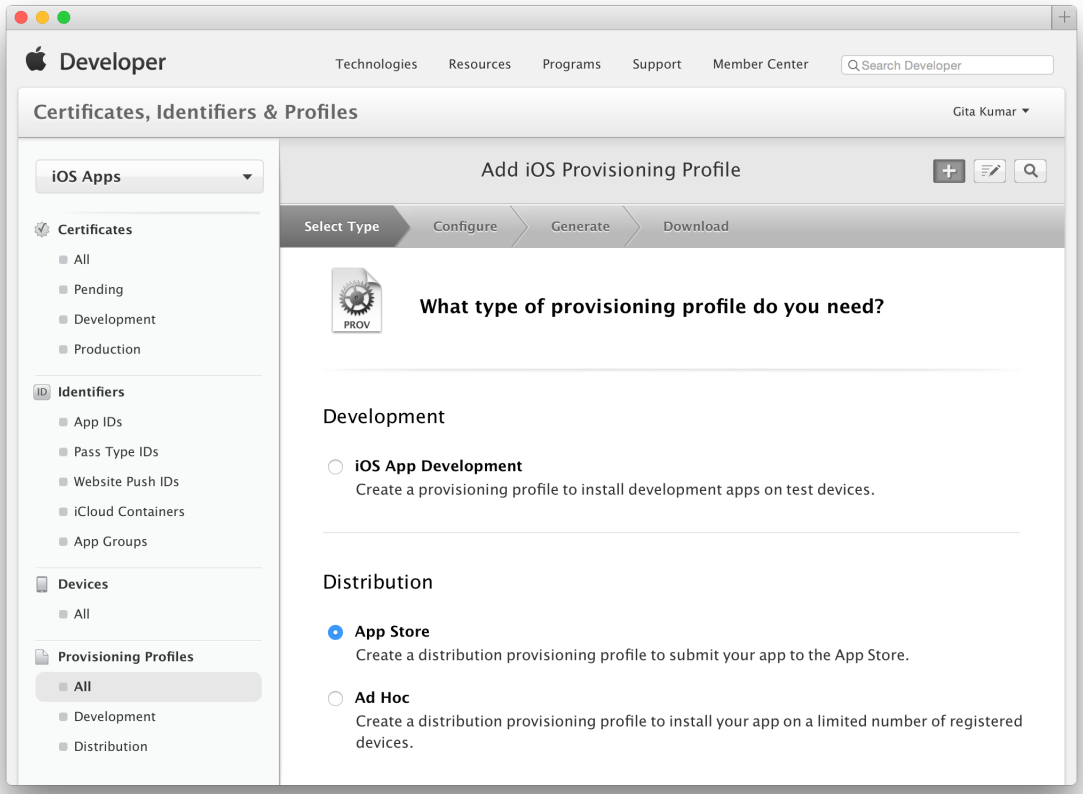
Creating Store Provisioning Profiles

Before uploading your app to the store, you provision it using a store provisioning profile. (For Mac apps that don't enable any app services, you can code sign your app using just a distribution certificate.) You don't select any devices to create a store provisioning profile.

To create a store provisioning profile

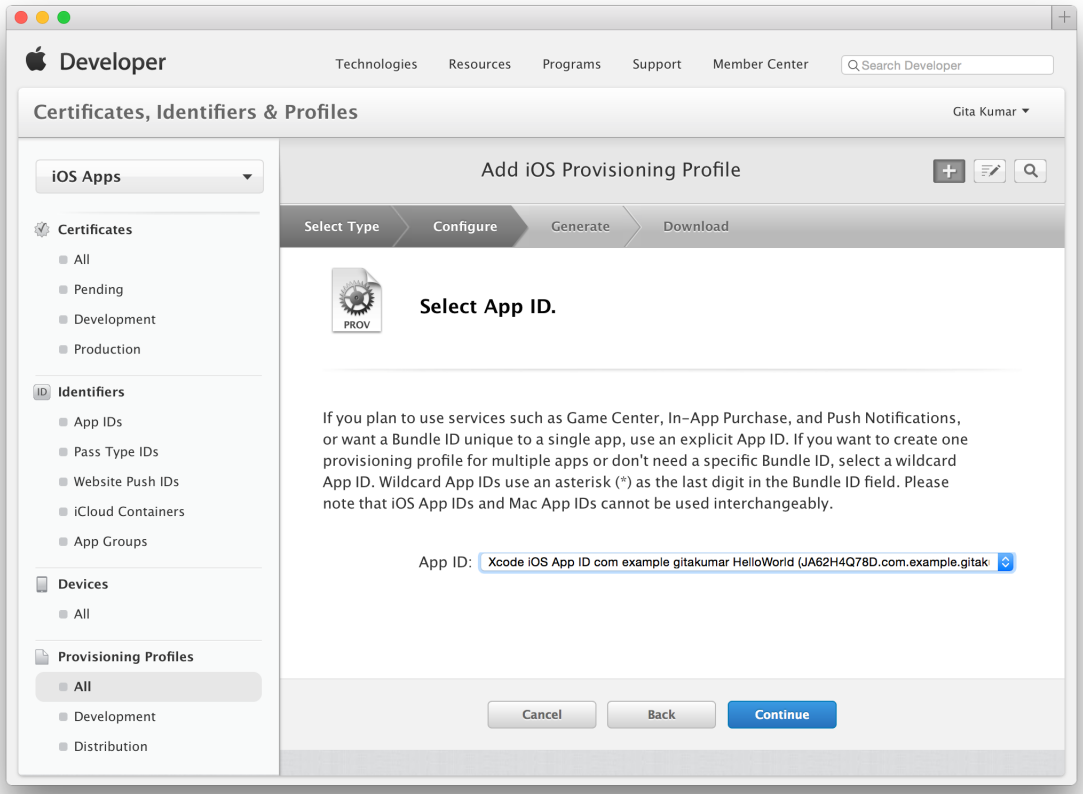
1. In Certificates, Identifiers & Profiles, select Provisioning Profiles.
2. Click the Add button (+) in the upper-right corner.
3. Select App Store for iOS apps or Mac App Store for Mac apps as the distribution method, and click Continue.

The screenshot below shows App Store selected for an iOS app.



4. Choose the App ID you used for development (the App ID that matches your bundle ID) from the App ID pop-up menu, and click Continue.

If you used a team provisioning profile during development and the menu contains only the Xcode Wildcard App ID, select it. If the menu contains an Xcode-managed explicit App ID (it begins with “Xcode” and contains your bundle ID), select that App ID. If you created your own App ID, select that one.



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.

   Wait while Member Center generates the provisioning profile.

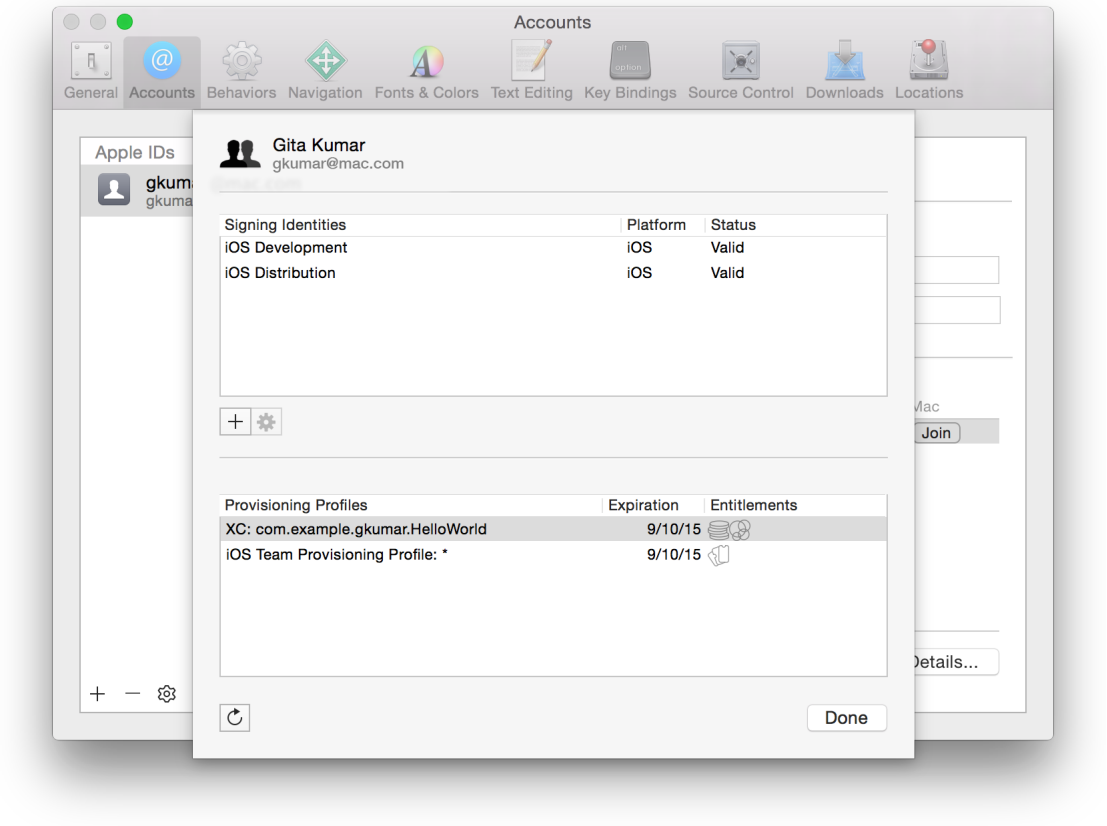
7. At the bottom of the page, Click Done.

Refreshing Provisioning Profiles in Xcode

Changes you make in Member Center aren’t automatically reflected in Xcode. For example, if you create or edit a provisioning profile in Member Center, refresh provisioning profiles in Xcode to download the changes. If you revoke a development certificate, refresh provisioning profiles to regenerate team provisioning profiles managed by Xcode. When Xcode refreshes provisioning profiles, it also checks to see if you’re missing signing identities and may ask if you want to request them.

To refresh provisioning profiles in Xcode

1. In the Xcode Preferences window, click Accounts.
2. Select your team, and click View Details.



3. In the dialog that appears, click the Refresh button in the lower-left corner under the Provisioning Profiles table.

   Xcode updates the list of profiles in the Provisioning Profiles table.

Using Custom Provisioning Profiles

Occasionally, you may want to use your own custom development provisioning profile instead of the team provisioning profile that Xcode manages for you. For example, you might do this if you want to limit development of an app to a subset of developers or if you're testing different app configurations. If so, create a development provisioning profile, as described in Creating Provisioning Profiles Using Member Center (page 200), and set your code signing identity build setting to use the new profile.

When you build the app, you code sign it with the signing identity matching the 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. Choosing 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 isn't 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 can't 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. For a description of each type of certificate that may appear in this menu, refer to Table 13-2 (page 182).

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're consistent. For multiple targets that use the same code signing identity, set this build setting at the project level. For multiple targets that use different code signing identities, you 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 Provisioning Profile build setting to your development profile and the Code Signing Identity build setting to your development certificate.

To set the code signing identity to your development certificate

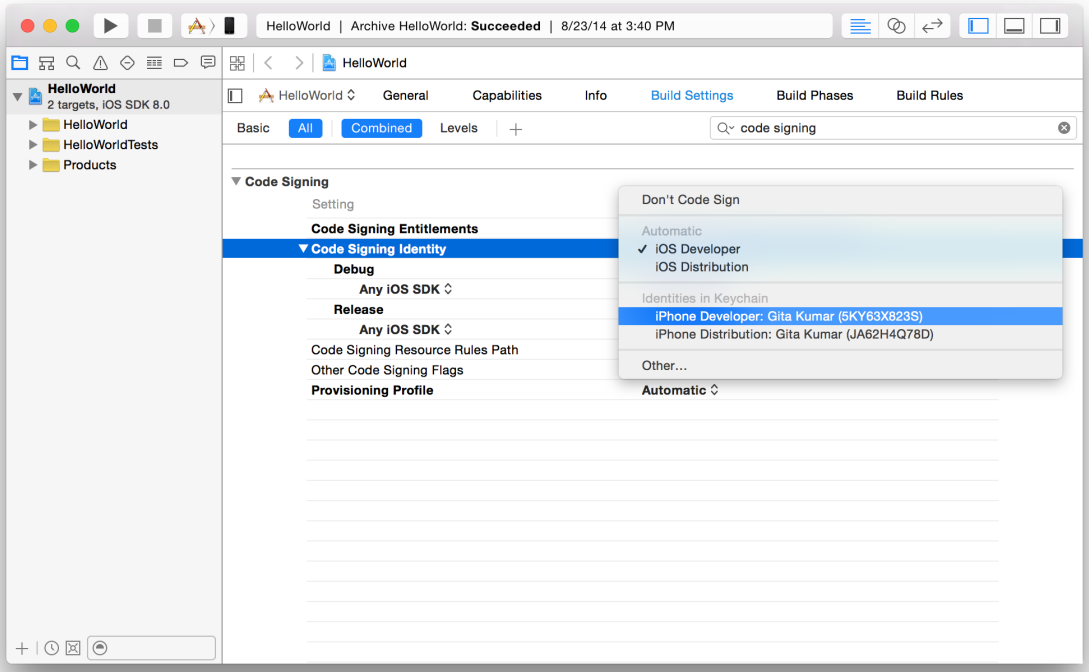
1. In the Xcode project editor, select the target.

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

2. Click Build Settings.
3. In the Build Settings pane, click All and type `code signing` in the search field.
4. From the Provisioning Profile pop-up menu, choose your development provisioning profile.

   Xcode automatically sets the Code Signing Identity build setting to "iOS Developer" for iOS apps and "Mac Developer" for Mac apps.
5. If necessary, from the Code Signing Identity pop-up menu, choose your development certificate.

For iOS apps, choose the certificate in the provisioning profile menu item that begins with the text “iPhone Developer:” followed by your name. For Mac apps, choose the certificate in the provisioning profile menu item that begins with the text “Mac Developer:” followed by your name.



Your app is code signed the next time you build 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 a Connected Device (page 88) to sign your app and launch it on a device.

To use the team provisioning profile again later, change the Provisioning Profile build setting to None. To learn more about Apple’s code signing technology, read *Code Signing Guide*.

Troubleshooting

If the development provisioning profile doesn’t appear in the Provisioning Profile pop-up menu, refresh provisioning profiles, as described in Refreshing Provisioning Profiles in Xcode (page 205). Then try to set the Provisioning Profile build settings 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 245).

Using Xcode-Managed Provisioning Profiles

Xcode creates and updates code signing identities and provisioning profiles for you. However, if your project was created using Xcode 4 or earlier, you may need to set the Provisioning Profile build setting to Automatic to use this feature.

To use Xcode-managed provisioning profiles

1. In the Xcode project editor, select the target.
2. Click Build Settings.
3. In the Build Settings pane, click All and type `Code Signing` in the search field.
4. From the Provisioning Profile pop-up menu, choose Automatic.

   Xcode sets the highest Code Signing Identity build setting to Don't Code Sign, and the "Any … SDK" Code Signing Identity build settings to iOS Developer (for iOS apps) and Mac Developer (for Mac apps).

If necessary, set the project-level Provisioning Profile build setting to Automatic. To troubleshoot provisioning profiles, read Provisioning Issues (page 244).

Editing Provisioning Profiles in Member Center

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.

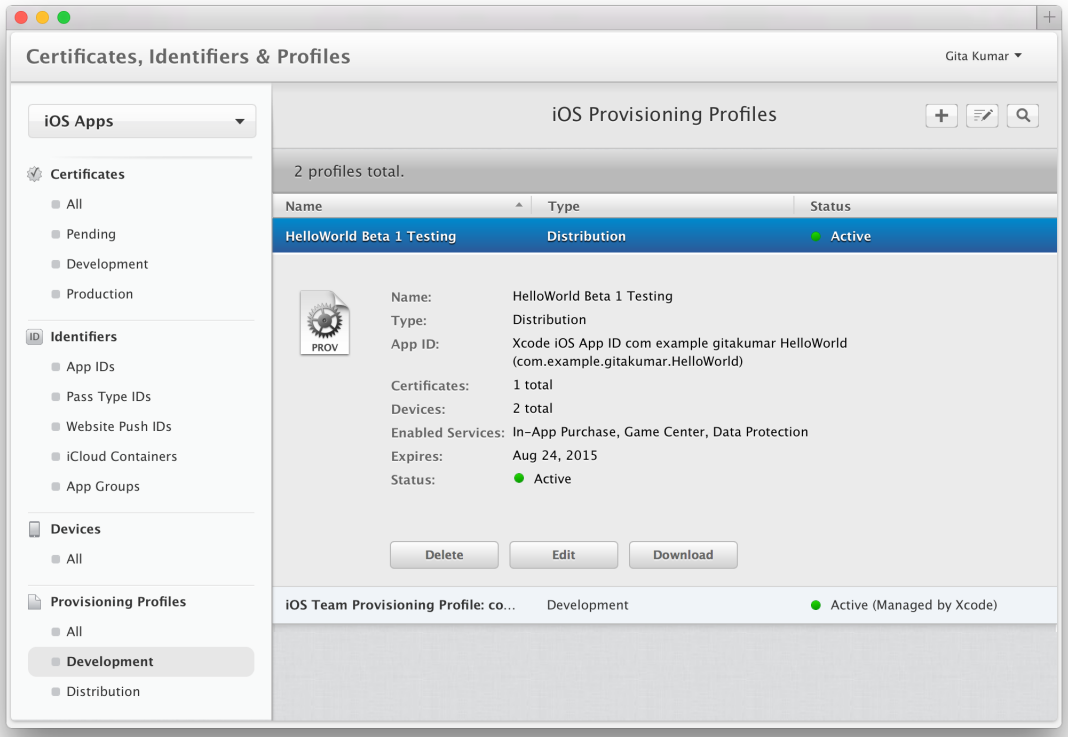
If you want to repair a provisioning profile because you re-created a certificate, follow the steps in Re-Creating Certificates and Updating Related Provisioning Profiles (page 178).

> **Note:** You can't modify the team provisioning profile managed by Xcode.

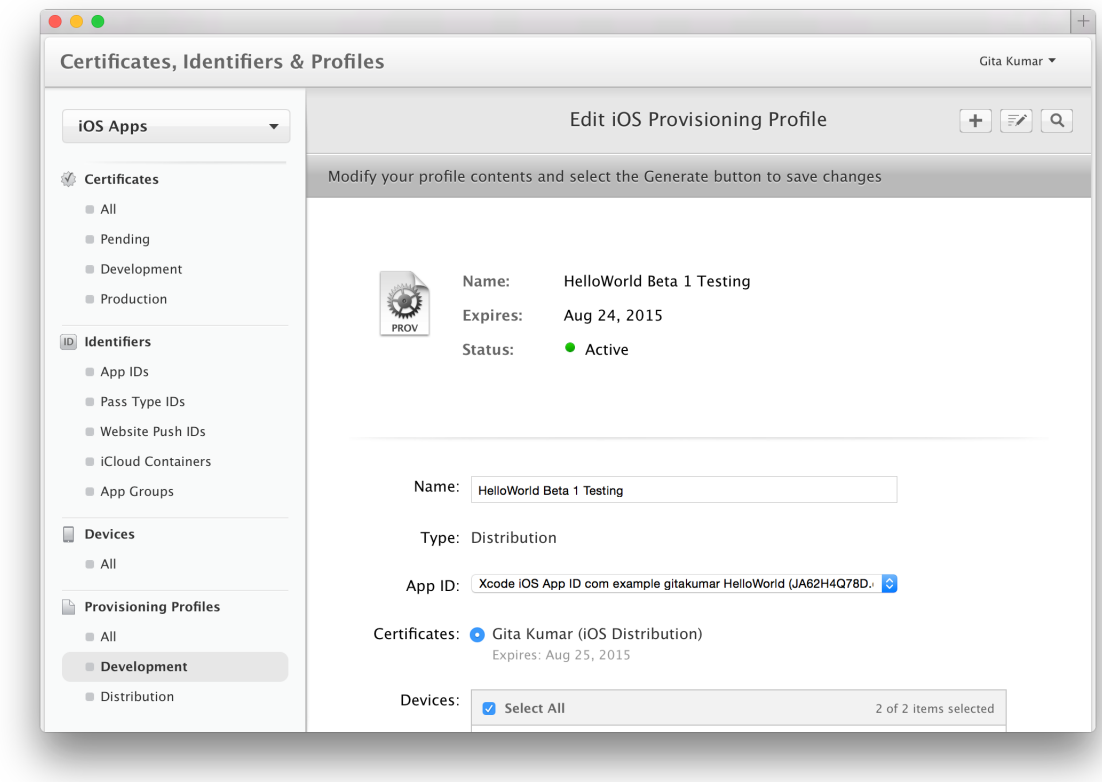
To edit a provisioning profile

1. In Certificates, Identifiers & Profiles, select Provisioning Profiles.
2. Under Provisioning Profiles, select All, Development, or Distribution.

3. Select the provisioning profile you want to modify, and 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 doing so, refresh provisioning profiles in Xcode, as described in Refreshing Provisioning Profiles in Xcode (page 205).

Renewing Expired Provisioning Profiles

If a provisioning profile expires, the provisioning profile's status displays Expired in Member Center. You renew an expired provisioning profile by editing and re-generating it, as described in Editing Provisioning Profiles in Member Center (page 209). You don't need to make any changes to the provisioning profile. Just scroll to the bottom of the Edit iOS Provisioning Profile or the Edit Mac Provisioning Profile page, and click Generate.

If the expired provisioning profile is installed on your device, remove it, as described in Verifying and Removing Provisioning Profiles on Devices (page 212). If the provisioning profile is an ad hoc provisioning profile, re-sign and distribute your app using the regenerated provisioning profile, as described in Beta Testing iOS Apps (page 97).

Verifying and Removing Provisioning Profiles on Devices

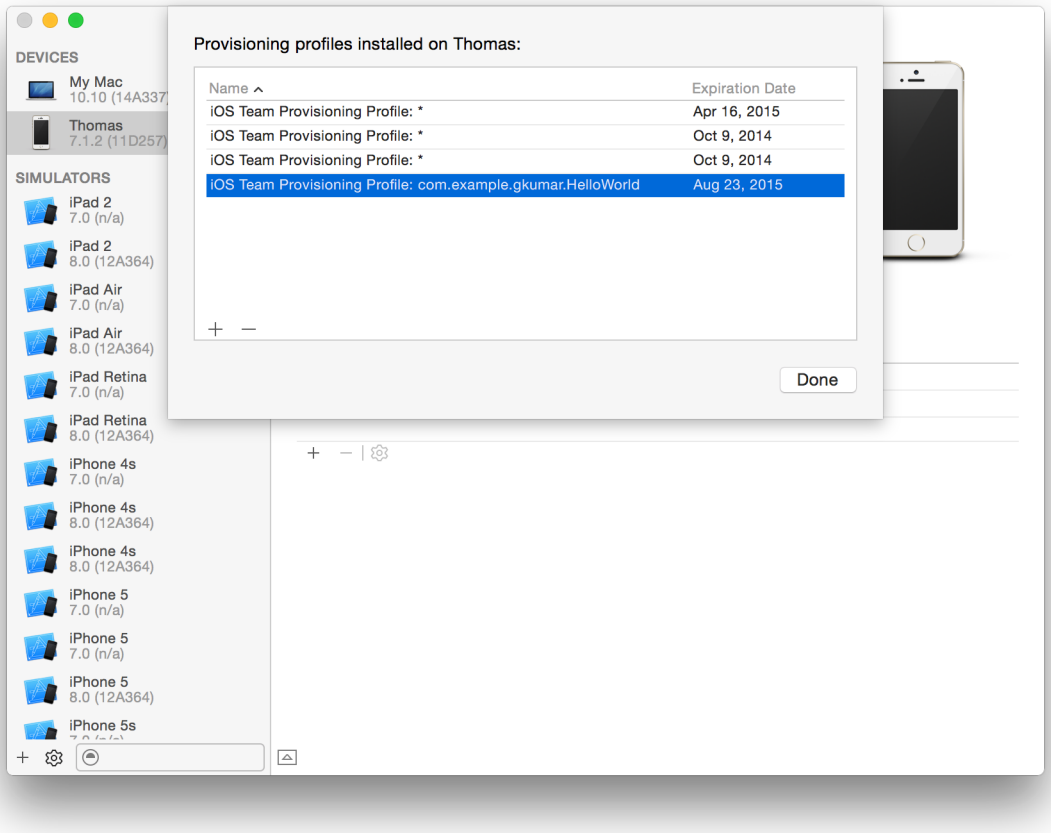
Use Xcode to verify if your provisioning profile is installed on a device or to remove expired provisioning profiles from a device.

> **Note:** You rarely install a provisioning profile yourself because when you launch an app on a device, iOS and OS X automatically install the embedded provisioning profile in the app's bundle on the device.

To verify or remove a provisioning profile

1. For iOS apps, connect the iOS device to your Mac.
2. Choose Window > Devices, and select the device under Devices.
3. At the bottom of the left column, click the Action button (the gear icon to the right of the Add button).
4. Choose Show Provisioning Profiles from the pop-up menu.

   The provisioning profiles sheet appears.



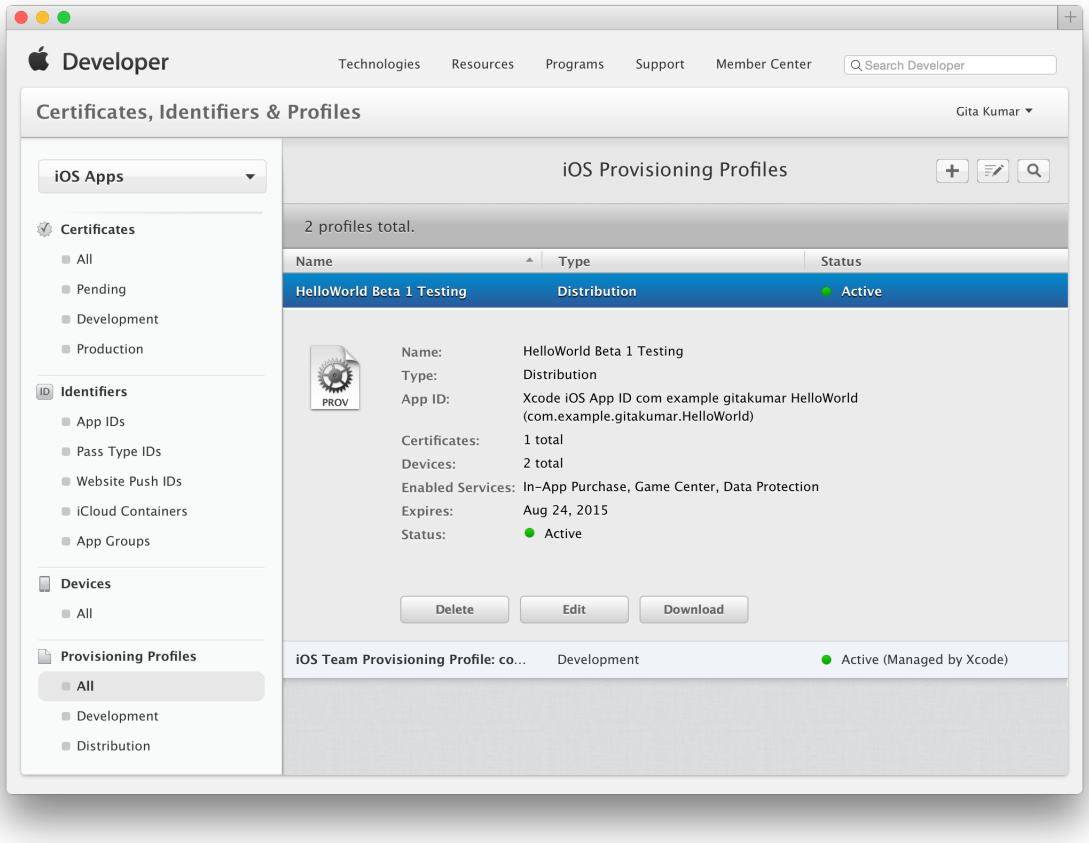
5. To remove a provisioning profile, select it from the list and click the Delete button (–) in the lower-left corner.
6. Click Done.

Removing Provisioning Profiles from Member Center

Occasionally, you may need to remove a provisioning profile from your team.

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. In the dialog that appears, click Delete.

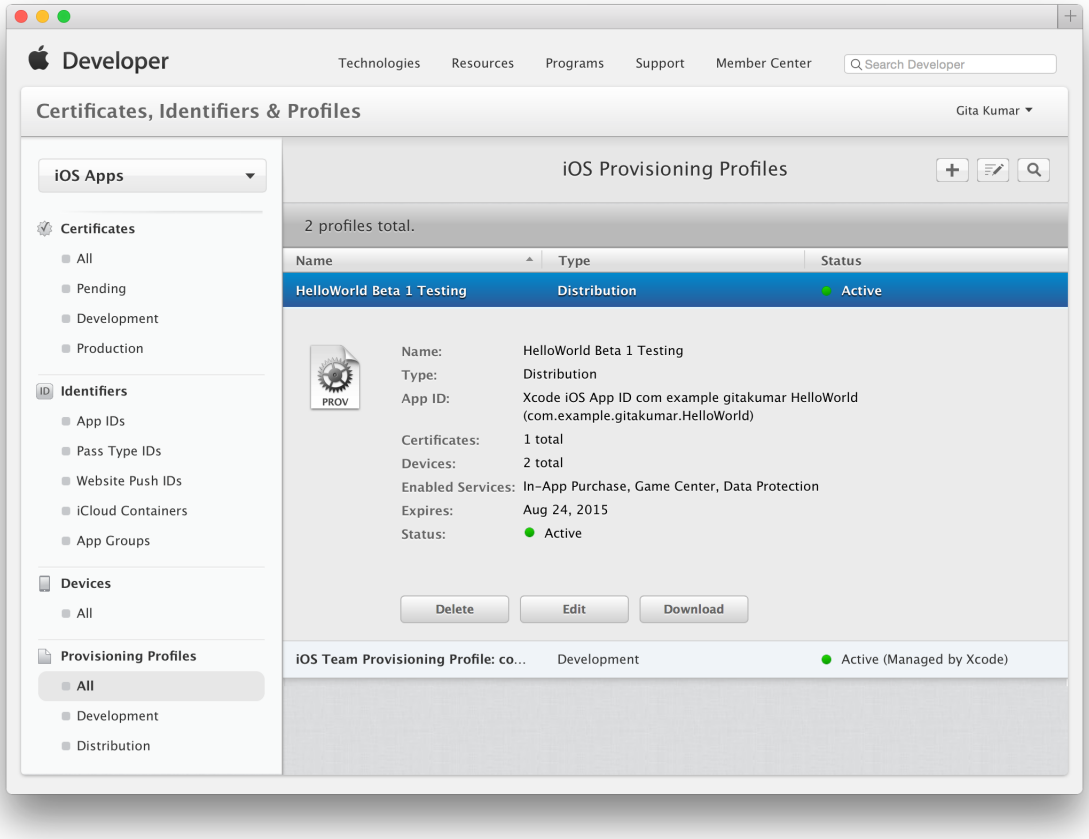
After doing so, refresh provisioning profiles in Xcode, as described in Refreshing Provisioning Profiles in Xcode (page 205), and remove the provisioning profile from your devices, as described in Verifying and Removing Provisioning Profiles on Devices (page 212).

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 provisioning profile.
4. Click Download.



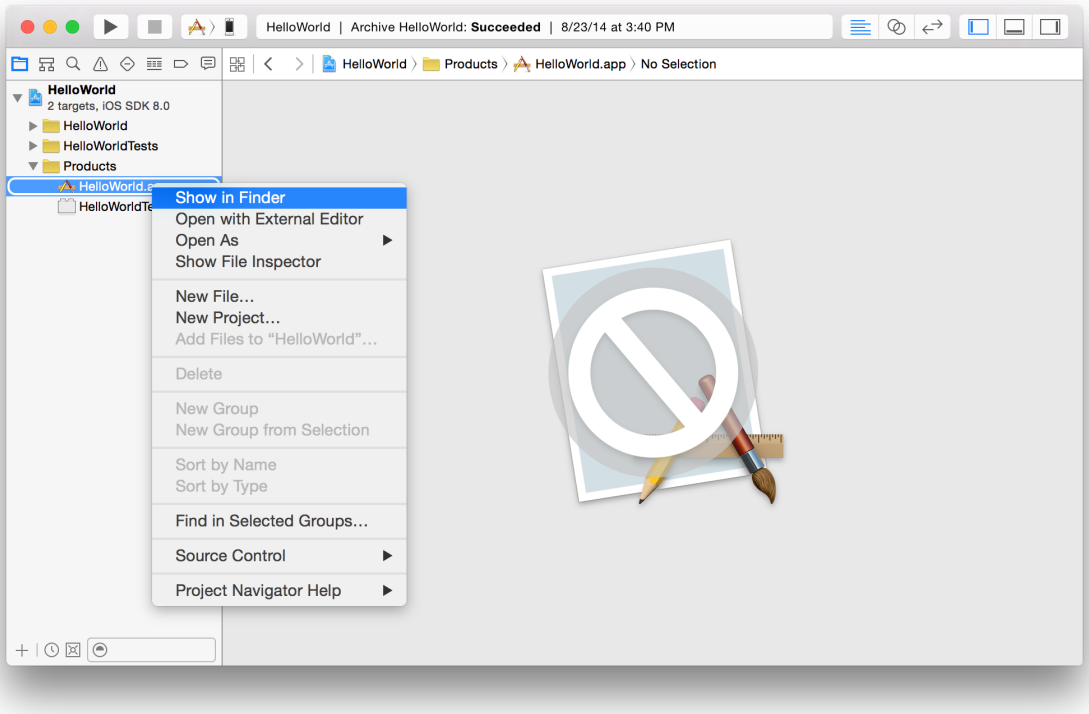
A file with the provisioning profile name and the `.mobileprovision` extension appears in your Downloads folder.

Verifying the App Binary Entitlements

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 and if discrepancies occur, by examining 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 file, and choose "Show in Finder" from the shortcut menu 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 `com.apple.developer.ubiquity-kvstore-identifier` entitlement key. The output for a Mac app that is enabled for App Sandbox contains the `com.apple.security.app-sandbox` entitlement key.

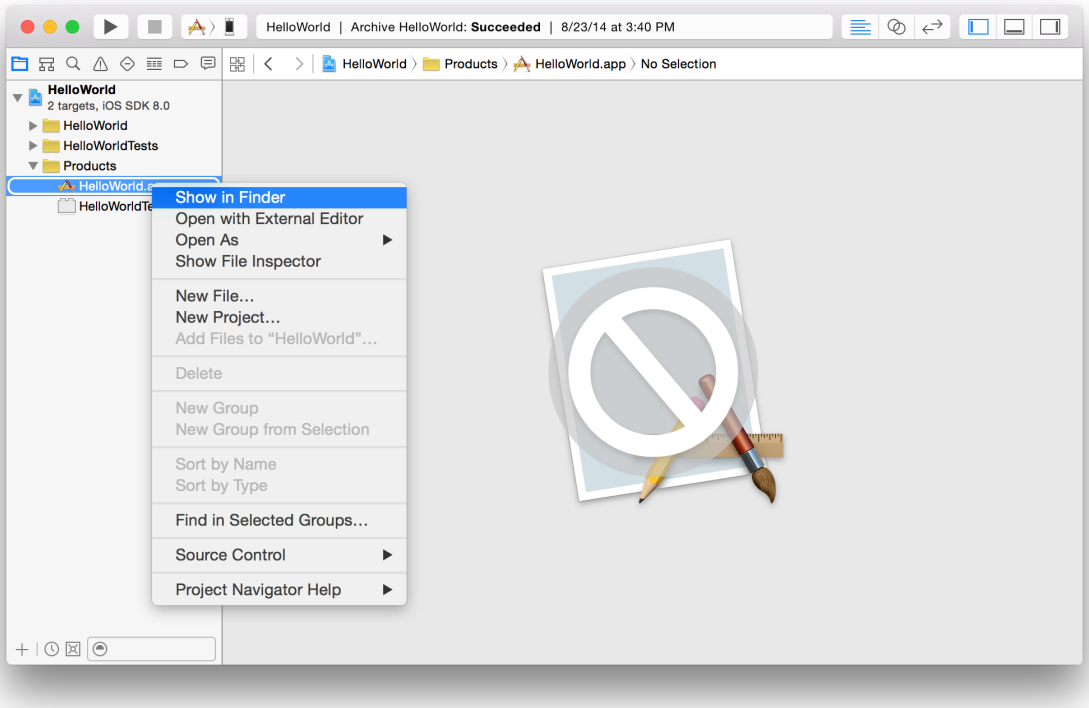
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>application-identifier</key>
    <string>G9B5P7QDV2.edu.self.HelloWorld</string>
    <key>com.apple.developer.pass-type-identifiers</key>
    <array>
        <string>G9B5P7QDV2.*</string>
    </array>
    <key>com.apple.developer.ubiquity-container-identifiers</key>
    <array>
        <string>G9B5P7QDV2.edu.self.HelloWorld</string>
    </array>
    <key>com.apple.developer.ubiquity-kvstore-identifier</key>
    <string>G9B5P7QDV2.edu.self.HelloWorld</string>
    <key>get-task-allow</key>
    <true/>
</dict>
</plist>
```

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

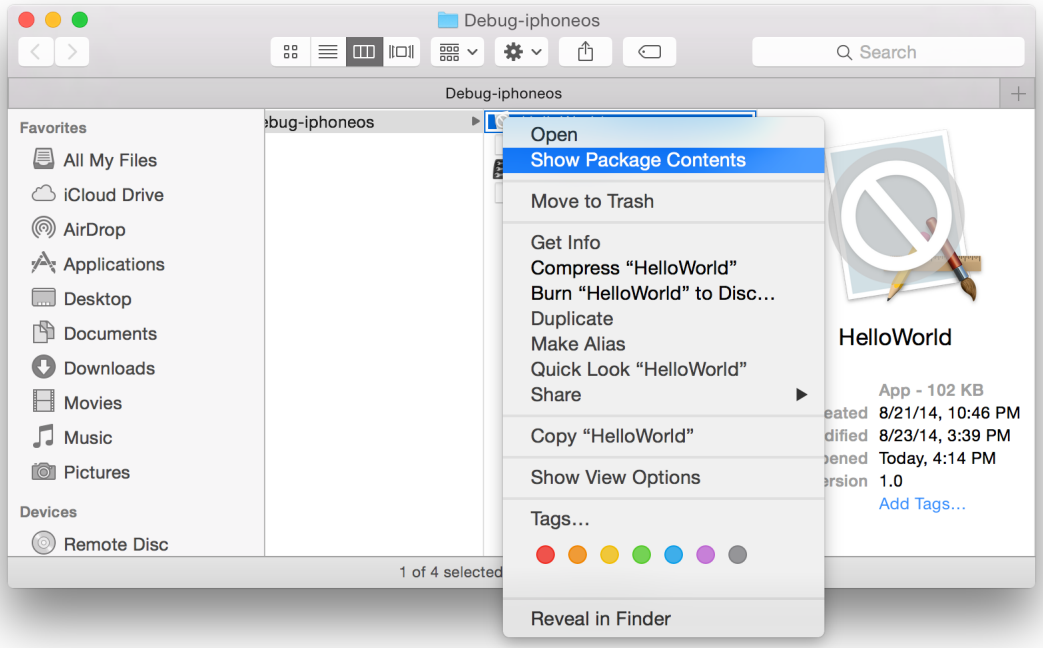
1. In Xcode, select your project in the project navigator.
2. Click the disclosure triangle next to the project to reveal the contents.
3. Click the disclosure triangle next to Products to reveal the binary.

4. Control-click the binary file, and choose “Show in Finder” from the shortcut menu to go to the Xcode build location in the Finder.



5. In the Finder, Control-click the binary file, and choose Show Package Contents from the shortcut 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`.

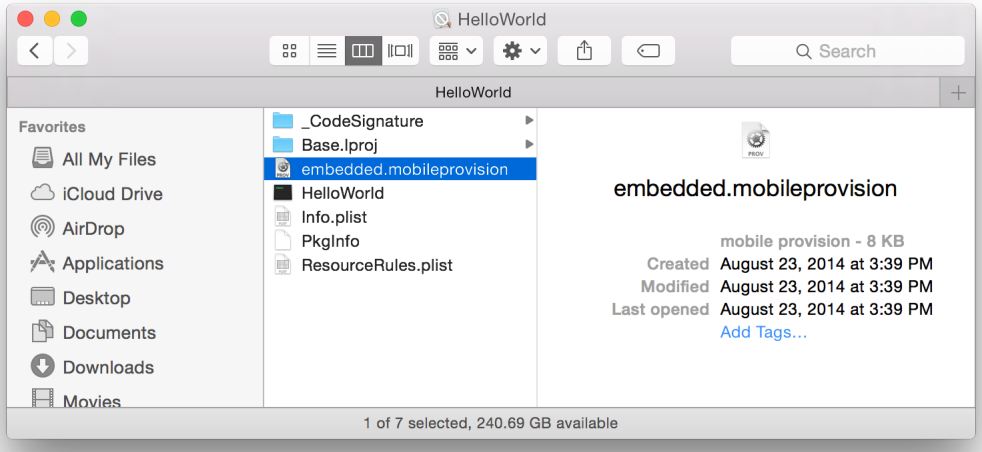


To verify the entitlements of the embedded provisioning profile

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>G9B5P7QDV2.*</string>
        <key>com.apple.developer.default-data-protection</key>
        <string>NSFileProtectionComplete</string>
        <key>com.apple.developer.pass-type-identifiers</key>
        <array>
            <string>G9B5P7QDV2.*</string>
        </array>
        <key>com.apple.developer.ubiquity-container-identifiers</key>
        <array>
            <string>G9B5P7QDV2.*</string>
        </array>
        <key>com.apple.developer.ubiquity-kvstore-identifier</key>
        <string>G9B5P7QDV2.*</string>
        <key>get-task-allow</key>
        <true/>
        <key>inter-app-audio</key>
        <true/>
        <key>keychain-access-groups</key>
        <array>
            <string>G9B5P7QDV2.*</string>
        </array>
    </dict>
```

See `codesign` and `security` for more ways to use these commands.

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 241).

Distributing iOS Developer Enterprise Program Applications

The iOS Developer Enterprise Program allows you to develop proprietary, in-house iOS applications that you can distribute to your employees. You distribute your applications outside the App Store.

If you are not a member of the iOS Developer Enterprise Program, go to iOS Developer Enterprise Program to join.

Developing iOS Developer Enterprise Program Applications

The workflow for developing iOS Developer Enterprise Program applications is similar to the workflow used by any large company that develops multiple applications for the store. During development, let Xcode manage your assets for you and use Member Center only as needed. Xcode will create your App ID and configure your project correctly to use the app services you enable.

Build Your Team (Team Agent)

If you are the team agent (the person who joins the iOS Developer Enterprise Program), build your team first by inviting team members and assigning roles. Assign the team admin role to people who help you manage your team. Assign the team member role to persons who develop your application but don't have permission to distribute it. The team agent and team admins share the responsibility of creating team provisioning profiles, approving development certificates, and registering iOS devices for team members. The tasks a team agent can perform are a superset of the tasks that a team admin can perform. Initially, the team agent is the only team member. To invite others, read Inviting Team Members and Assigning Roles (page 146).

To learn how to manage your team, read Managing Your Team in Member Center (page 144).

> **Important:** You have to be a Registered Apple Developer to join a team. If you invite someone who is not a Registered Apple Developer, that person can register when he or she accepts the invitation.

Create Shared Team Provisioning Profiles (Team Admin)

If you are a team admin, perform these steps to enable your development team:

1. Create a team provisioning profile and enable capabilities.

Follow the same steps as an individual developer to create a team provisioning profile and enable capabilities, described in Configuring Your Xcode Project for Distribution (page 23) and Adding Capabilities (page 48). Xcode automatically creates an appropriate App ID and provisioning profile for you. If you want to use APNs, read Configuring Push Notifications (page 76) for additional steps.

Alternatively, use Member Center to create App IDs and provisioning profiles, described in Maintaining Identifiers, Devices, and Profiles (page 184).

2. Approve team member development certificates.

   Team members must have a development certificate to run an application on an iOS device and use app services. Use Member Center to approve these requests, described in Approving Development Certificates (page 149).

3. Register test devices.

   Team members can't run an application on an iOS device that is not registered. Xcode automatically registers the devices used by the team agent and team admins. However, you must register team member devices as needed, described in Registering Team Member Devices (page 150).

4. Refresh team provisioning profiles as needed.

   After you approve development certificates or register iOS devices, refresh provisioning profiles, described in Refreshing Provisioning Profiles in Xcode (page 205). This adds any missing development certificates and registered devices to team provisioning profiles.

> **Important:** Each team member should have his or her own Apple ID and development certificate. Don't share signing identities between team members.

Team admins can also remove team members as needed. If you use contractors to develop your applications, invite them to join your team and assign them the team member role. When their contract ends, remove them from your team, described in Removing Team Members (page 152).

Begin Development (Team Member)

If you are a team member, perform these steps to get started:

1. When you receive an email invitation, follow the link in the email to join the team.
2. In Xcode, add your Apple ID to the Accounts preferences, described in Adding Your Apple ID Account in Xcode (page 20).
3. If necessary, assign your project to a team, described in Assigning the Xcode Project to a Team (page 30).
4. Request your development certificate, described in Requesting Signing Identities (page 156).
5. If you also want to run an application on an iOS device, locate the device ID, described in Locating Device IDs Using Xcode (page 192), and send it to a team admin.

6. After your development certificate is approved, refresh provisioning profiles to download the certificate and new team provisioning profile, described in Refreshing Provisioning Profiles in Xcode (page 205).

Testing iOS Developer Enterprise Program Applications

You use an ad hoc provisioning profile, described in Distributing Your App Using Ad Hoc Provisioning (page 104), to export an iOS Developer Enterprise Program application from Xcode for beta testing. Only the team agent and team admins can create an ad hoc provisioning profile for distribution. After you export your application, consider using the Xcode service to distribute it to testers and other team members. See *Xcode Server and Continuous Integration Guide* for more information about using the Xcode service.

Important: As a member of the iOS Developer Enterprise Program, you don't have an iTunes Connect account or the ability to distribute your application for beta testing using iTunes Connect.

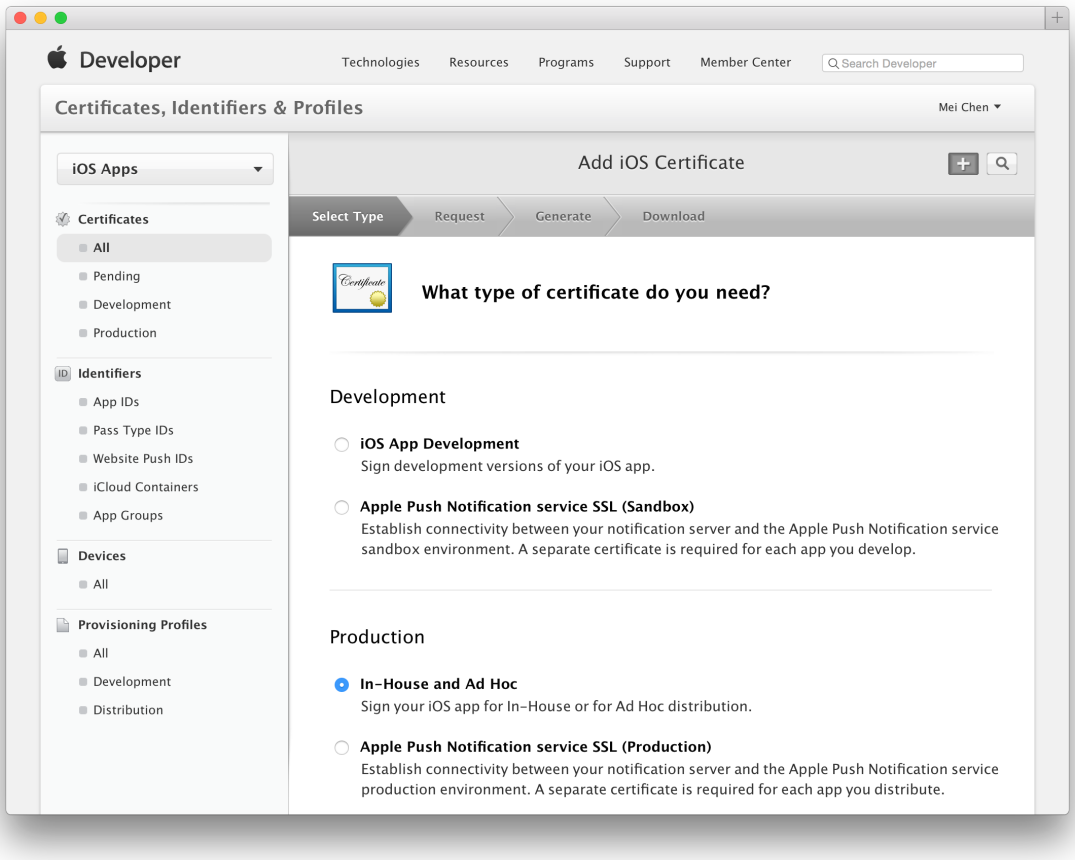
Requesting Additional Enterprise Distribution Certificates

As a member of the iOS Developer Enterprise Program, you are allowed to request multiple distribution certificates.

To request another distribution certificate

1. In Certificates, Identifiers & Profiles, select Certificates.
2. Click the Add button (+) in the upper-right corner.

3. Under Production, select “In-House and Ad Hoc” and click Continue.



4. Follow the instructions to create a certificate signing request (CSR) using Keychain Access, and click Continue.
5. Click Choose File.
6. Select a CSR file (with a `.certSigningRequest` extension), and click Choose.
7. Click Generate.
8. Click Download.

   The certificate file appears in your Downloads folder.

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

Managing Expiring Certificates and Provisioning Profiles

You are responsible for managing your team’s certificates and provisioning profiles. iOS Developer Enterprise Program certificates expire after three years and provisioning profiles expire after one year.

Before a distribution certificate expires, request an additional distribution certificate, described in Requesting Additional Enterprise Distribution Certificates (page 224). You cannot renew an expired certificate. Instead, replace the expired certificate with the new certificate, described in Replacing Expired Certificates (page 178).

If a distribution provisioning profile expires, verify that you have a valid distribution certificate and renew the provisioning profile, described in Renewing Expired Provisioning Profiles (page 212).

Xcode manages your development certificates and team provisioning profiles for you.

Distributing Your Application In-House

To export your application for distribution to your employees and outside the store:

1. Archive your application.
2. Export the archive as an *iOS App* file (a file with a `.ipa` filename extension).
3. Distribute the iOS App file using MDM.

Creating an Archive

Create an archive of your application regardless of the type of distribution method you select. Xcode archives allow you to build your application and store it, along with critical debugging information, in a bundle that's managed by Xcode.

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

To create an archive

1. In the Xcode project editor, choose iOS Device or your device name from the Scheme toolbar menu.

   You can't 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.

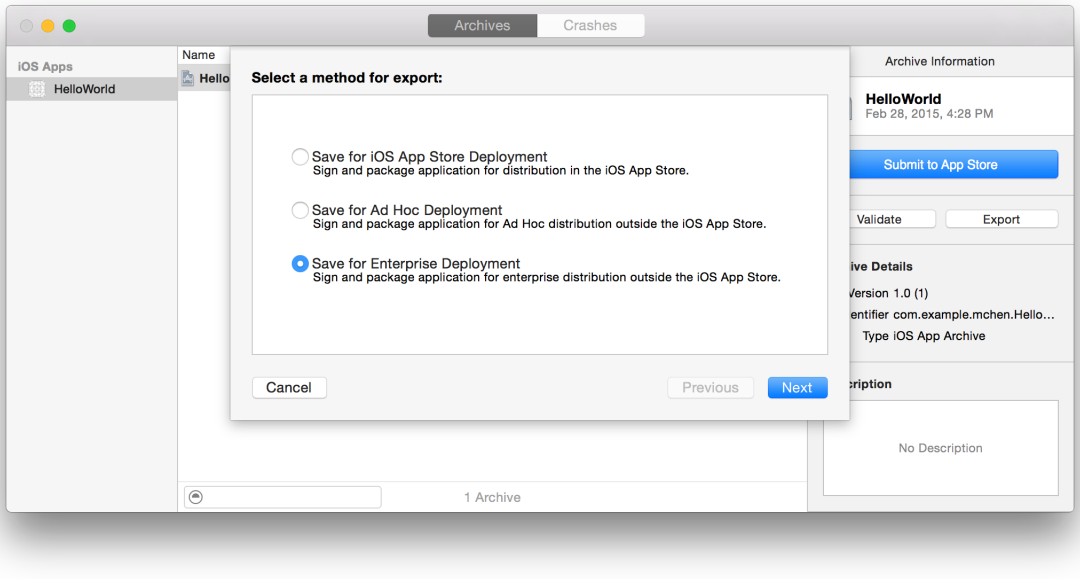
Xcode runs preliminary validation tests on the archive and may display a validate warning in the project editor. For example, if you don't set required application icons, as described in Setting Individual App Icon and Launch Image Files (page 42), an `Info.plist` warning message appears. If you see this warning, fix the issue and create the archive again.

Creating an iOS App File

You create an iOS App file so that users can install your application on their device. You generate an iOS App file (a file with a `.ipa` filename extension) from the archive. Xcode automatically creates the necessary distribution certificates and provisioning profiles for you when you export the application.

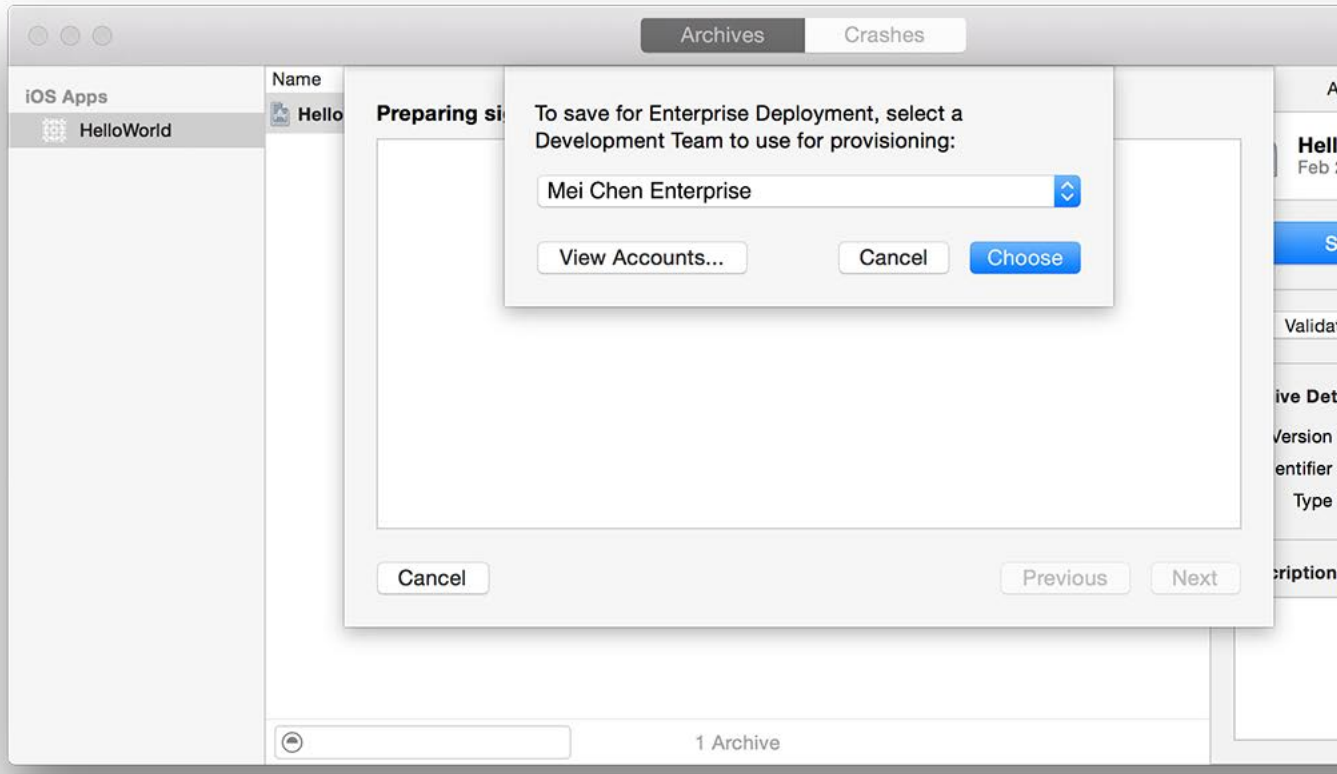
To create an iOS App file for distribution

1. In the Archives organizer, select the archive.
2. Click the Export button.
3. Select "Save for Enterprise Deployment," and click Next.



4. In the dialog that appears, choose a team from the pop-up menu, and click Choose.

If necessary, Xcode creates a distribution certificate, provisioning profile, and explicit App ID for you. (You can not distribute an enterprise app using a wildcard App ID.) The name of the distribution provisioning profile begins with the text `XC:` followed by the App ID. If you are using a wildcard App ID, the name of the distribution provisioning profile is `XC:*`.



5. In the dialog that appears, review the application, its entitlements, and provisioning profile, and click Export.
6. Enter a filename and location for the iOS App file, and click Export.

   The file has an `.ipa` extension.

Learn More About Server Tools

Take advantage of Xcode server tools that support large software development teams.

To learn about	Read
Continuous integration Use Xcode service running on OS X Server to automate building, analyzing, testing, and archiving your application.	*Xcode Server and Continuous Integration Guide*
Automated testing Set up tests that can be run by Xcode service.	*Testing with Xcode*
Host source control repositories on servers Use Xcode service to connect to remote repositories.	*Xcode Server and Continuous Integration Guide* *Source Control Management Help*

Recap

In this chapter, you learned a variation of the development and distribution steps for iOS Developer Enterprise Program members. You learned how to build your team, create shared team assets, export your app for testing, and later, export your app for distribution outside the App Store.

Distributing Applications Outside the Mac App Store

In some cases, you may want to distribute an application outside the Mac App Store. Because that application won't be distributed through the Mac App Store, use a Developer ID certificate to give your users assurance that you're 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 isn't signed with a Developer ID certificate issued by Apple, it won't launch on a Mac that has Gatekeeper enabled. To avoid this situation, sign your applications and installer packages using a Developer ID certificate. Also, 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 steps to create and test Developer ID-signed applications for distribution outside the Mac App Store.

Creating Developer ID-Signed Applications or Installer Packages

Creating a Developer ID-signed application or installer package is a multistep process. First you tell Xcode that you intend to distribute your application outside the Mac App Store and then 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: Before you begin, enroll in the Mac Developer Program, as described in Adding a Developer Program to Your Team (page 21). Only team agents belonging to the Mac Developer Program are eligible to request Developer ID certificates and sign applications or installer packages using them.

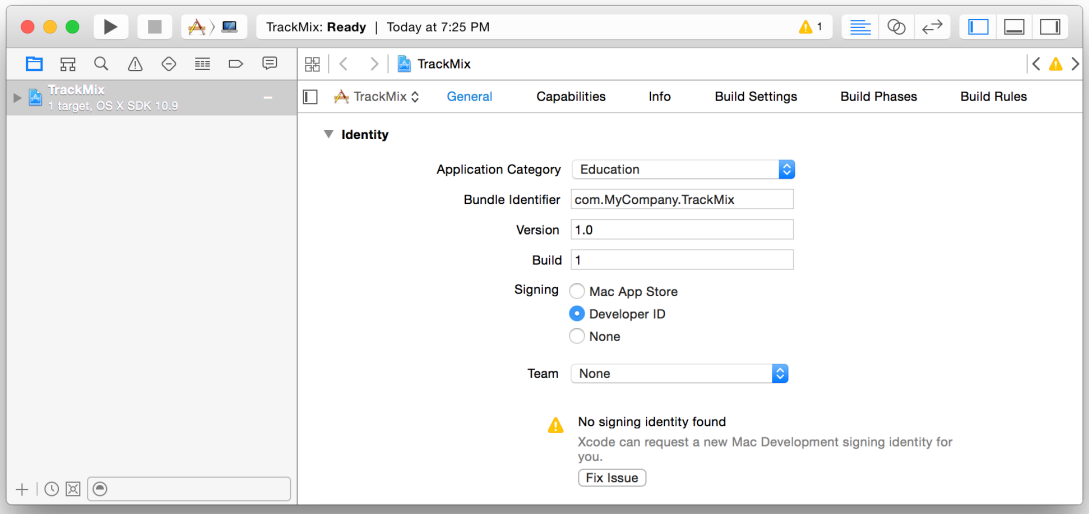
Setting the Signing Identity to Developer ID

First set the signing identity in the General pane to Developer ID.

To set the signing identity to Developer ID

1. In the project navigator, select the target to display the project editor.
2. Click General and, if necessary, click the disclosure triangle next to Identity to reveal the settings.

3. Verify that your bundle ID is unique.
4. Under Signing, select Developer ID as the signing identity.



5. If necessary, choose your team or “Add an Account” from the Team pop-up menu.

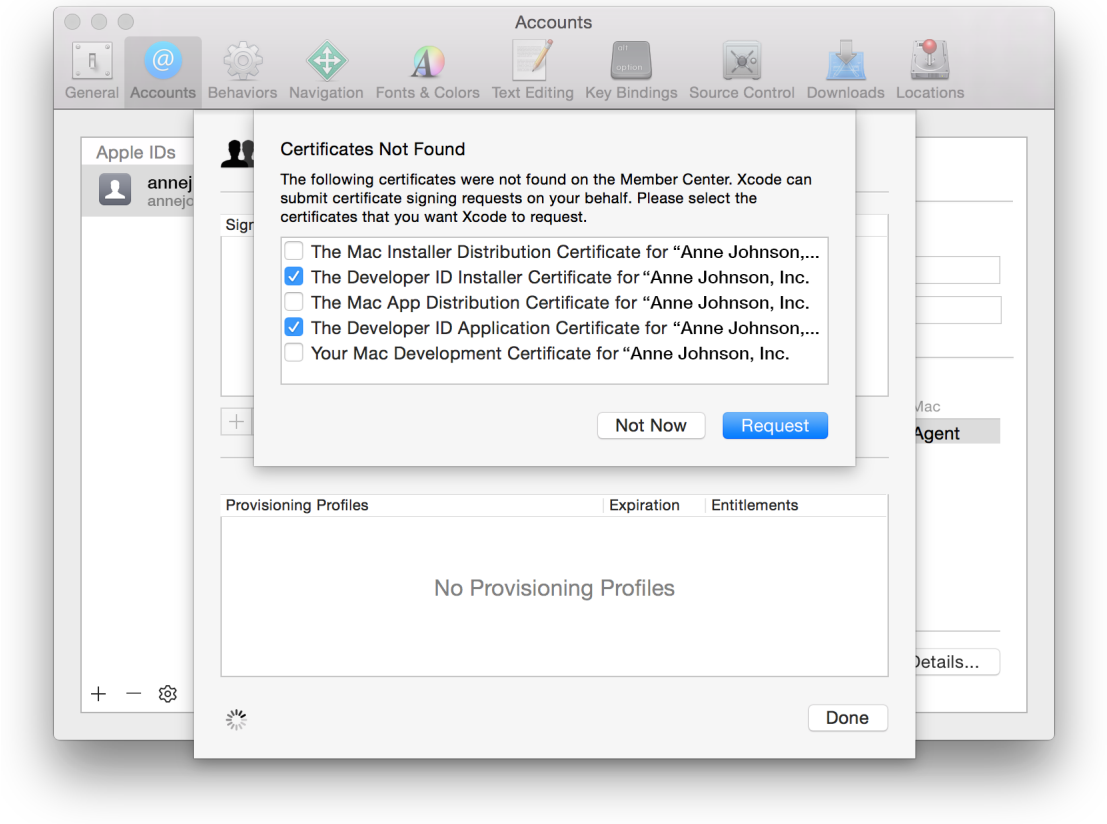
   A warning message and Fix Issue button may appear below the Team pop-up menu. You don’t use a team provisioning profile when signing your app with a Developer ID certificate so can ignore this message.

You can’t use some app services if you distribute outside the Mac App Store. If you enable a capability in the Capabilities pane, as described in Adding Capabilities (page 48), the Signing radio button reverts to Mac App Store. To add your Apple ID to Xcode and add the Mac Developer Program, read Managing Accounts (page 18).

Requesting Developer ID Certificates

You use signing certificates that begin with the text “Developer ID” to distribute your application outside the Mac App Store.

When you refresh provisioning profiles for the first time, as described in Refreshing Provisioning Profiles in Xcode (page 205), Xcode asks whether to request all types of certificates on your behalf. Be sure to select the Developer ID certificates from the Certificates Not Found dialog and click Request. Alternatively, use Accounts preferences to specifically request any missing Developer ID certificates, described in Requesting Signing Identities (page 156).



To use these certificates, you also need the Developer ID Certification Authority intermediate certificate that Xcode installs in your keychain for you to use these certificates. If you’re missing this intermediate certificate, read Installing Missing Intermediate Certificate Authorities (page 164) to restore it.

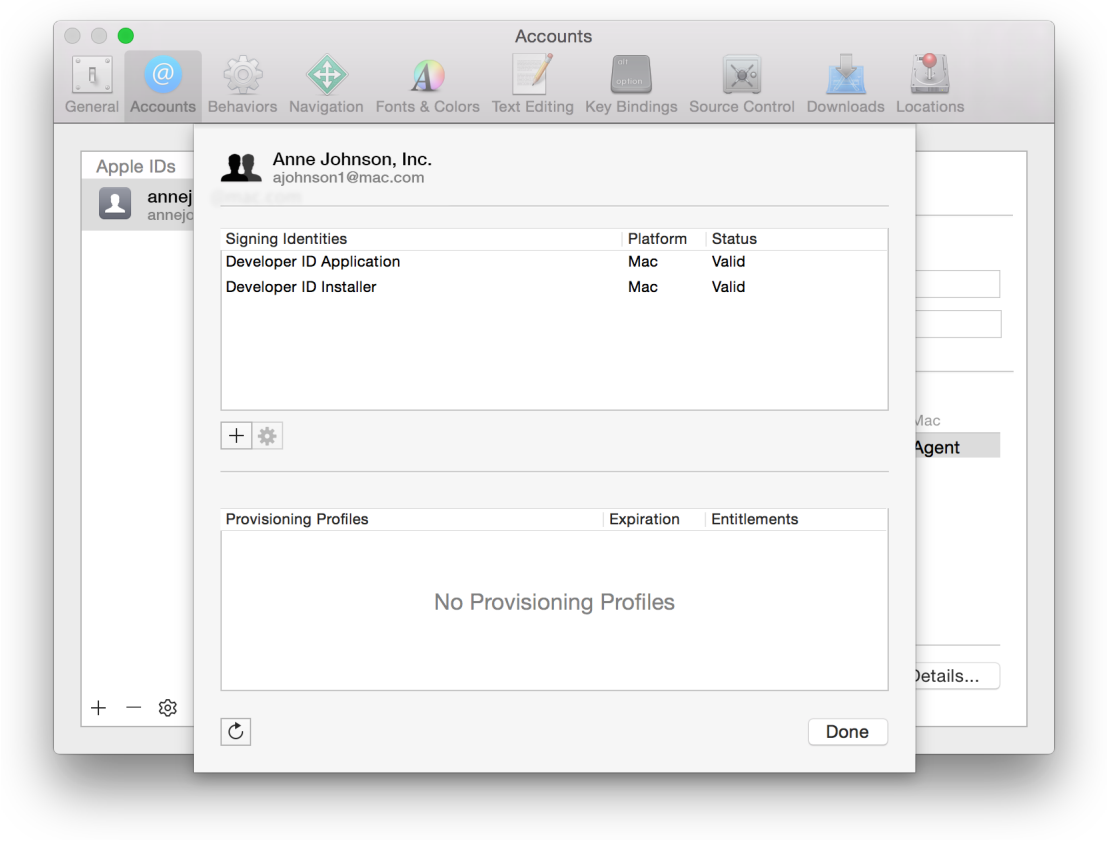
You should immediately back up your Developer ID signing identities after creating them, as described in Exporting Your Developer Profile (page 166).

If you want multiple Developer ID certificates, read Requesting Additional Developer ID Certificates (page 162).

Note: Only a team agent can request Developer ID certificates. If you're an individual developer, you're the team agent and can request these certificates. Contact `product-security@apple.com` if you want to revoke Developer ID certificates.

Verifying Your Steps

To verify your steps, view your Developer ID certificates in Accounts preferences, as described in Viewing Signing Identities and Provisioning Profiles (page 155).



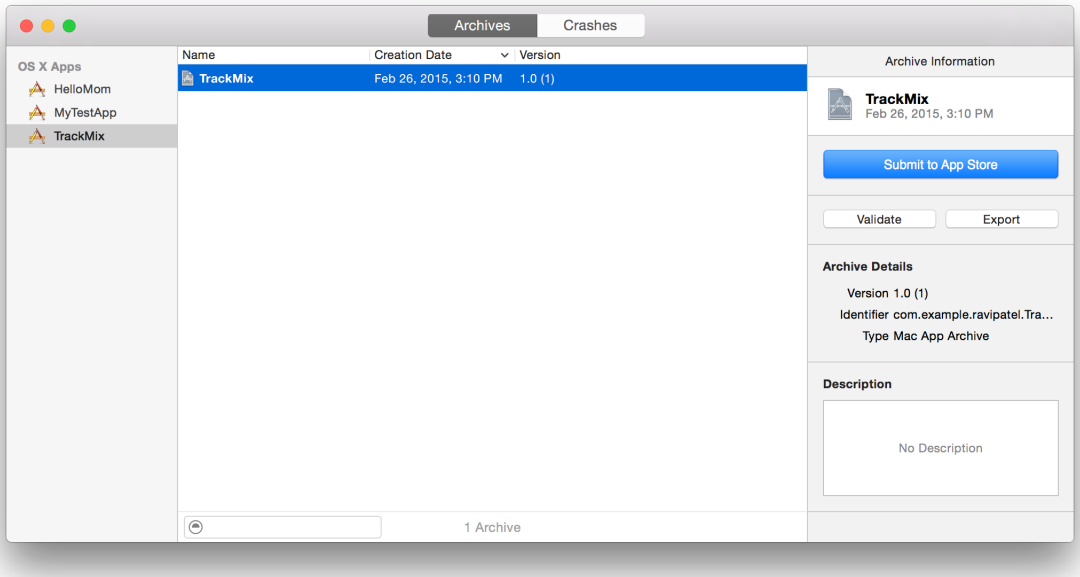
Creating an Archive

Before creating the archive, build and run your app one more time to ensure that it's the version you want to distribute.

To create an archive

1. In the Xcode project editor, select the project.
2. Choose Product > Archive.

The Archives organizer appears and displays the new archive.

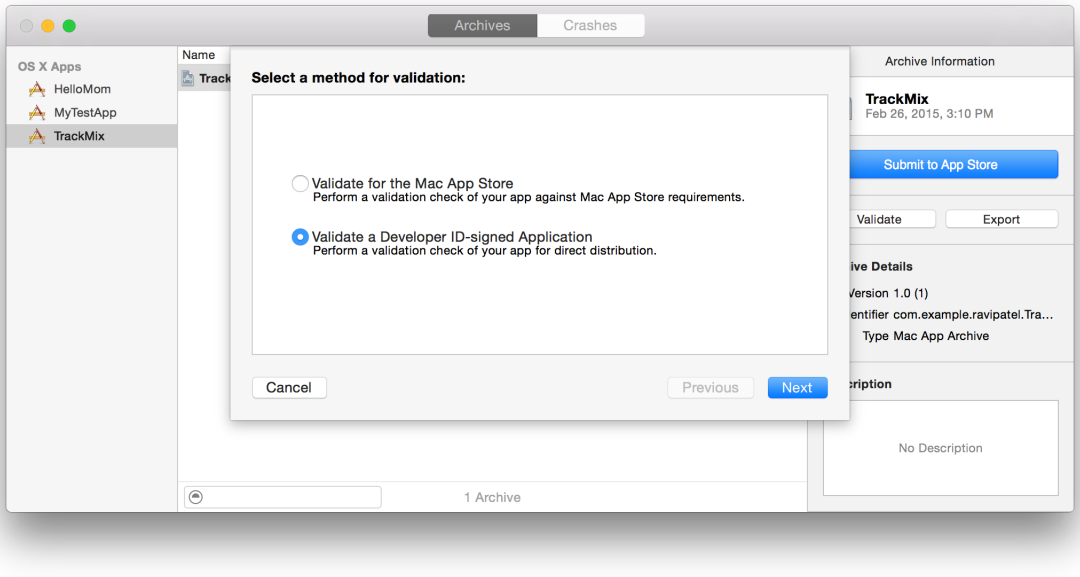


Validating a Developer ID-Signed Application

Immediately after creating the archive, validate it and fix any validation errors before continuing.

To validate a Developer ID-signed archive

1. In the Archives organizer, select the archive and click the Validate button.
2. In the dialog that appears, select “Validate a Developer ID-signed Application” as the validation method and click Next.



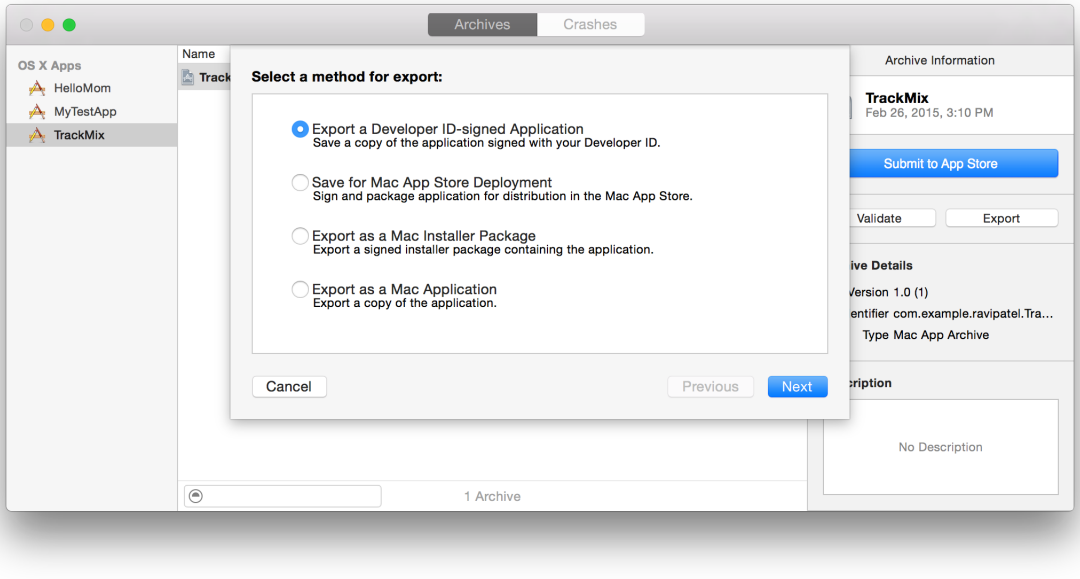
3. In the dialog that appears, choose a team from the pop-up menu and click Choose.
4. Review the signing identity and entitlements, and click Validate.
5. Review validation issues found, if any, and click Done.

Exporting a Developer ID-Signed Application

To export your application for distribution outside the Mac App Store, use the Archives organizer.

To create a Developer ID-signed application

1. In the Archives organizer, select the archive and click Export.
2. In the dialog that appears offering a choice of distribution methods, select "Export a Developer ID-signed Application," and click Next.



3. In the dialog that appears, choose a team from the pop-up menu and click Choose.

   If you're an individual developer, your name appears in the pop-up menu; otherwise, your company name appears in the pop-up menu.
4. In the dialog that appears, review the signing identity and entitlements, and click Export.
5. Enter a filename and location to save the signed application, and click Export.

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` with 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 process includes code signing from the command line, read *Code Signing Guide*.

Verifying 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. Use the `spctl(8)` command-line utility for verifying and testing Gatekeeper too. To simulate the end-user experience, you need to quarantine your application and test it again with Gatekeeper enabled.

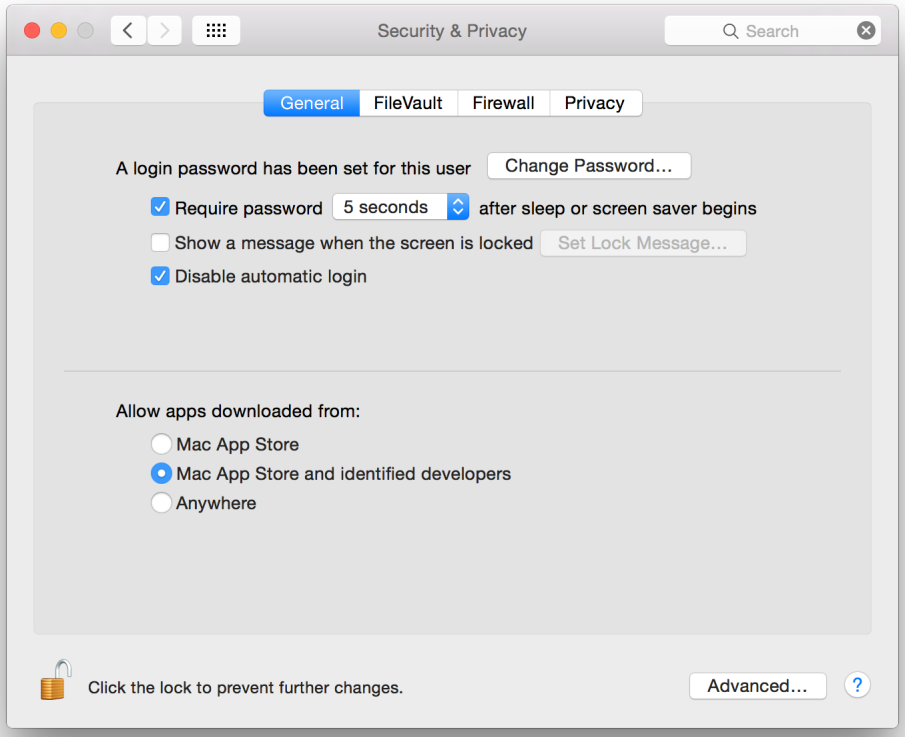
Enabling and Disabling Gatekeeper

You turn on and off Gatekeeper by using the Security & Privacy preferences in System Preferences. You can turn off Gatekeeper and verify the status of Gatekeeper using the `spctl(8)` command-line utility.

To enable or disable Gatekeeper using the Security & Privacy 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 Mac App Store or Anywhere and in the dialog that appears, confirm your selection.

To disable Gatekeeper using the spctl command

1. In Terminal, enter the following command:

```
$ sudo spctl --master-disable
```

2. Press Return.
3. When prompted, enter your administrator password.

To confirm that Gatekeeper is enabled using the spctl command

1. In Terminal, enter the following command:

```
$ spctl --status
```

2. Press Return.

If Gatekeeper is enabled, the output of this command is:

```
assessments enabled
```

If Gatekeeper is disabled, the output of this command is:

```
assessments disabled
```

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's Developer ID signed. (Learn about quarantine in this Knowledge Base article.) You can also test the behavior of Gatekeeper for an application that isn't 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

1. Enable Gatekeeper on your test Mac by selecting "Mac App Store and identified developers" in the Security & Privacy preferences in System Preferences.
2. Enter the following command in Terminal by replacing `TrackMix.app` with the path to your application.

```
$ spctl –a –v TrackMix.app
```

3. Press Return.

   If the application is correctly signed, the output of this command is:

```
./TrackMix.app: accepted
source=Developer ID
```

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

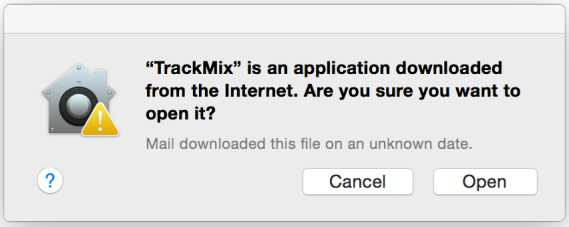
1. Enable Gatekeeper on your test Mac, as described in Enabling and Disabling Gatekeeper (page 236).
2. Quarantine a copy of your Developer ID–signed application. You can do this in either of the following ways:
   - Email your Developer ID–signed application to yourself and use the copy that Mail downloads.
   - Host your Developer ID–signed application on your own local or remote server and use the copy that Safari downloads.

You're ready to test Gatekeeper behavior.

To test Gatekeeper behavior for your Developer ID-signed application

1. 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're 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 application is built correctly.

> **Important:** If an alert doesn't appear at this point, it's likely that you have opened a nonquarantined copy of your application.

To test Gatekeeper behavior for blocking applications that aren't Developer ID signed

1. Enable Gatekeeper on your test Mac, as described in Enabling and Disabling Gatekeeper (page 236).
2. Quarantine a copy of your application that isn't 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 doesn't prevent them from launching.

Recap

In this chapter, you learned how to distribute your Mac application outside 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's 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 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.

You're Missing Signing Identities or Code-Signing Certificate

Xcode detects when you're missing a signing identity, as needed, depending on the operation you're performing. For example, if you attempt to run an app on an iOS device, a dialog may appear asking if Xcode should request a missing development certificate. If this happens, click Fix Issue or Request in the dialog that appears.

If the certificate already exists in Member Center, a "Your account already has a valid certificate" dialog appears. Typically, this happens when you move from one Mac to another. If possible, export your certificates as a developer profile file on the other Mac, and then import them on your new Mac, as described in Exporting and Importing Certificates and Profiles (page 166). If you don't have a backup of your developer profile, click the "Revoke and Request" button when the "Your account already has a valid certificate" dialog appears.

You can also request specific types of certificates, as described in Requesting Signing Identities (page 156).

No Matching Signing Identity or Provisioning Profiles Found

If a warning message and Fix Issue button appear below the Team pop-up menu in the General pane or in a section of the Capabilities pane in the project editor, read the message and click the Fix Issue button. Xcode may present a series of dialogs asking for more information. For example, a dialog might ask you to select a team for your project, request your development certificate, or register a device. Xcode needs these assets to create your team provisioning profile. If Xcode doesn't resolve the issue, first follow all the steps in Configuring Identity and Team Settings (page 26) to verify your project configuration.

For iOS apps, connect a device you want to use for development to your Mac and choose it from the Scheme toolbar menu before clicking Fix Issue. (Xcode automatically registers connected devices selected from this menu.) If the device appears in the ineligible list in the menu, read Your iOS Device Isn't Listed or Is Ineligible in the Scheme Toolbar Menu (page 247).

Read The Private Key for Your Signing Identity Is Missing (page 242) if you're missing a private key in your keychain.

The Private Key for Your Signing Identity Is Missing

Code signing fails if the private key for your signing identity isn't in your keychain.

If the private key for your development certificate is missing, Xcode displays a warning message and several options to fix the issue below the Team pop-up menu in the General pane in the project editor. If you created a backup of your signing identities on this or another Mac, as described in Exporting Your Developer Profile (page 166), click the Import Developer Profile button to restore the private key in your keychain. Otherwise, click the "Revoke and Request" button to create a new signing identity. Remove the signing identity with the missing private key from your keychain, as described in Removing Signing Identities from Your Keychain (page 172).

If you use a custom development provisioning profile that you manage yourself, it becomes invalid after revoking the development certificate. Read Editing Provisioning Profiles in Member Center (page 209) to regenerate it.

If the private key for a distribution certificate is missing, try to restore it from a developer profile backup, as described in Importing Your Developer Profile (page 171). If you can't retrieve your private keys from another Mac, refer to Re-Creating Certificates and Updating Related Provisioning Profiles (page 178) to re-create these types of certificates. You can perform these steps for one or more certificates.

The Private Key for a Developer ID Certificate Is Missing

Follow the same steps in The Private Key for Your Signing Identity Is Missing (page 242) to repair Developer ID certificates, except contact Apple at `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 162).

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, as described in Verifying Using Keychain Access (page 160), 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're missing the intermediate certificate. If you don't have

a certificate called *Apple Worldwide Developer Relations Certification Authority* in your system keychain, read Installing Missing Intermediate Certificate Authorities (page 164) 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 245).

Your Provisioning Profile or Signing Identity Doesn't Appear in Xcode Menus

Occasionally, your provisioning profile or signing identity doesn't appear in a Provisioning Profile or a pop-up menu when distributing your app using the Devices window or when setting the Code Signing Identity build setting in the project editor. When this occurs, refresh provisioning profiles in Xcode, as described in Refreshing Provisioning Profiles in Xcode (page 205).

If your provisioning profile still doesn't appear in the Code Signing Identity build setting menu, choose Don't Code Sign or choose a certificate under Automatic Profile Selector from the Code Signing Identity pop-up menu. The next time you use the Code Signing Identity pop-up menu, your provisioning profile should appear in the menu.

Duplicate Signing Identities or Provisioning Profiles Appear in Accounts Preferences

The view details dialog in Accounts preferences displays signing identities that are in your keychain, not certificates in Member Center. You can have multiple accounts and belong to different teams, but you should have only one of each type of certificate for each team. For a list of the certificate types, refer to Table 13-2 (page 182). If duplicate signing identities appear, you can remove the expired or revoked signing identities from your keychain, as described in Removing Signing Identities from Your Keychain (page 172).

If duplicate provisioning profiles appear in the view details dialog in Accounts preferences, refresh provisioning profiles in Xcode, as described in Refreshing Provisioning Profiles in Xcode (page 205).

Your Certificates Have Expired

You can't renew expired certificates. Read Replacing Expired Certificates (page 178) 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'll need new Developer ID certificates to sign updates and create new applications.

Your Request Has Timed Out

If a dialog appears stating that your request has timed out, the Certificates, Identifiers & Profiles section of Member Center is probably down for maintenance. To see whether you have access to Certificates, Identifiers & Profiles, sign in to Member Center using the same Apple ID credentials you entered in Xcode. In Certificates, Identifiers & Profiles, select Certificates or Provisioning Profiles. If you can view your assets, Certificates, Identifiers & Profiles is not down.

Provisioning Issues

Common provisioning issues result from using the wrong provisioning profile with your app or using an invalid or expired provisioning profile.

Provisioning Profiles Installed on Your Device Are Invalid

If the provisioning profile on your development device has expired or is invalid, remove it as described in Verifying and Removing Provisioning Profiles on Devices (page 212).

Provisioning Profiles Appear Invalid in Member Center

If a team provisioning profile appears invalid in Member Center, refresh provisioning profiles in Xcode, as described in Refreshing Provisioning Profiles in Xcode (page 205). For example, if you enable push notifications for an App ID using Member Center, use Xcode to regenerate the associated team provisioning profiles.

If a provisioning profile you manage appears invalid in Member Center, remove or edit it, as described in Removing Provisioning Profiles from Member Center (page 214) and Editing Provisioning Profiles in Member Center (page 209). A provisioning profile is invalid if it contains a revoked or expired certificate. A provisioning profile is also invalid if its App ID entitlements change or its App ID is deleted. A development or ad hoc provisioning profile is invalid if it contains a disabled device.

No Such Provisioning Profile Was Found

If a "Your build settings specify a provisioning profile with […] however, no such provisioning profile was found." warning message appears below the Team pop-up menu in the General pane, you may have incorrect Provisioning Profile and Code Signing Identity build settings from using an earlier version of Xcode.

When this occurs, click the Fix Issue button below the warning message. If necessary, click the Fix Issue button again. The Provisioning Profile build setting should be None and the Code Signing Identity build setting should be Automatic and iOS Developer for iOS apps, and Automatic and Mac Developer for Mac apps.

If you're using a custom development provisioning profile, read Using Custom Provisioning Profiles (page 206) for how to configure your project.

Build and Code Signing Issues

If you use the team provisioning profile that Xcode manages for you during development, as described in Team Provisioning Profiles in Depth (page 95), Xcode fixes code signing and provisioning issues for you before you attempt to build your app. In this case, you shouldn't set the Code Signing Identity build settings yourself. However, if you want to use a custom development provisioning profile and set these build settings, as described in Using Custom Provisioning Profiles (page 206), you may encounter build issues described in this section. Common build errors tend to involve incorrect code signing identities.

Xcode Can't Find Your Provisioning Profile

You'll 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 'xxxxxxxxx-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.

Xcode Doesn't Trust Your Certificate

You get this error message when Xcode can't verify the authenticity of your development or distribution certificate:

```
Code Sign error: CSSMERR_TP_NOT_TRUSTED
```

If the trust setting isn't Use System Defaults, you'll receive a `CSSMERR_TP_TRUSTED` error message from the `codesign` command-line utility when you build and run your app. Don't 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'll 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, set this build setting at the project level, not the target. However, if it's 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 pop-up menu, try refreshing the provisioning profiles, as described in Refreshing Provisioning Profiles in Xcode (page 205). If you are using an Xcode-managed provisioning profile, read Using Xcode-Managed Provisioning Profiles (page 209). If you are using a custom provisioning profile, read Using Custom Provisioning Profiles (page 206).

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)"
```

```
[BEROR]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, try deleting the duplicate code signing identities from your keychain, as described in Removing Signing Identities from Your Keychain (page 172).

The App ID in 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.

Your iOS Device Isn't Listed or Is Ineligible in the Scheme Toolbar Menu

If you have a project or workspace open and your connected iOS device isn't listed as a 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 Devices (iOS Only) (page 37) for details.

2. 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 hasn't 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.

Archiving and Submitting Issue

If you select an archive in the Archives window and the Validate and Submit button are disabled, verify that the archive contains a single top-level app.

Document Revision History

This table describes the changes to *App Distribution Guide*.

Date	Notes
2015-04-08	Updated per Xcode 6.3. Updated Archives organizer workflows and added Crashes organizer steps to the "Analyzing Crash Reports" chapter.
2014-12-18	Applied minor edits throughout.
2014-10-20	Applied minor edits.
2014-10-16	Updated per Xcode 6.1 and iTunes Connect changes.
2014-09-17	Updated for Xcode 6.
2014-02-11	Applied minor edits throughout.
2013-12-12	Added "Deleting App IDs" in "Maintaining Identifiers, Devices, and Profiles" and updated iTunes Connect documentation links.
2013-10-22	Added steps to manage asset catalogs, revoke certificates in Xcode, install iOS developer previews, renew expired provisioning profiles, verify app entitlements, and troubleshoot other issues.
2013-09-18	Updated per Xcode 5.
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's 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.*`, it matches the bundle ID `com.mycompany.MyApp`.

certificate authority An organization that authorizes a certificate.

Certificates, Identifiers & Profiles An area of Member Center available to iOS, Mac, and Safari Developer Program members that provides resources needed to develop iOS and Mac apps and Safari Extensions.

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.

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.

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 services 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 services. 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 aren't 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 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's required to be in a developer's keychain to ensure that a signing certificate is issued by a trusted source.

iOS App file 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 apps.

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 isn't 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.

signing identity A digital identity used for code signing, including archive signing. A signing identity includes the certificate with its private and public keys stored in the keychain.

store Used as a short form of the App Store or the Mac App Store when there's 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's 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 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.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010