

App Distribution Quick Start

Contents

About This Document 4

[How to Use This Document](#) 4

[See Also](#) 5

Setup 6

[Install the Latest Version of Xcode](#) 6

[Create an Xcode Project](#) 6

[Enroll in a Developer Program](#) 9

Adding Your Account to Xcode 10

[About Accounts and Teams](#) 10

[Add Your Apple ID To Xcode](#) 10

[Verify Your Account](#) 11

[Verify Your Xcode Account Credentials](#) 11

[Verify Your Member Center Credentials](#) 15

[Verify Your iTunes Connect Credentials](#) 17

[Recap](#) 18

Creating Your Team Provisioning Profile 19

[About Team Provisioning Profiles](#) 19

[Set the Bundle ID](#) 21

[Assign Your App to a Team](#) 21

[Create the Team Provisioning Profile](#) 25

[Export Your Signing Identities](#) 28

[Recap](#) 32

Launching Your App on Devices 33

[About Devices](#) 33

[Launch Your App](#) 33

[Launch Your iOS App on a Device](#) 34

[Launch Your Mac App](#) 36

[Verify Your Profiles](#) 36

[Verify Signing Identities and Profiles Using Xcode](#) 36

[Verify Your Certificates, Identifiers, and Profiles Using Member Center](#) 39

[Verify Signing Identities Using Keychain Access](#) 41
[Recap](#) 43

Enabling App Services 44

[View App Services](#) 44
[For Mac Apps, Enable App Sandbox](#) 46
[Enable App Services](#) 48
[Verify App ID Entitlements](#) 50
 [View Provisioning Profiles in Xcode](#) 50
 [View Provisioning Profiles in Member Center](#) 53
 [Verify the App ID Settings](#) 54
[Learn More About App Services](#) 55
[Recap](#) 57

Next Steps 58

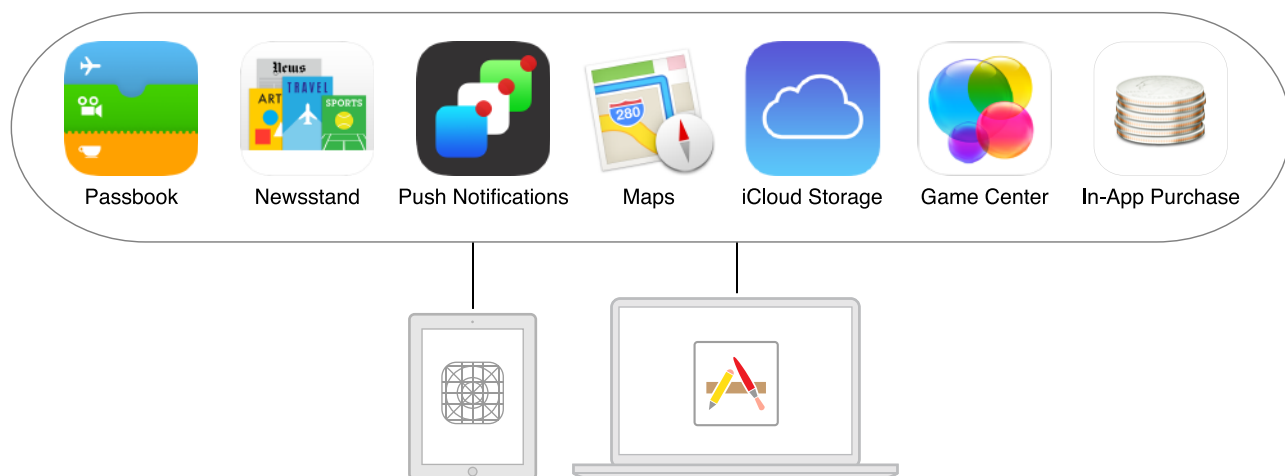
[Where to Go from Here](#) 58

Document Revision History 60

Glossary 61

About This Document

This quick start gets you started code signing and provisioning an app that you eventually submit to the App Store or Mac App Store. Even when you launch your app through Xcode for early testing, your app must be provisioned and code signed to run on an iOS device (an iPad, iPhone, or iPod touch) and to use certain app services such as iCloud storage, Game Center, and In-App Purchase.



This document teaches you a common Xcode workflow you perform while developing your app. You'll learn how to add your account to Xcode, create signing identities and provisioning profiles in Xcode, verify access to Member Center, launch your app on devices, and optionally, enable app services that require provisioning. You'll also learn best practices, such as backing up your signing identities.

This quick start is a companion document for *App Distribution Guide*, which covers all phases of development and alternative workflows, such as distributing your Mac app outside the Mac App Store.

How to Use This Document

This document applies to both iOS and Mac apps unless otherwise indicated. It is assumed that you're the person who enrolls in the iOS Developer Program or Mac Developer Program and who has permission to create code signing and provisioning assets in Member Center.

Important: If you're a team member for a company, note that some steps need to be performed by the person—a team agent or admin—who manages your team. Specifically, the team agent or admin needs to approve your development certificate request and register your device for you, as described in *Managing Your Team* in *App Distribution Guide*.

Read the “About...” section at the beginning of each chapter to learn more about the assets you'll create and understand the terms used in Xcode dialogs and messages. Read the “Verify...” sections to learn where your assets reside and how to inspect them, which is the first step in troubleshooting.

A glossary defines key terms used in this document.

See Also

If you're new to iOS or OS X development, read *Start Developing iOS Apps Today* or *Start Developing Mac Apps Today* for an introduction to Xcode and how to write code. For comprehensive coverage of provisioning, distribution, and troubleshooting, read *App Distribution Guide*. Also see [Where to Go from Here](#) (page 58) in this document.

Setup

This quick start requires:

- A Mac computer with Xcode 6 or later installed
- For the best experience, the latest OS X and Xcode releases installed
- An Xcode project that builds without errors
- Membership in either the iOS Developer Program or the Mac Developer Program

Install the Latest Version of Xcode

Xcode, Apple's integrated development environment (IDE), is the primary tool for developing your app. It includes a source editor, a graphical user interface editor, and many other features. Xcode simplifies the provisioning and code signing process so that you don't need to leave Xcode to develop your app. Later in the distribution process, you'll use Xcode and other tools to distribute your app for beta testing and to the store.

To install the latest version of Xcode, go to the [Mac App Store](#).

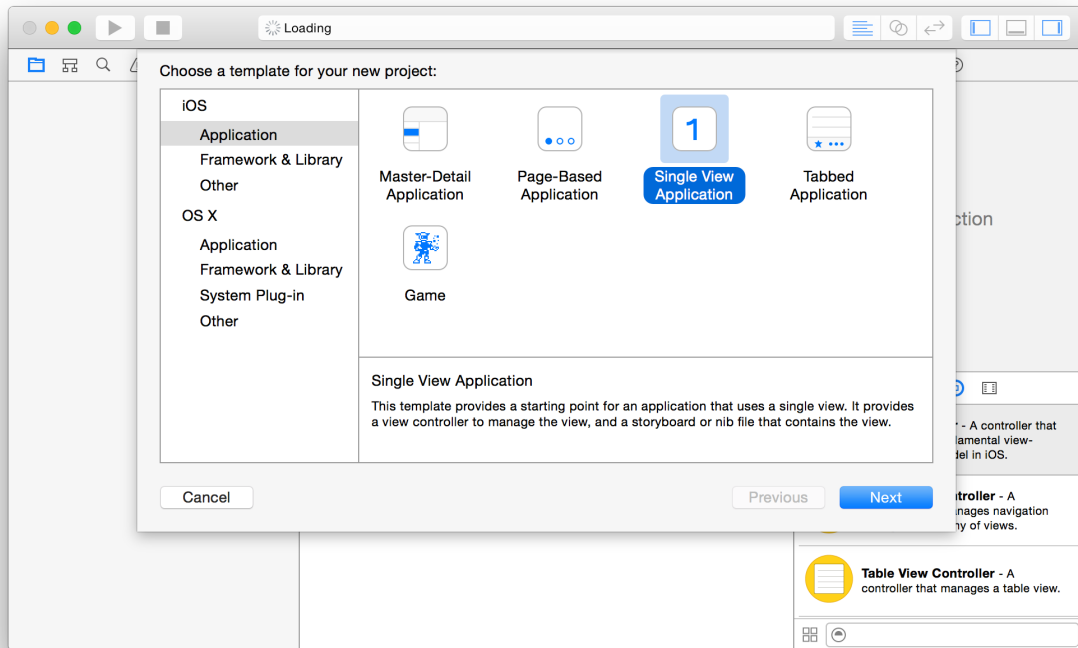
Create an Xcode Project

If you don't have an Xcode project that builds and runs without errors, you can create a simple app now just to learn the code signing and provisioning steps in this document.

To create an Xcode project

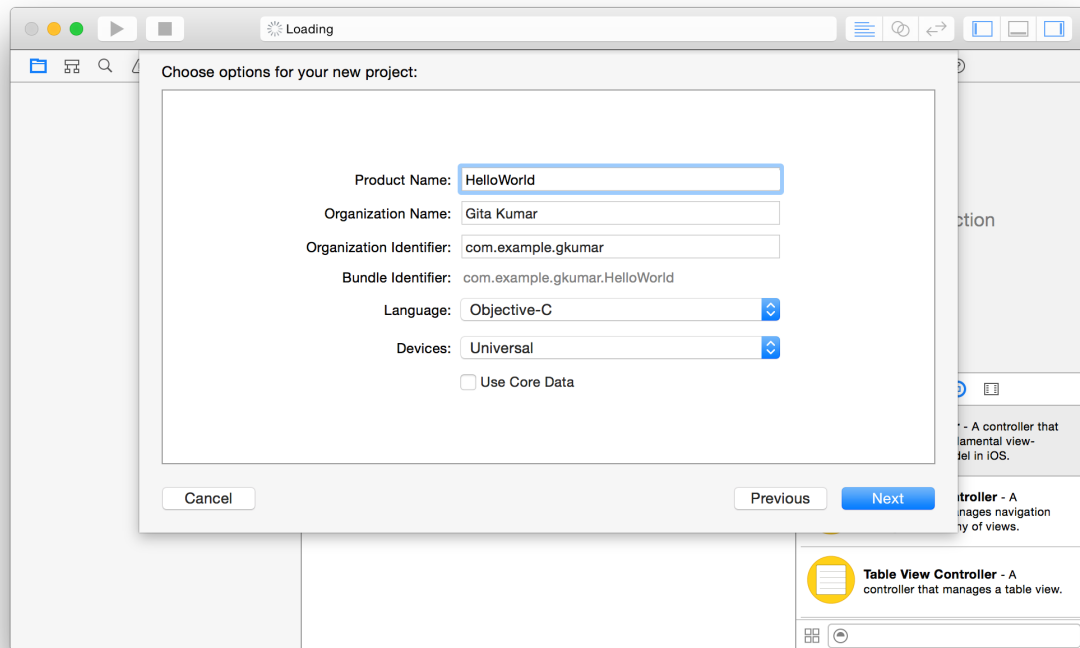
1. Open Xcode.
2. Choose File > New > Project, or click "Create a new Xcode project" in the "Welcome to Xcode" window.
3. In the iOS or OS X section, select Application, select a template from the list of templates, and click Next.

For example, to create an app with a single empty window, for iOS apps select Single View Application, and for Mac apps select Cocoa Application.



4. In the dialog that appears, fill in the Product Name and Company Identifier text fields.

The company identifier should be in reverse DNS format. If you don't have a company identifier, use `com.example.` followed by your name, and replace it later. The other default values in this dialog should suffice for now. The following screenshot shows options for creating an iOS app. The Mac app options are similar, but not identical.

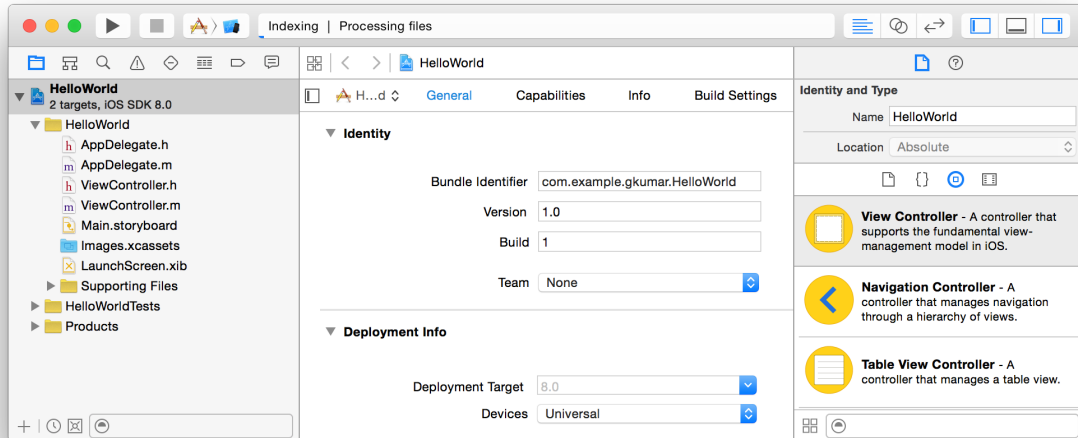


5. From the Language pop-up menu, choose a programming language .
6. Click Next.

A dialog asks you where to save your project.

7. Specify a location for your project, optionally deselect "Create git repository on," and click Create.

For iOS apps, a new project window appears, similar to the one below.



Enroll in a Developer Program

If you plan to submit your app to the store, you first need to join the iOS Developer Program or Mac Developer Program. After you join the iOS Developer Program, you can start running your iOS app on devices. For Mac apps, you can start using app services.

To enroll in either of these programs, go to the [Apple Developer Program Enrollment](#) website and follow the instructions.

Adding Your Account to Xcode

When you add your Apple ID to the Xcode Accounts preferences, Xcode displays all the teams and developer programs you belong to. Xcode also shows your role on the team and details about your signing identities and provisioning profiles that you'll create later in this document.

About Accounts and Teams

Apple Developer Programs provide everything you need to distribute your iOS or Mac app. You join the **iOS Developer Program** to submit your iOS app to the **App Store**. You join the **Mac Developer Program** to either submit your app to the **Mac App Store** or distribute it outside the Mac App Store. After you join a developer program, you can use platform-specific app services that are available only to apps submitted to the store. You also have access to more tools—**Member Center** and **iTunes Connect**—where you manage metadata about your organization and your app.

You use your **Apple ID**, which uniquely identifies you, to enroll in a developer program. You create and enroll a team for which you're the principal developer. You can enroll as an individual or a company where an individual is considered a one-person team. The person who creates the team becomes the **team agent**, the legal contact and administrator of the team who has all privileges and full access to Member Center and iTunes Connect.

Using Member Center, a team agent for a company can add people to the team and assign roles that define their individual privileges. A **team admin** can perform nearly all the same tasks as a team agent—for example, a team admin can manage the team but not sign legal agreements. A **team member** can only run apps on devices and use certain app services if approved by a team agent or admin.

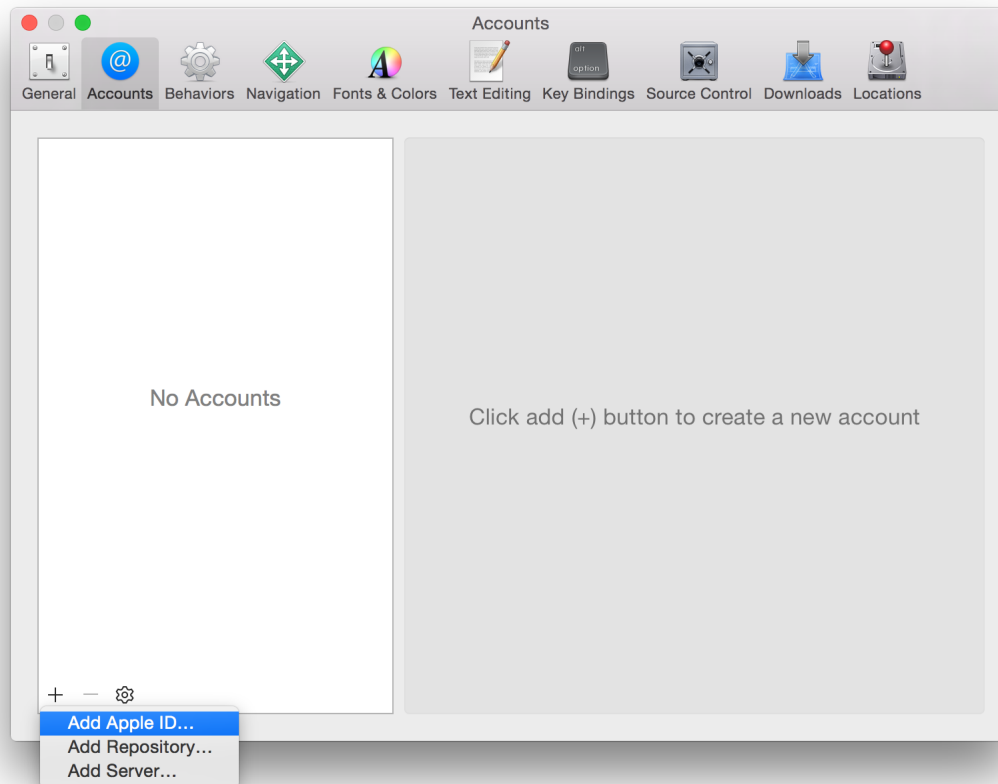
Add Your Apple ID To Xcode

After you join a developer program, add your Apple ID to Accounts preferences in Xcode.

To add an Apple ID to Xcode

1. Choose Xcode > Preferences.
2. At the top of the window, click Accounts.
3. In the lower-left corner, click the Add button (+) .

4. Choose Add Apple ID from the pop-up menu.



5. In the dialog that appears, enter your Apple ID and password, and click Add.

Verify Your Account

Xcode uses your Apple ID credentials to sign in to Member Center and iTunes Connect on your behalf. Occasionally, you'll need to sign in to Member Center and iTunes Connect to perform tasks yourself. Before continuing, verify that Xcode and you have access to these tools.

Verify Your Xcode Account Credentials

In Xcode, verify that your Apple ID appears in Accounts preferences and viewing the details of your account shows your signing identities and provisioning profiles.

To view account details in Xcode

1. If necessary, choose Xcode > Preferences, and at the top of the window, click Accounts.

The teams that you belong to appear in a table to the right of your selected Apple ID. The team name appears in the Name column, and your role appears under the iOS or Mac column depending on which developer program the team belongs to. If your role is Agent or Admin, you can perform all the steps in this document. If your role is Member, a team agent or admin will perform some of the steps on your behalf.

A Join button appears under the iOS or Mac column if you don't belong to that developer program. For iOS apps, you must belong to the iOS Developer Program, and for Mac apps, you must belong to the Mac Developer Program. To add a developer program to your team, click the Join button. Your default browser will display the developer program enrollment webpage. Click "Enroll now" and follow the instructions.

2. Select the team you want to view in the table, and click View Details.

In the dialog that appears, view your signing identities and provisioning profiles. Xcode shows the signing identities that are in your keychain. If this is the first time you're code signing an app, no signing identities or provisioning profiles appear in these tables.

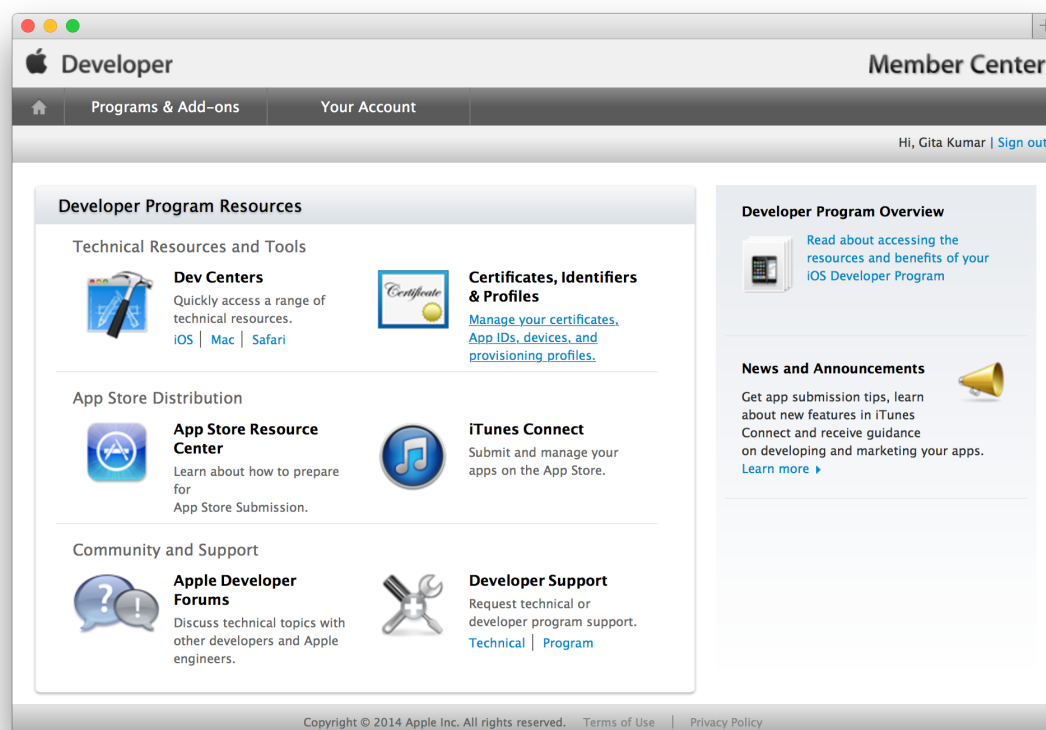
3. Click Done to close the dialog.

Verify Your Member Center Credentials

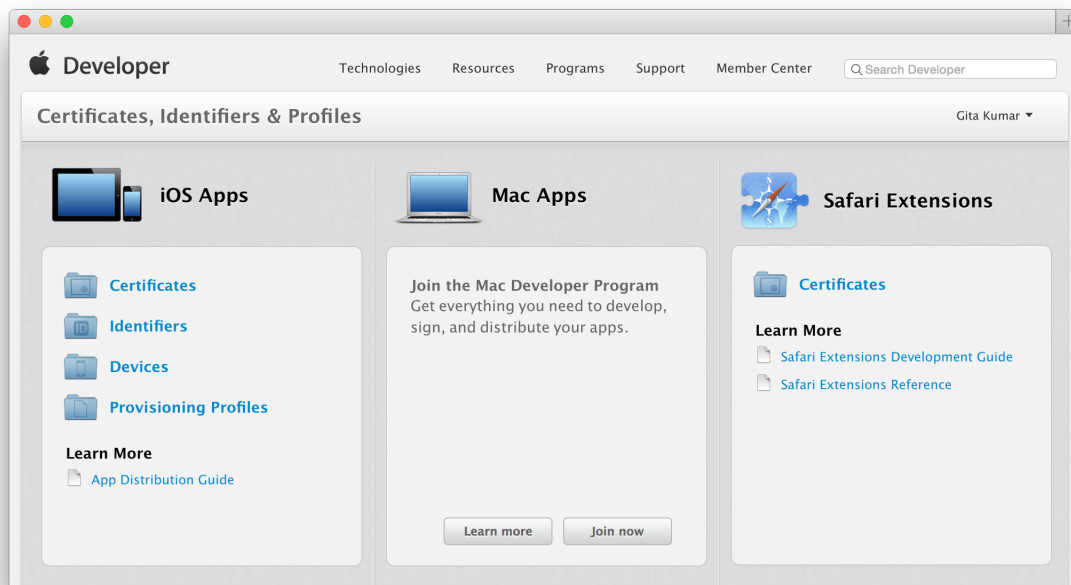
Verify that you have access to Member Center and that the assets in the Certificates, Identifiers, and Profiles area match the assets in Xcode.

To view your code signing and provisioning assets in Member Center

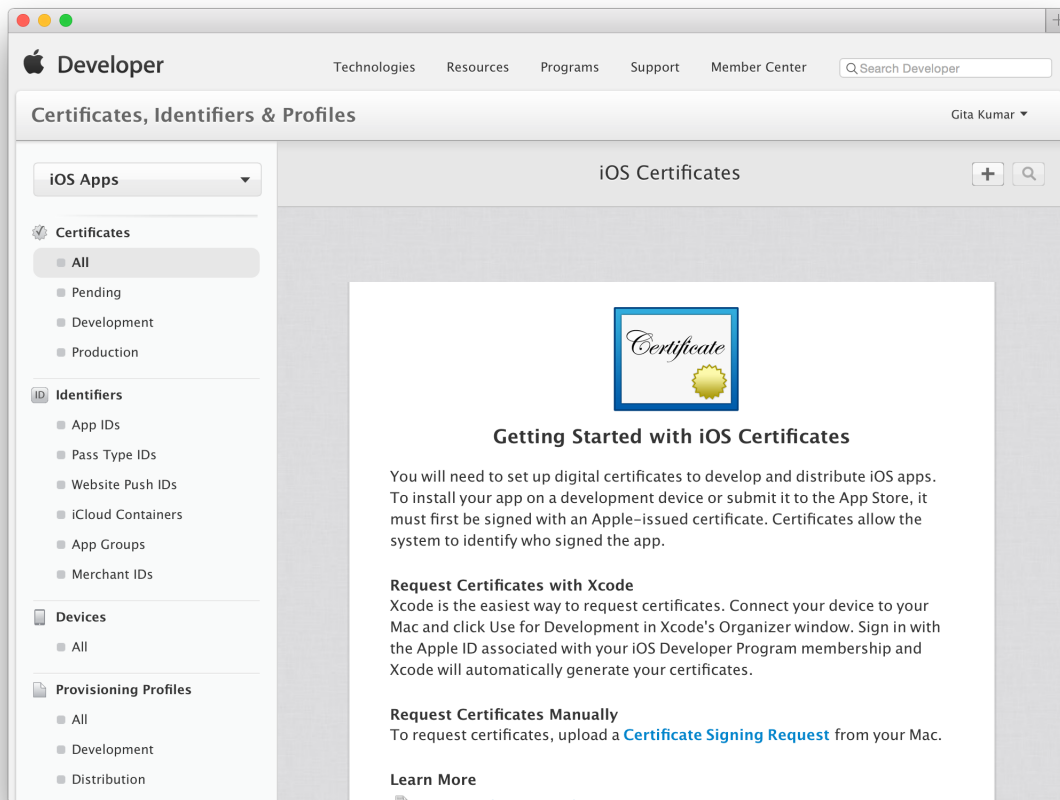
1. In your browser, go to developer.apple.com.
2. In the upper-right corner, click Member Center.
3. In the dialog that appears, enter your Apple ID and password, and click Sign In.
4. In the “Technical Resources and Tools” section under Developer Program Resources, click the icon or text below Certificates, Identifiers & Profiles.



5. Under iOS Apps or Mac Apps, click either Certificates, Identifiers, Devices, or Provisioning Profiles to view the assets for that developer program.



The certificates and provisioning profiles you view in Member Center should match the signing identities and provisioning profiles that appear in Xcode. If your membership is new, a Getting Started webpage is displayed instead.



Alternatively, go to developer.apple.com/account from your browser to go directly to the Certificates, Identifiers & Profiles webpage. You'll use Member Center in other verification steps, so don't sign out yet.

Verify Your iTunes Connect Credentials

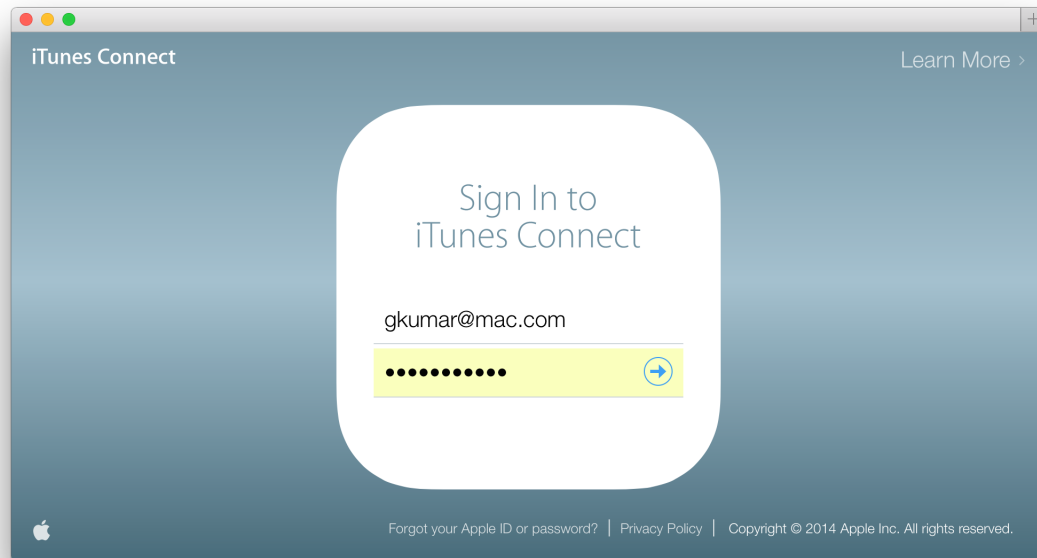
Once you join a developer program, you have access to iTunes Connect where you enter metadata about your organization and your app, including information about some app services—In-App Purchase and Game Center—that you may enable later. If you're the team agent—the person who joined the developer program—you have access to iTunes Connect automatically. If you're a team admin or team member for a company, you won't have access to iTunes Connect until the team agent creates an iTunes Connect account for you.

Although iTunes Connect is not used in this document, verify your iTunes Connect credentials now.

To go to iTunes Connect from Member Center

1. If necessary, sign in to [Member Center](#).

2. In the Developer Program Resources section under App Store Distribution, click the icon or text below iTunes Connect.
3. Enter your Apple ID and password, and click Sign In.



Later, you'll enter information about your app using iTunes Connect.

4. In the upper-right corner, choose Sign Out from the pop-up menu.

Alternatively, enter itunesconnect.apple.com in your browser to sign in to iTunes Connect.

Recap

In this chapter, you learned how to add your Apple ID to Xcode and add a developer program, if necessary. You also learned how to sign into Member Center and iTunes Connect. In the next chapter, you'll learn how to create your signing identity and team provisioning profile.

Creating Your Team Provisioning Profile

If you enter information about your app and assign a team to your app, Xcode creates the necessary signing identities and provisioning profiles for you. Specifically, Xcode creates a team provisioning profile that allows your app to run on all your devices and use app services. For iOS apps, you also connect the iOS device to the Mac that you want to use for testing.

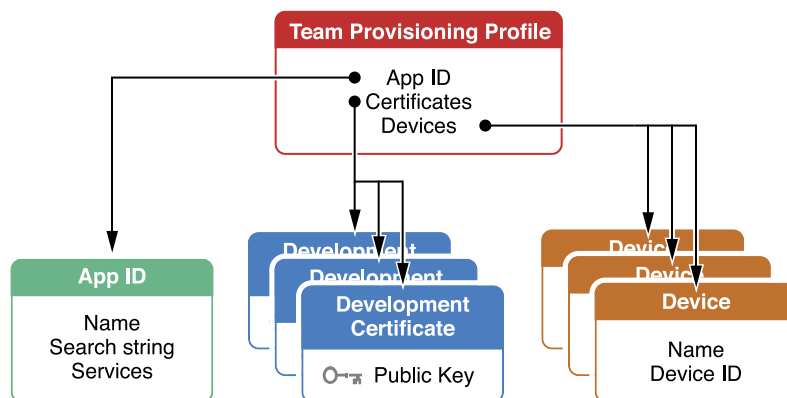
Mac Note: If you only enable App Sandbox, you're not required to use a development provisioning profile to code sign. However, it's more convenient and less error-prone to use the team provisioning profile that Xcode manages for you than to set the code signing build settings yourself.

About Team Provisioning Profiles

Provisioning is the process of preparing and configuring an app to launch on devices and use app services. During development, you choose which devices can run your iOS app and which app services your iOS or Mac app can access. A **provisioning profile** is downloaded from Member Center and embedded in the app bundle, and the entire bundle is **code signed**. The embedded provisioning profile is installed on the iOS or OS X system before the app is launched. If the information in the provisioning profile doesn't match certain criteria, your app won't launch. You indirectly configure a **development provisioning profile** using Xcode.

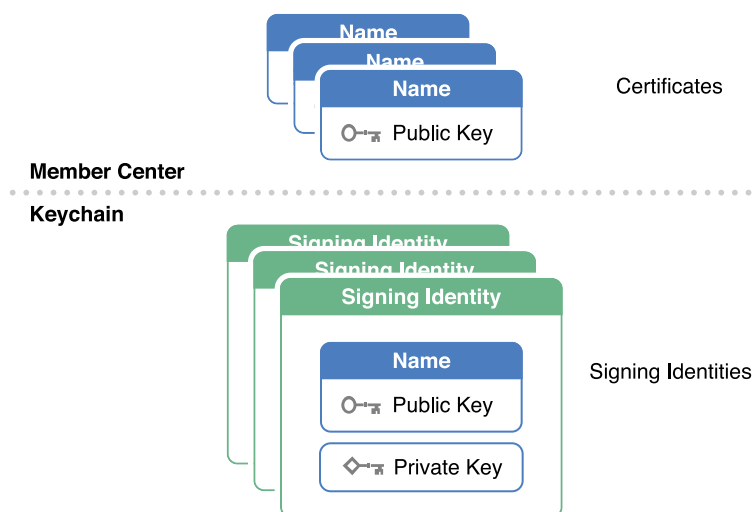
To save you time, Xcode creates and manages a type of development provisioning profile, called a **team provisioning profile**, for you. The team provisioning profile allows all your apps to be signed and run by all team members on all their devices. For an individual, the team provisioning profile allows all your apps to run on all your devices. Xcode also performs configuration steps for each app service you enable that requires some type of provisioning. Although Xcode simplifies this process, it helps to understand your code signing and provisioning assets and to know where they reside.

Xcode creates a team provisioning profile and its components as needed in Member Center. Xcode requests a development certificate for you if it's missing. The **development certificate** is used in the team provisioning profile to identify you. A device must be registered to create a provisioning profile, so Xcode may ask you to connect an iOS device. For Mac apps, Xcode automatically registers the Mac that is running Xcode.



Xcode creates an App ID to match your **bundle ID**, which is a unique identifier for your app stored in the app bundle. An **App ID** is used to identify one or more of your apps. An App ID is compared to the app's bundle ID to determine if it matches. It can be an **explicit App ID**, which matches a single app, or a **wildcard App ID**, which matches multiple apps. Xcode initially creates a wildcard App ID and an explicit App ID only if needed. For example, if you enable an app service that requires an explicit App ID, Xcode creates and uses an explicit App ID in a new team provisioning profile it also creates. Therefore, you may have one team provisioning profile used by all your apps or multiple app-specific team provisioning profiles in Member Center.

You use a signing identity to code sign your app. When Xcode requests your development certificate, the certificate with its public key is stored in Member Center, and the **signing identity**—the certificate with its public and private key—is stored in your keychain. You can't code sign without this private key.



Set the Bundle ID

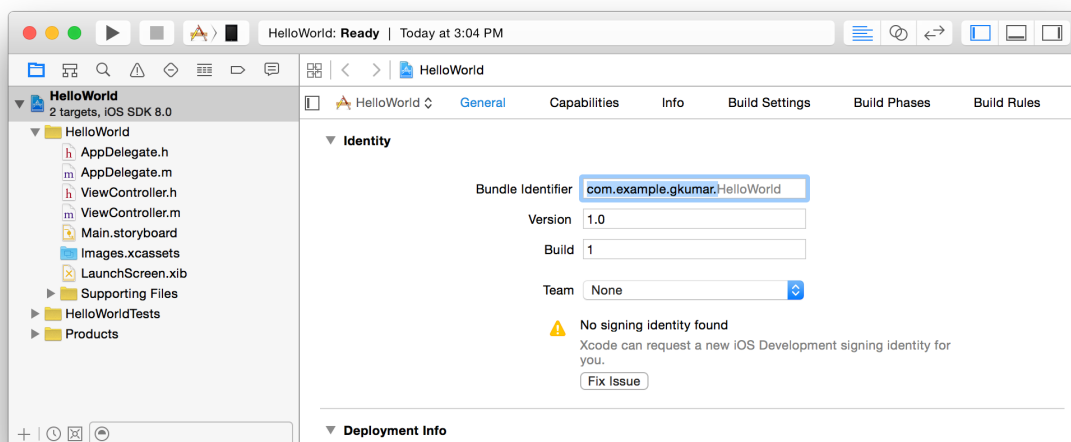
After an app is available on the store, its bundle ID can't be changed. Verify it now, and change it if necessary.

The App Store and services use the bundle ID to uniquely identify a single app. The bundle ID string should be in reverse DNS format. To create the default bundle ID for a new project, Xcode concatenates the company identifier with the product name—for example, it concatenates `com.example.ajohnson` with `MyFirstApp` to create `com.example.ajohnson.MyFirstApp`. Unlike domain names, bundle IDs are case sensitive. You may prefer to use a lowercase bundle ID or a different prefix.

To change the bundle ID, replace the company identifier prefix in the bundle ID. Alternatively, replace the entire bundle ID in the `Info.plist` file or in the project editor Info pane. Later, you'll enter the same bundle ID in iTunes Connect.

To change the bundle ID prefix in Xcode

1. To display the project navigator, 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 **General**, and if necessary, click the disclosure triangle next to **Identity** to reveal the settings.
4. In the **Bundle Identifier** field, replace the bundle ID prefix.



Assign Your App to a Team

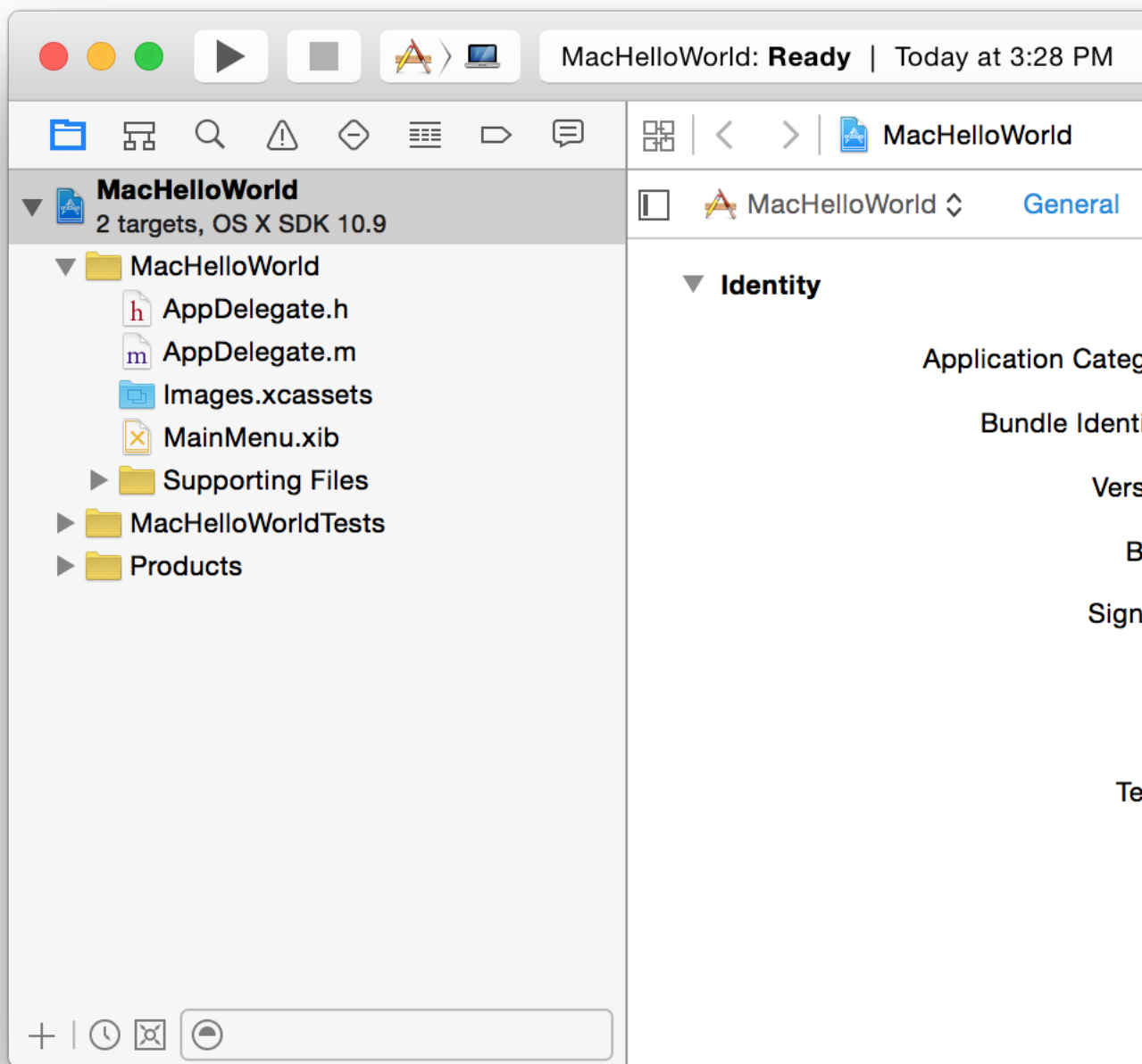
You can join multiple teams using the same Apple ID—for example, you can enroll as an individual and later join another team. Because of this, the Xcode project needs to be assigned to a team so that Xcode knows where to create your code signing and provisioning assets.

To assign the Xcode project to a team

1. In the project navigator, view the Identity settings.

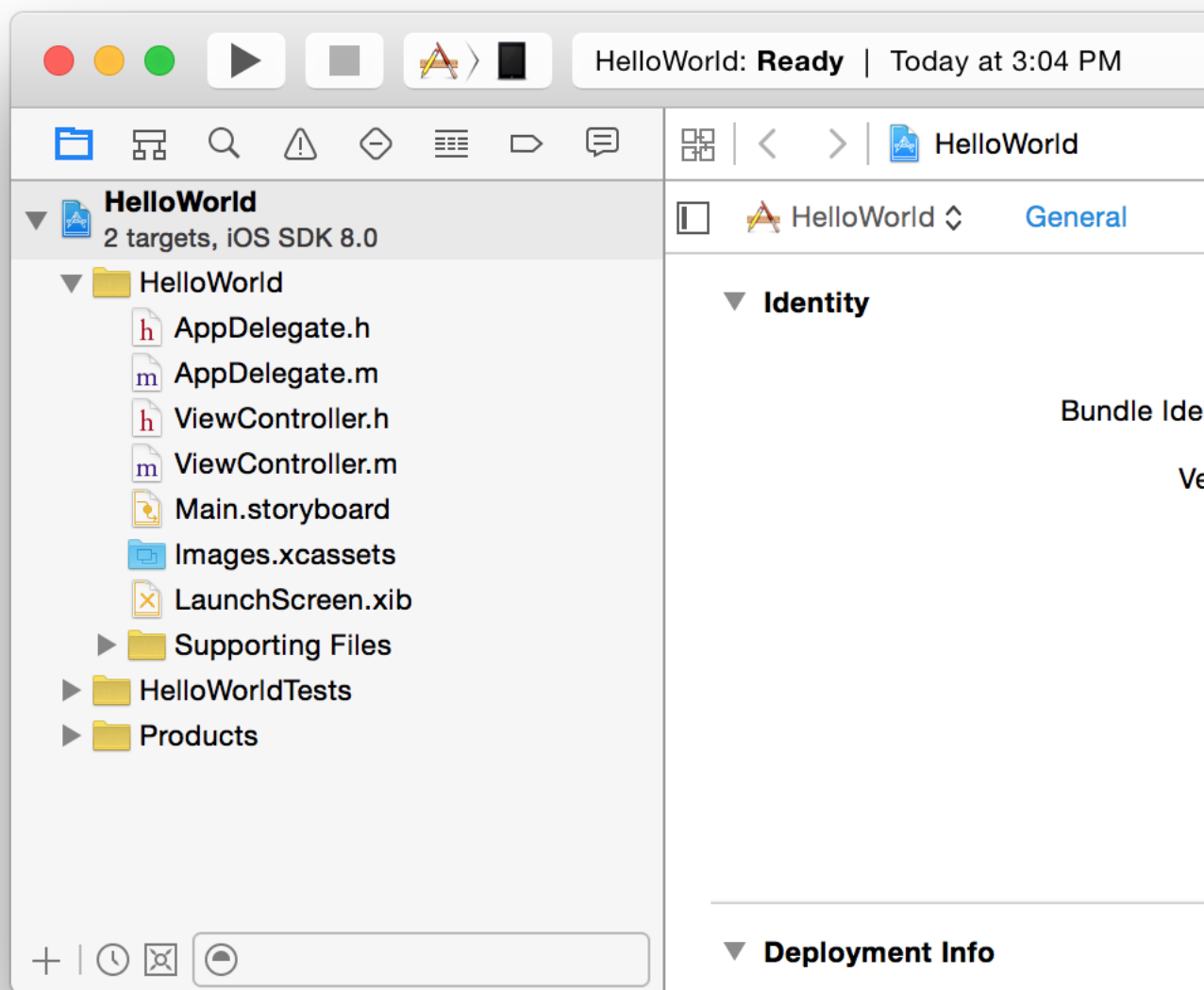
If necessary, select the target, click General, and click the disclosure triangle next to Identity to reveal the settings.

- For Mac apps, select Mac App Store as the type of signing identity, to enable the Team pop-up menu.



- Choose your team from the Team pop-up menu.

If you're an individual, choose your name from the pop-up menu.



Xcode may attempt to create a team provisioning profile if you have a device connected or have previously registered a device. The warning message under the Team pop-up menu may change. You'll resolve any issues in the next section.

Create the Team Provisioning Profile

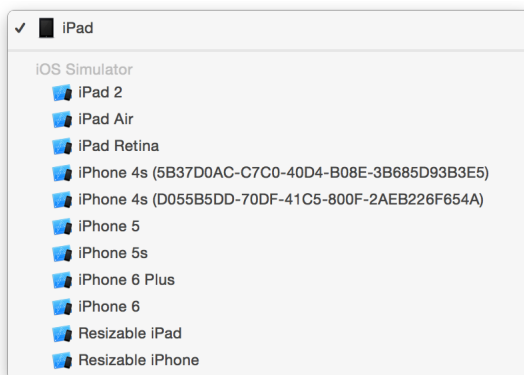
Xcode saves you time by performing multiple steps on your behalf to create the team provisioning profile. For example, Xcode automatically registers your device and requests your development certificate, both of which are needed to create a team provisioning profile. Xcode performs these steps when you click the Fix Issue button below the Team pop-up menu. Xcode may also perform these steps when you assigned a team to your project or refresh provisioning profiles in the Accounts preferences. If no warning message and Fix Issue button appear below the Team pop-up menu, Xcode already created your team provisioning profile and you can skip this section.

Team Member Note: A team agent or admin needs to create the team provisioning profile on behalf of team members. The team agent or admin also registers team member devices and approves their development certificates. If you're a team member, skip [Generate a Team Provisioning Profile](#) (page 25) and instead send the device ID to your team agent or admin and request your development certificate using Xcode Accounts preferences, as described in Requesting Signing Certificates. Wait for the team agent or admin to create a team provisioning profile containing your assets, and then refresh provisioning profiles in Xcode before continuing with [Export Your Signing Identities](#) (page 28).

To create a team provisioning profile

1. In the project navigator, if necessary, select the target, click General, and click the disclosure triangle next to Identity to reveal the settings.
2. For iOS apps, connect an iOS device you want to use for development.

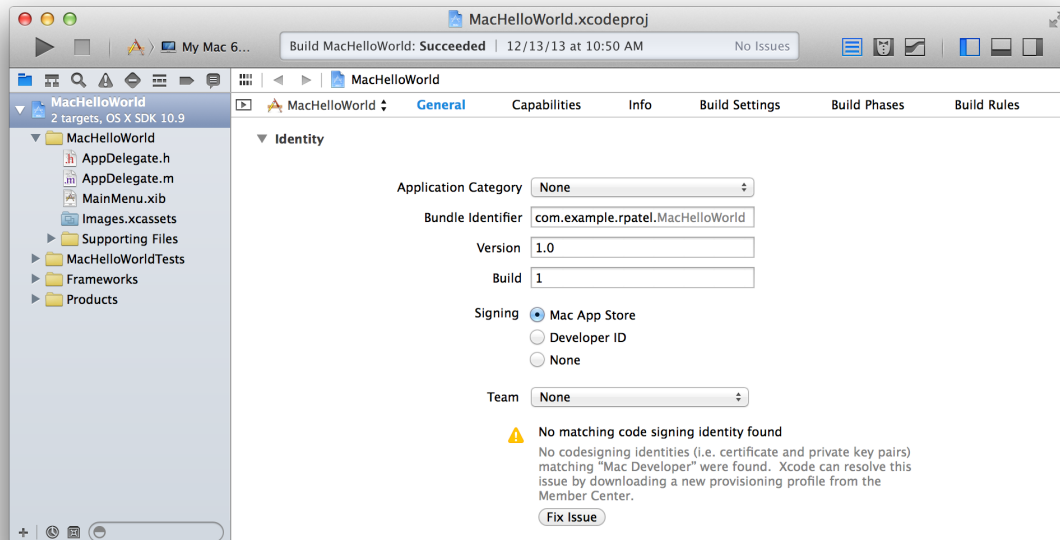
An iOS device needs to be connected to your Mac and eligible to be registered. An iOS device is eligible if you can select it from the Scheme pop-up menu.



If your iOS device 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 pop-up menu.

3. 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.

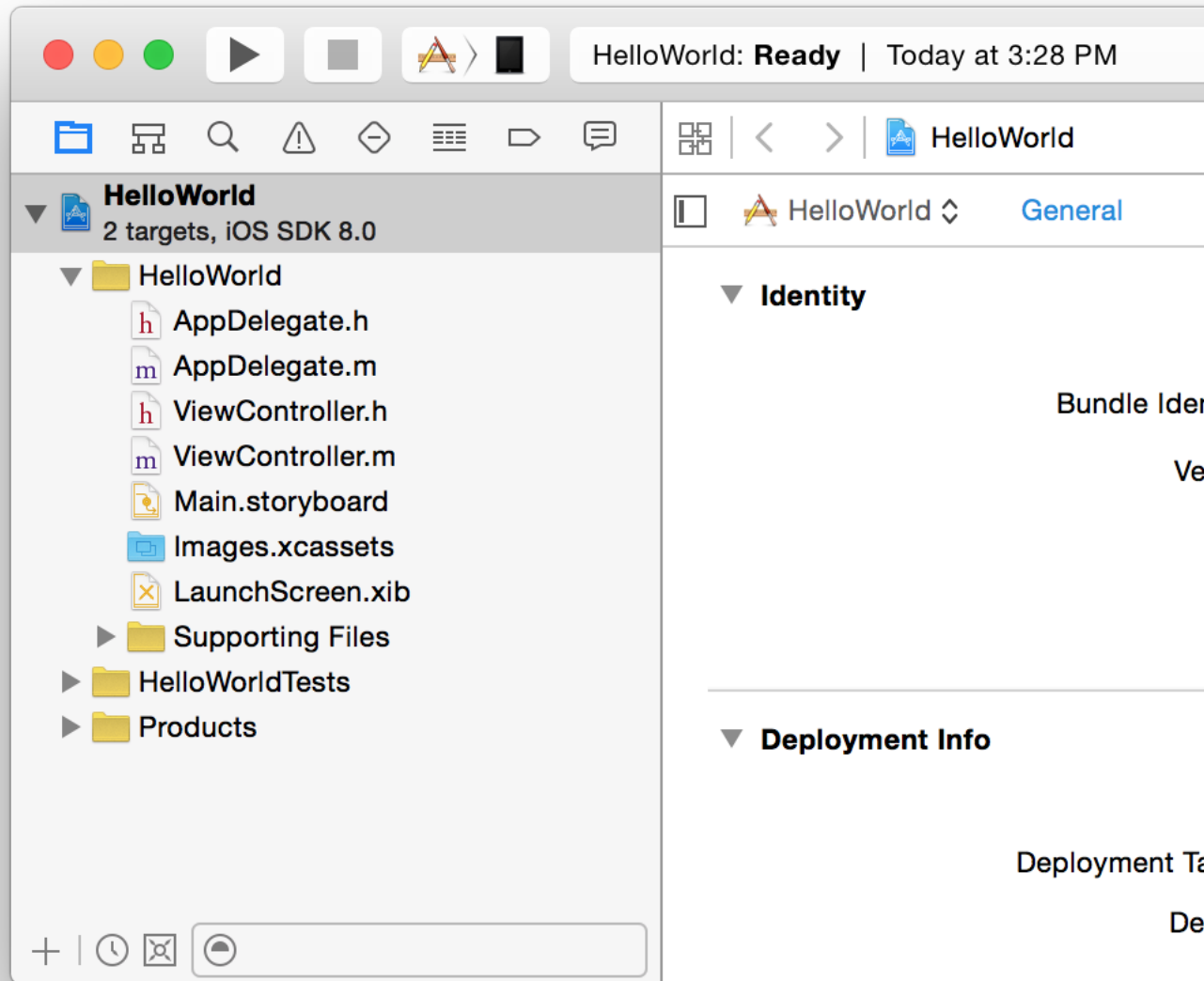


4. If a Fix Issue button appears below the Team pop-up menu, click it.

Xcode starts performing all the steps needed to create a team provisioning profile for your app.

Wait while Xcode performs all the steps needed to create a team provisioning profile for your app. Xcode may request your development certificate before creating the team provisioning profile. If Xcode successfully creates your team provisioning profile, the warning message under the Team pop-up menu in the General pane disappears.

The screenshot below shows the General pane for an iOS app when Xcode successfully creates the team provisioning profile.



Export Your Signing Identities

After you create the team provisioning profile, export your signing identities to back up your private keys. Xcode exports all your signing identities and provisioning profiles to a file. The signing identities include your public and private keys so that you can import the file on another Mac and continue code signing your app. You can always download provisioning profiles from Member Center so you primarily export your assets to back up your private keys. Because your signing identities represent your App Store credentials, the file is encrypted and should be stored in a safe place. If you delete the private key from your keychain and lose this backup file, you need to re-create the certificate.

To export your signing identities

1. Choose Xcode > Preferences.
2. At the top of the window, click Accounts.
3. Click the Action button (the gear icon to the right of the Delete button) in the lower-left corner.

4. Choose Export Accounts from the pop-up menu.

5. Enter a filename in the Save As field, and enter a password in both the Password and Verify fields.

The file is encrypted and password protected.

6. Click Save.

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

7. In the dialog that appears, click OK.

Recap

In this chapter, you learned how to enter identity information about your app in Xcode and create your team provisioning profile. You also learned how to back up your signing identities to keep them safe. In the next chapter, you'll learn how to code sign and launch your app on a device through Xcode.

Launching Your App on Devices

All iOS apps and Mac apps that use app services must be provisioned and code signed to launch on a device. When you created the team provisioning profile in an earlier chapter, Xcode set the project code signing and provisioning build settings for you. Now when you click the Run button, Xcode builds your app, embeds the team provisioning profile in the app bundle, and code signs the app. For iOS apps, the app launches on a device (an iPad, iPhone, or iPod touch) if that device is in the team provisioning profile. Similarly, a Mac app that uses certain app services launches if the Mac is in the team provisioning profile. Therefore, if your iOS device or Mac is not registered, Xcode registers and adds it to the team provisioning profile before building your app.

About Devices

A **device ID** uniquely identifies an iPhone, iPad, iPod touch, or a Mac. Your team can register up to 100 devices per year to use for development and testing. As a convenience, all devices you register through Xcode are automatically added to the team provisioning profile. (The team provisioning profile contains an App ID that matches one or more of your apps, all development certificates for team members, and all registered devices.)

Alternatively, if you're a team agent or an admin for a company, you can collect the device IDs from your team members and bulk register them using Member Center. You can also disable devices using Member Center, but the device count for your team isn't reduced until the time of your annual membership renewal. Because Xcode manages the team provisioning profile, you need to regenerate the team provisioning profile using Xcode whenever you register or disable devices in Member Center.

Launch Your App

It takes just a few steps to launch your iOS app on a device, and just one click to launch your Mac app. Unlike a Mac app, an iOS app requires you to connect and select an iOS device before clicking the Run button.



Tip: To avoid dialogs and warning messages in Xcode, create your team provisioning profile, as described in [Creating Your Team Provisioning Profile](#) (page 19), before launching your app on a device.

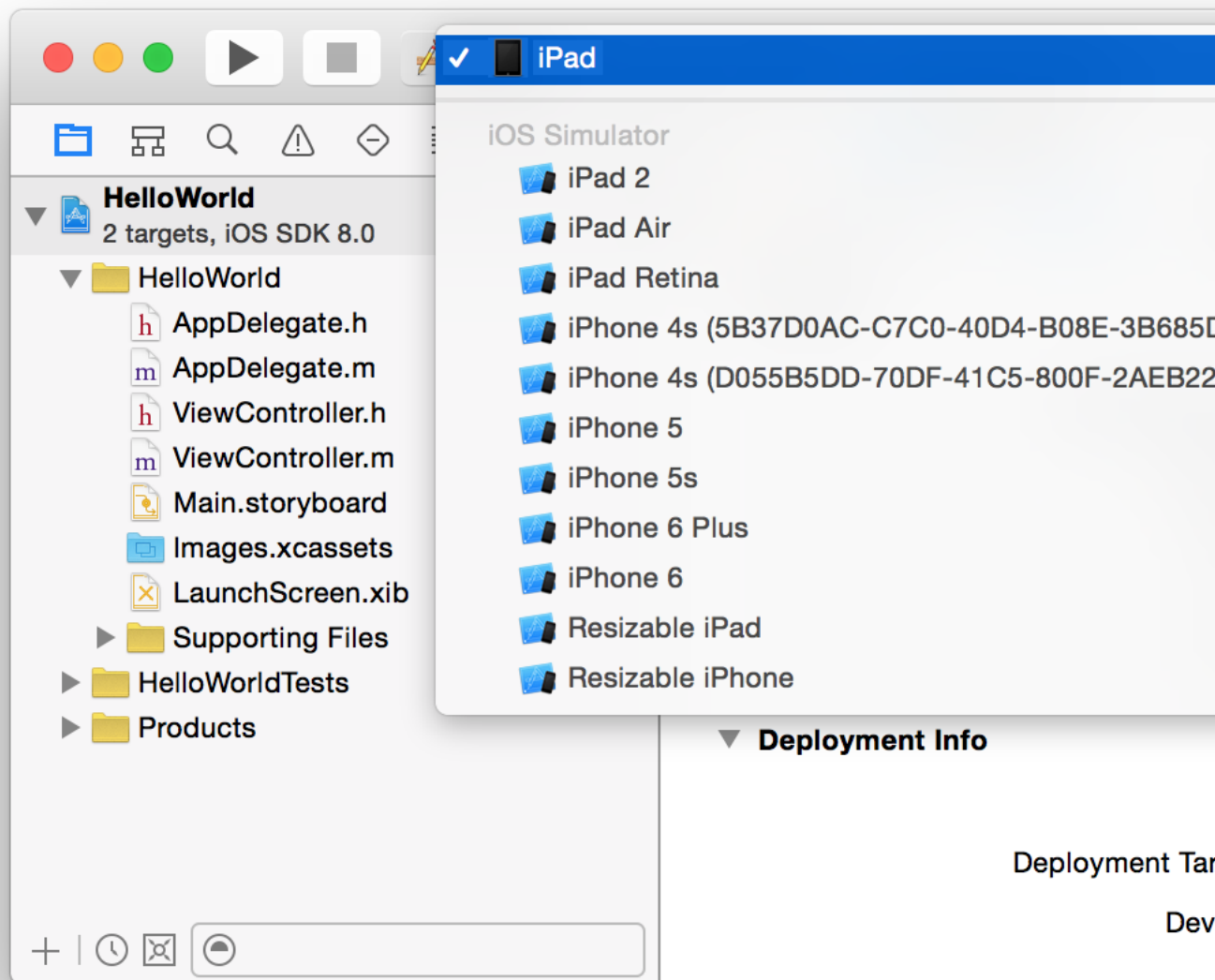
Launch Your iOS App on a Device

You can use the same device that you added to the team provisioning profile in an earlier step or connect another device that matches your deployment target.

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 destination Scheme pop-up menu.



If your iOS device is not selectable from the Scheme pop-up 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 pop-up menu.

3. Click the Run button.

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

4. If a “No matching provisioning profile found” warning message appears under the Team pop-up menu, click the Fix Issue button.

If you connect a device which isn’t in the team provisioning profile, Xcode needs to add it to the team provisioning profile before it can launch the app on the device.

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

Launch Your Mac App

To launch your Mac app, click the Run button in the project navigator. If necessary, 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.

Verify Your Profiles

To better understand how Xcode manages the team provisioning profile for you, compare your code signing and provisioning assets in Xcode, Member Center, and Keychain Access.

Verify Signing Identities and Profiles Using Xcode

Xcode shows the signing identities that appear in your keychain and any provisioning profiles it downloads from Member Center. After you launch your app on a device, your development certificate and team provisioning profile should appear in Xcode.

To view signing identities and provisioning profiles in Xcode

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. For iOS apps, an iOS Development signing identity and an iOS Team Provisioning Profile appear. For Mac apps, a Mac Development signing identity and a Mac Team Provisioning Profile appear.

In the Provisioning Profiles table, notice that the cell in the team provisioning profile row and Entitlements column is blank. You'll learn more about setting entitlements in the next chapter.

4. Click Done to close the dialog.

Verify Your Certificates, Identifiers, and Profiles Using Member Center

Member Center stores all your certificates, identifiers (including device IDs and App IDs), and provisioning profiles.

View Your Certificates

Initially, you'll use a development certificate to run your app on devices, but later you'll create different types of certificates to distribute your app. Only the public keys for your certificates are stored in Member Center.

To view certificates in Member Center

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

The name, type, and expiration date of the certificate should match the information that you view in Xcode. For iOS apps, an iOS Development certificate should appear and for Mac apps, a Mac Development certificate should appear. The screenshot below shows an iOS certificate.



View Your Devices

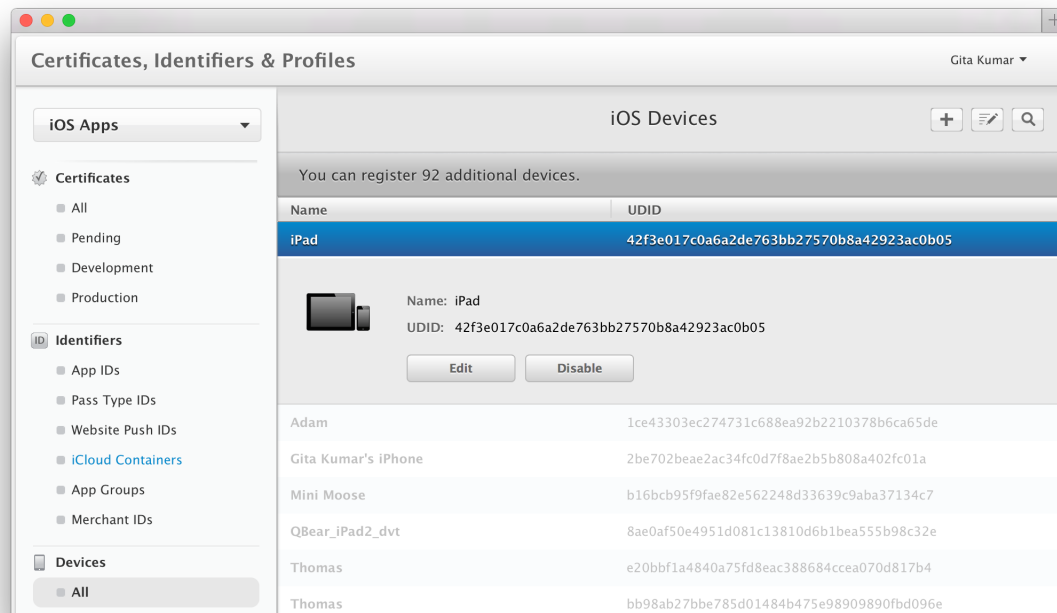
Because you can register devices using Xcode and Member Center, it's a good idea to occasionally verify all the devices for your team.

To view your devices

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

2. In the Devices section, select All.

The name and identifier for your device should appear under iOS Devices or Mac Devices. Enabled devices appear in black text, and disabled devices appear in gray text. The screenshot below shows an iOS device selected in the list.



View Your App ID and Team Provisioning Profile

Initially, Xcode registers a wildcard App ID and creates a corresponding team provisioning profile on your behalf that can be used by all your apps. Later, when you enable app services, Xcode may create an explicit App ID that matches your app's bundle ID.

To view your wildcard App IDs and team provisioning profile

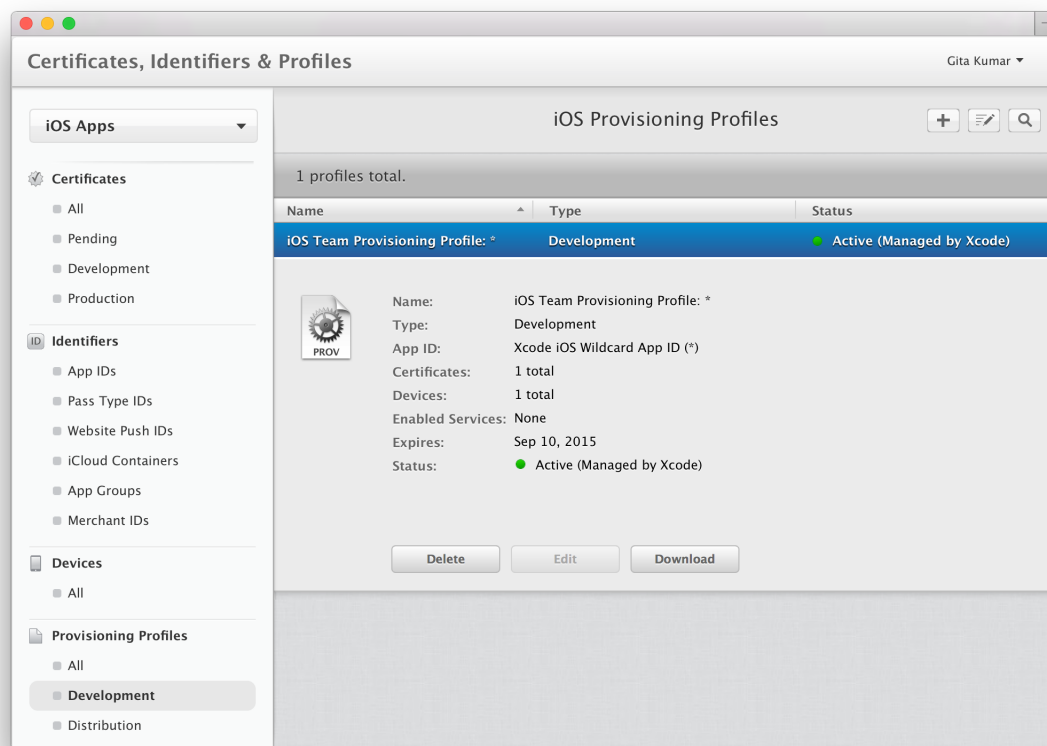
1. In [Certificates, Identifiers & Profiles](#), select Identifiers.
2. In the Identifiers section, select App IDs.

For iOS apps, the wildcard App ID that Xcode registers is called Xcode iOS Wildcard App ID. For Mac apps, it's called Xcode Mac Wildcard App ID. Explicit App IDs that Xcode registers for you begin with the text "Xcode iOS App ID" or "Xcode Mac App ID."

3. Select Provisioning Profiles.
4. In the Provisioning Profiles section, select All or Development.

For iOS apps, the team provisioning profile begins with the text "iOS Team Provisioning Profile." For Mac apps, it begins with the text "Mac Team Provisioning Profile."

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



Verify that the number of certificates and devices equals the number of development certificates and devices you viewed in the Certificates and Identifiers section. If you're an individual, the number of development certificates in the team provisioning profile should be 1. Enabled Services should be None.

Verify Signing Identities Using Keychain Access

Your signing identity—which includes the certificate with its public and private key—is stored in your keychain. Xcode alerts you if there's a problem with a signing identity, but it's useful to know where signing identities appear in Keychain Access in case you need to troubleshoot them yourself.

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.

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).

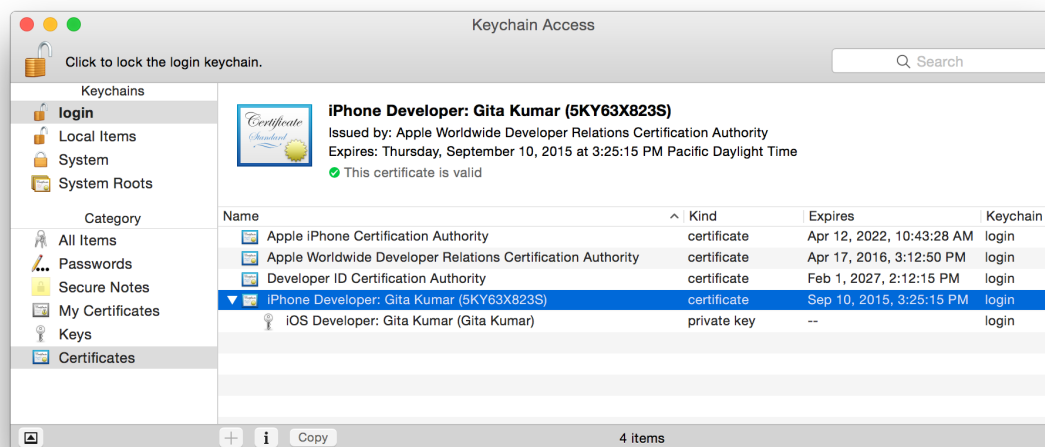


3. Verify that the certificates are valid.

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.”

4. To view your private key, click the disclosure triangle to the left of the certificate name.

If the disclosure triangle doesn’t appear, your private key is missing. If this happens, you’ll need to import the signing identities from another Mac or re-create them.



Recap

In this chapter, you learned how to launch your app on devices. You also learned more about Xcode-managed team provisioning profiles by using different tools to view your code signing and provisioning assets.

Enabling App Services

Certain app services—such as iCloud and push notifications—are available only to apps distributed through the store and require additional configuration in your Xcode project, Member Center, and sometimes iTunes Connect. To avoid code signing and provisioning issues, let Xcode perform the configuration steps for you when you enable these app services in your Xcode project. There should be no need for you to configure the Xcode project or enable these app services in Member Center yourself.

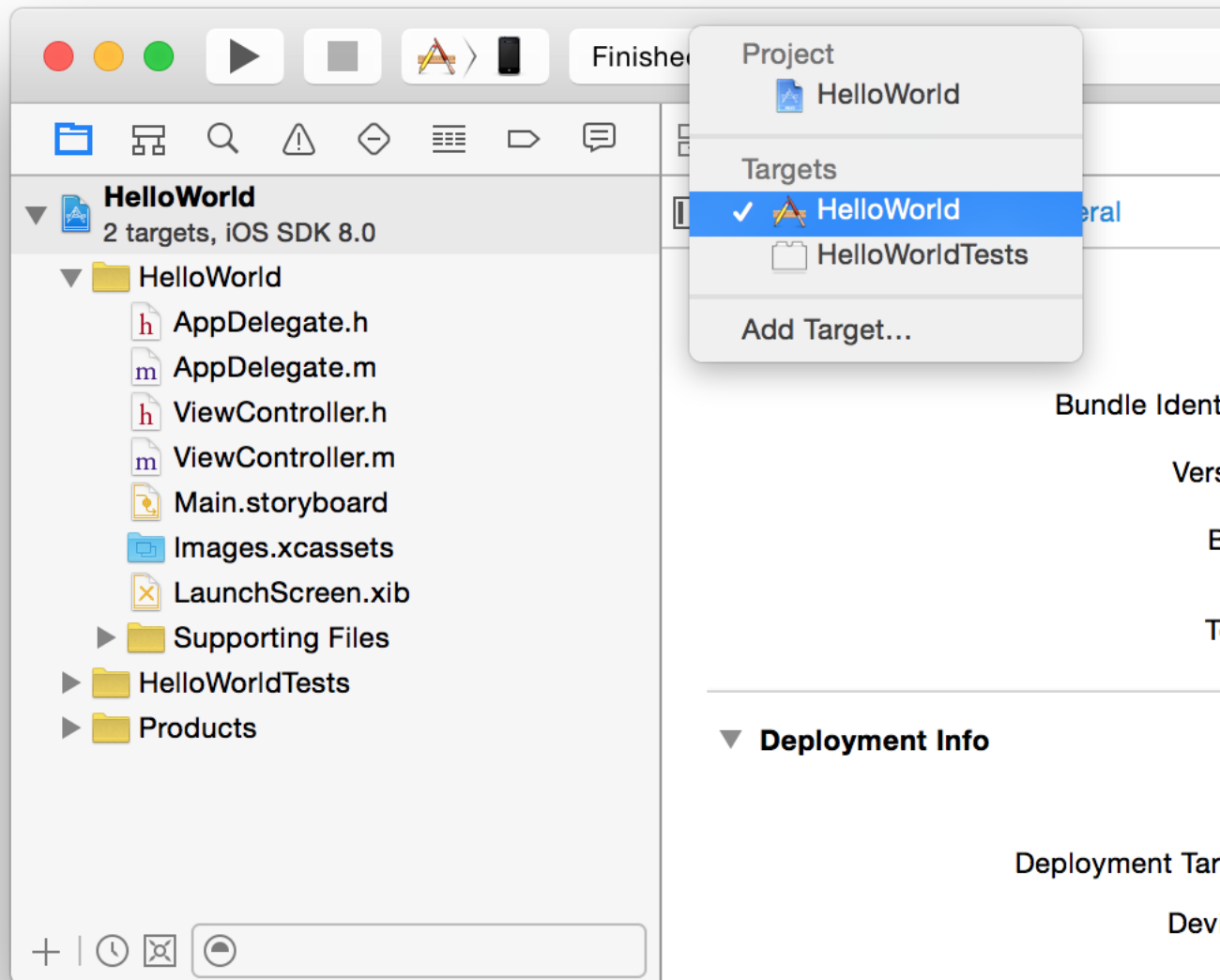
View App Services

The key app services you can add to your app are listed in the Capabilities pane in the project editor. Be sure to select the correct target in your project before enabling app services.

To open the Capabilities pane

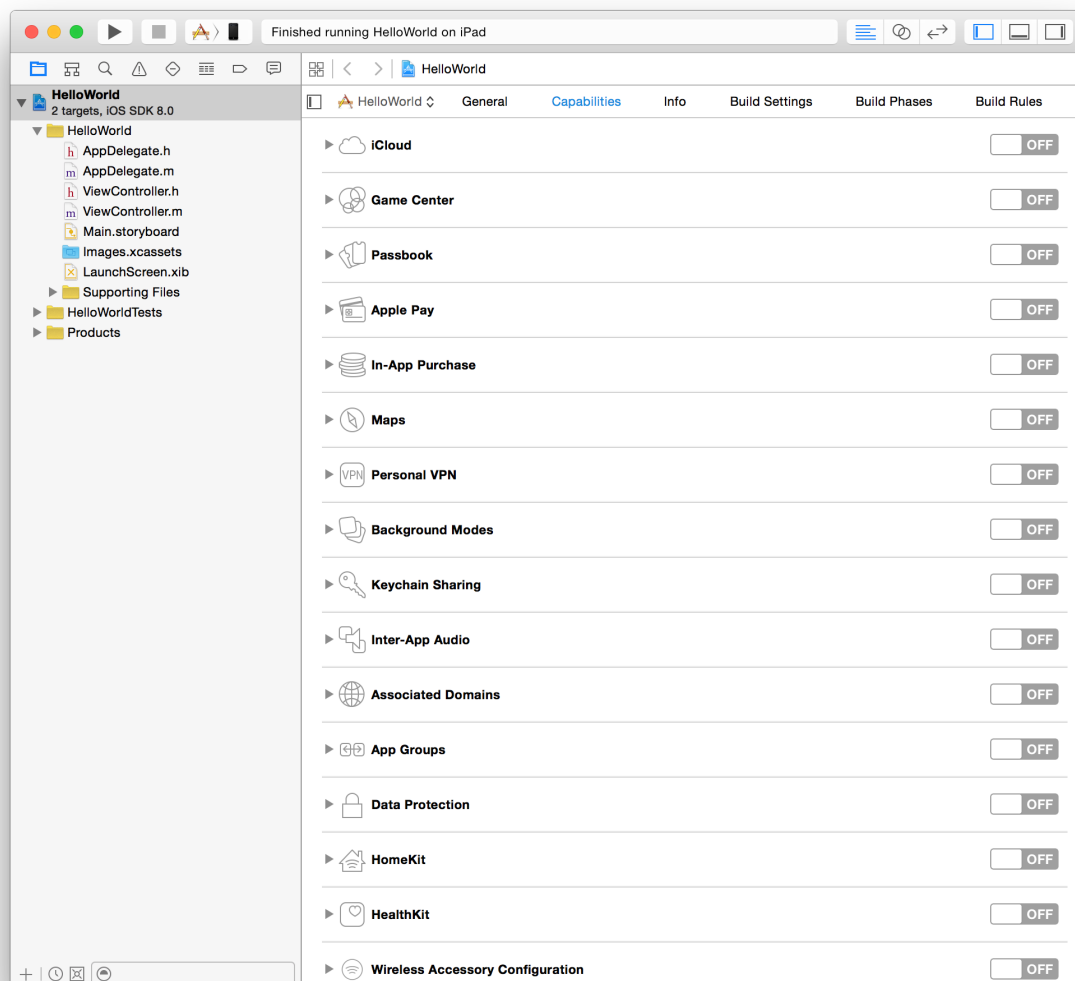
1. Choose View > Navigators > Show Project Navigator.

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 key app services.

The app services available for iOS and Mac apps are different. The screenshot below shows the Capabilities pane for an iOS app.

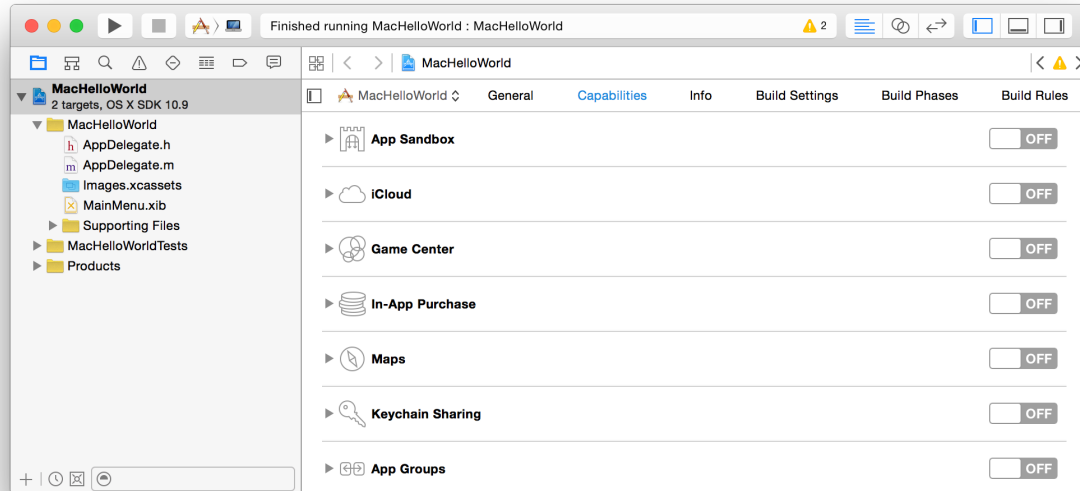


For Mac Apps, Enable App Sandbox

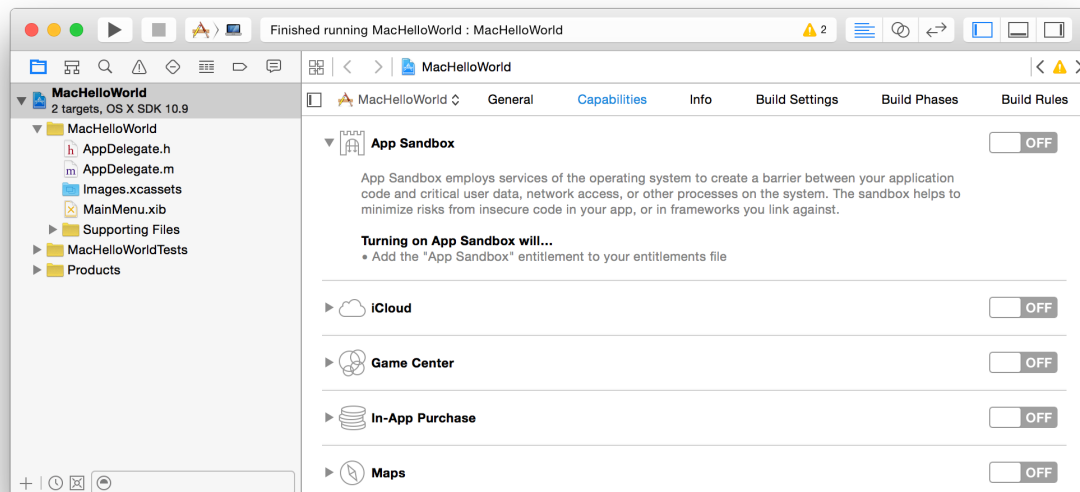
All apps submitted to the Mac App Store are required to use App Sandbox, so if you're developing a Mac app for the store, enable App Sandbox now.

To configure App Sandbox

1. In the project navigator, click Capabilities.

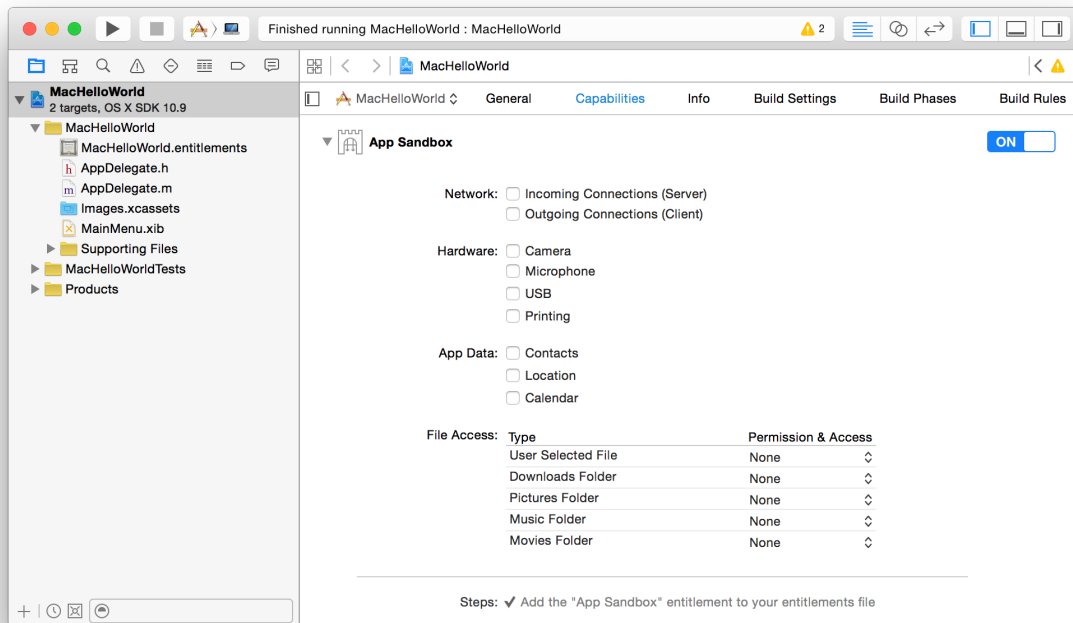


2. If App Sandbox isn't enabled, select the switch in the App Sandbox row.



Xcode adds an entitlements file to your project and enables the App Sandbox entitlement.

3. In the App Sandbox area, use the checkboxes to set additional entitlements for the app to do its job.



Enable App Services

In most cases, enabling a capability is as simple as turning on a switch. However, some app services—such as iCloud, Keychain Sharing, Passbook, and Background Modes—require more configuration. Additional options and instructions for those app services appear after the capability is enabled. Other app services—such as Game Center, In-App Purchase, and Maps—require additional configuration in iTunes Connect and are not covered in this document.



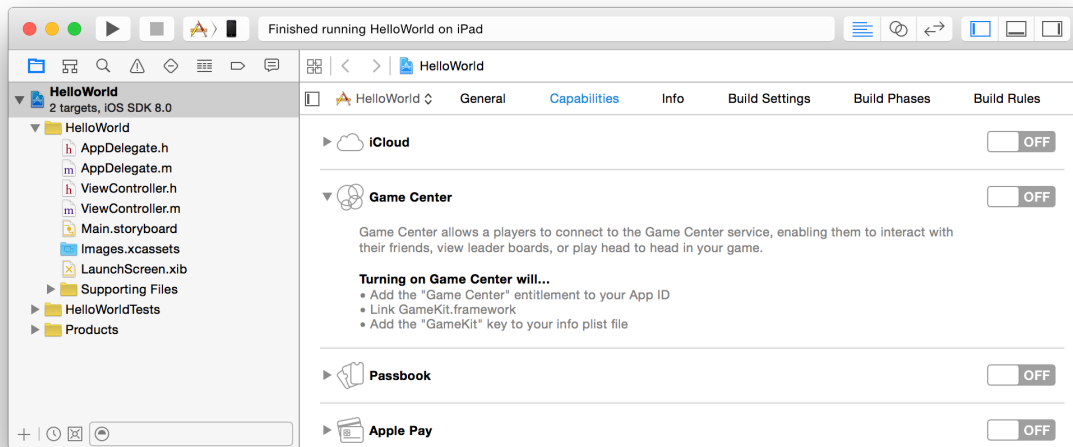
Tip: To avoid dialogs and warning messages in Xcode, create your team provisioning profile, as described in [Creating Your Team Provisioning Profile](#) (page 19), before enabling app services.

Team Member Note: You must be a team agent or admin to enable some app services.

To enable a capability

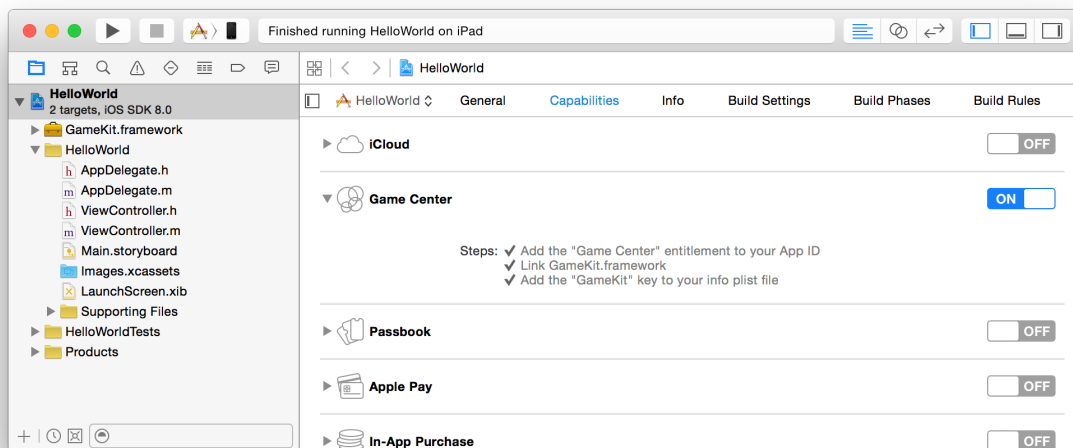
1. In the project navigator, click Capabilities.
2. Click the disclosure triangle next to a capability and read its description.

Xcode displays the steps it will perform to enable the capability, so you don't need to perform these steps yourself. For example, to view Game Center configuration steps, click the disclosure triangle next to Game Center.



3. Select the switch in the row of the capability you want to enable.

For example, to enable Game Center, select the switch in the Game Center row. If Xcode successfully enables the capability, checkmarks appear to the left of the configuration steps, as in the screenshot below for an iOS app:



Some configuration steps include adding frameworks to your project. For example, when you enable Game Center, GameKit framework is added to your project.

4. If an error occurs, follow the instructions in the error message, and click the Fix Issue button to resolve the problem.

For example, Xcode may fail to communicate with Member Center and complete all the steps. When you click the Fix Issue button, Xcode attempts to perform any remaining steps.

After enabling app services in your Xcode project, click the Run button in the project navigator to build and run your app using the new team provisioning profile.

Verify App ID Entitlements

To better understand how provisioning works, verify that Xcode adds entitlements to your App ID (according to the app services you enabled) and regenerates the corresponding team provisioning profile.

Remember that Xcode initially creates a team provisioning profile containing a wildcard App ID that can be used to sign multiple apps and enable some app services. For example, if you enable Passbook, the Passbook entitlement is added to the wildcard App ID. Later, when you enable app services that require an explicit App ID, Xcode registers an explicit App ID that uniquely matches the app's bundle ID. Xcode then creates another team provisioning profile that contains the explicit App ID. If an App ID already exists for your app, Xcode uses it in the team provisioning profile instead. Therefore, you may see multiple team provisioning profiles in your account—one for the wildcard App ID and one for each explicit App ID in your account.

View Provisioning Profiles in Xcode

Whenever you create a development certificate, register a device, or change an entitlement, Xcode regenerates and downloads the team provisioning profile. You can view the downloaded provisioning profiles in Xcode.

To view your team provisioning profile in Xcode

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

In the dialog that appears, view iOS Team Provisioning Profile or Mac Team Provisioning Profile in the Provisioning Profiles table. The Entitlements column contains a whitelist of the services your app can use. If your team provisioning profile uses the Xcode-managed wildcard App ID—where the bundle ID search string is an asterisk (*)—the entitlements are a superset of all your apps that use this team provisioning profile. Place the pointer over the entitlements icons to see each entitlement key.

For example, Xcode uses the wildcard App ID if only Passbook is enabled in this screenshot for an iOS app:

If you enable iCloud, which requires an explicit App ID, Xcode creates a new provisioning profile using the explicit App ID, as shown in the iOS app screenshot below.

For iOS apps, if you enable an app service that requires an explicit App ID, both the Game Center and In-App Purchase entitlements are added to your App ID by default. Therefore, Game Center and In-App Purchase entitlements appear in the Entitlements column. For Mac apps, In-App Purchase is enabled by default for an explicit App ID. However, to use these services from your app, enable them in the Capabilities pane.

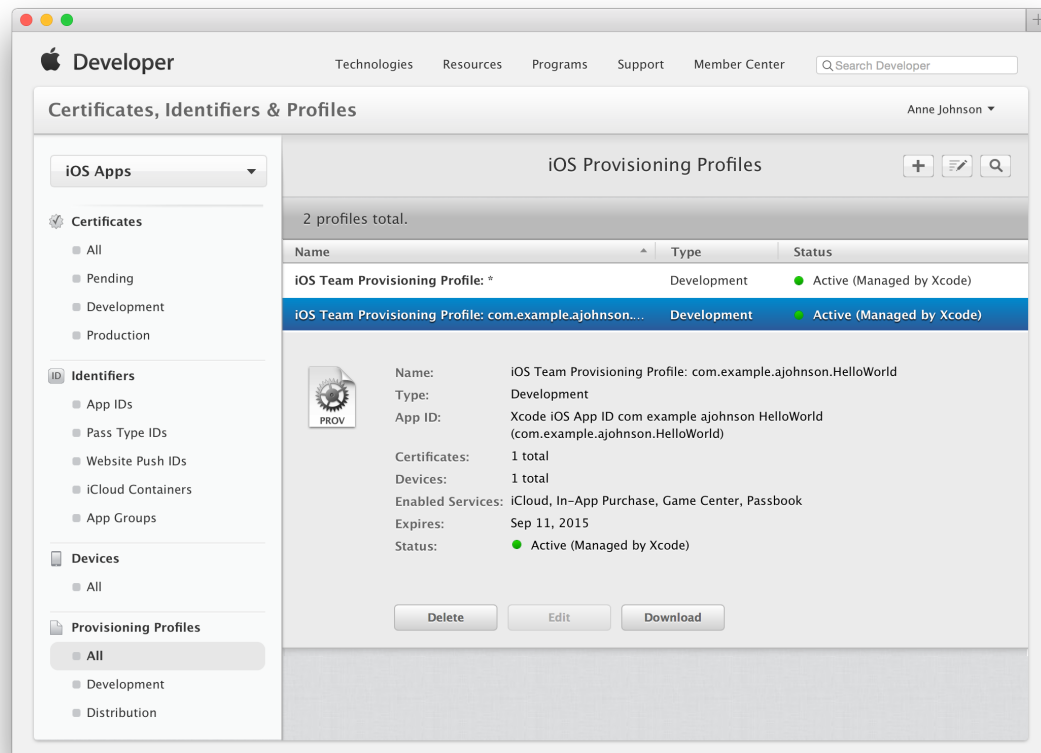
View Provisioning Profiles in Member Center

You can view the same entitlement settings in Member Center that you view in Xcode, but you can see more details of the provisioning profiles in Member Center.

To view your team provisioning profile in Member Center

1. In [Certificates, Identifiers & Profiles](#), under Provisioning Profiles, select All or Development.
If you add a capability that requires an explicit App ID, multiple active team provisioning profiles appear. Xcode uses the team provisioning profile containing the explicit App ID that matches your app's bundle ID; otherwise, Xcode uses the team provisioning profile containing the wildcard App ID.
2. Select iOS Team Provisioning Profile or Mac Team Provisioning Profile.

The entitlements appear in the Enabled Services list and should match the entitlements you viewed in the Accounts preferences in Xcode. The entitlements in Member Center may be a superset of the entitlements you added to your project using the Capabilities pane. (For iOS apps, In-App Purchase and Game Center are enabled by default for an explicit App ID.)



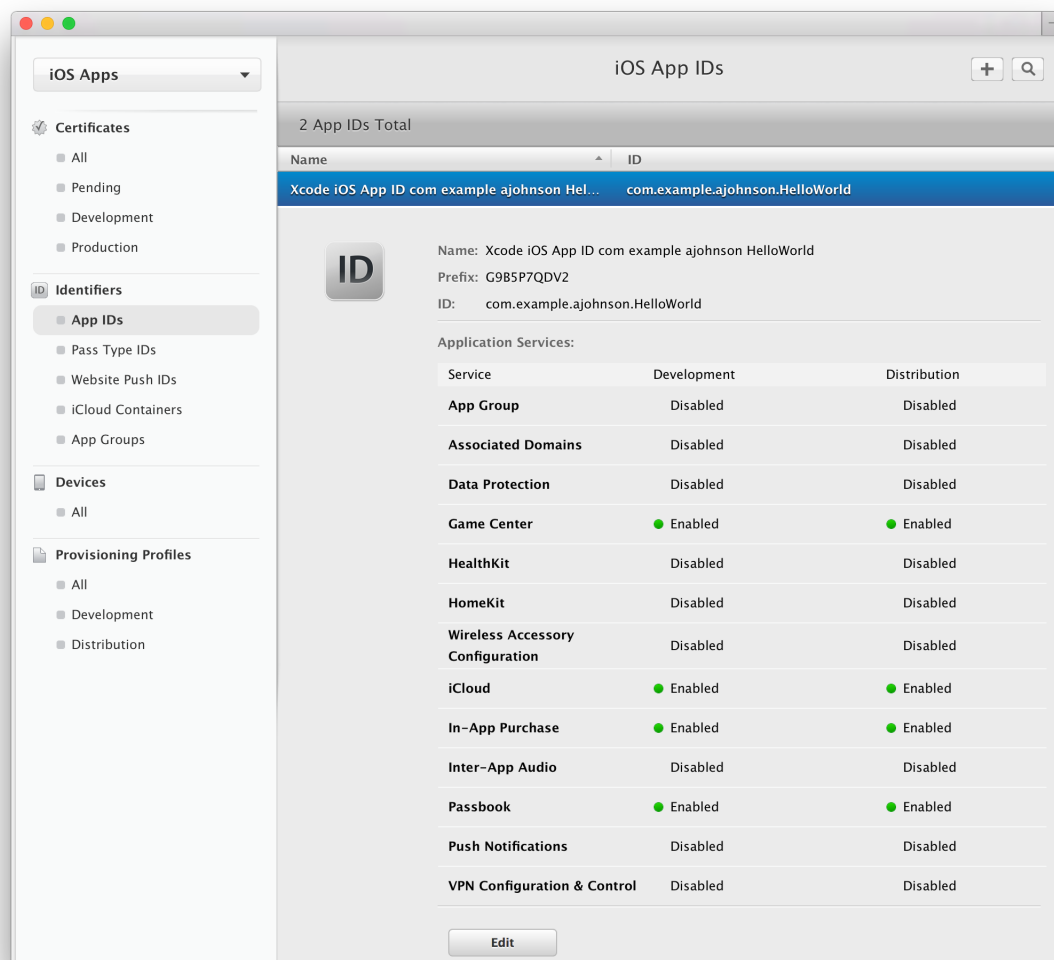
Verify the App ID Settings

Xcode adds entitlements to an App ID, not to a team provisioning profile, so verify App ID settings in Member Center to learn where this information is located.

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 your bundle ID if it exists; otherwise, select the Xcode-managed wildcard App ID.

A green circle followed by “Enabled” appears in the Development column of the services your app can use.



You can also edit App ID settings using Member Center, but any changes you make in Member Center won't configure app services in your Xcode project. Use the Capabilities pane in Xcode to fully configure app services. Push notifications is the only service, shown in Member Center, that Xcode doesn't configure for you.

Learn More About App Services

Refer to this table for more information on key app services that appear in the Capabilities pane in Xcode or require additional configuration using Xcode, Member Center, or iTunes Connect.

To learn about	Read
App Sandbox Provides the last line of defense against stolen, corrupted, or deleted user data if malicious code exploits your Mac app.	Configuring App Sandbox for Mac Apps <i>App Sandbox Design Guide</i> <i>Entitlement Key Reference</i>
iCloud Storage Key-Value and Document Storage Allows sharing of user's data among multiple instances of your app running on different iOS devices and Mac computers.	Adding iCloud Support in <i>App Distribution Guide</i> <i>iCloud Design Guide</i>
CloudKit Provides a database-like model for storing both private and public data where public data is owned by the app and can be shared among all users of the app.	<i>CloudKit Quick Start</i> <i>CloudKit Framework Reference</i>
Game Center Allows players to connect devices to Apple's social gaming network and to exchange information.	<i>Game Center Programming Guide</i> <i>Game Center Configuration Guide for iTunes Connect</i>
Passbook For iOS apps, presents digital representations of information—such as a coupon, ticket for a show, or boarding pass—that allow users to redeem a real-world product or service.	Configuring Passbook for iOS Apps
In-App Purchase Embeds a store directly into your app by accessing the store and securely processing payments from the user.	<i>In-App Purchase Programming Guide</i> <i>In-App Purchase Configuration Guide for iTunes Connect</i>
Maps Makes your app's routing information available to Maps and other apps that need point-to-point directions.	Configuring Maps in <i>App Distribution Guide</i>
Background Modes Enabling background modes allows your app to continue running in the background.	Configuring Background Modes for iOS Apps in <i>App Distribution Guide</i>
Keychain Sharing Allows your app to share passwords in the keychain with other apps developed by your team.	Configuring Keychain Sharing in <i>App Distribution Guide</i>

To learn about	Read
Inter-App Audio Allows your app to export audio that other apps can use.	Inter-App Audio Examples in <i>Inter-App Audio Examples</i>
Data Protection For iOS apps, adds a level of security to files stored on disk by your iOS app.	Protecting Data Using On-Disk Encryption in <i>App Programming Guide for iOS</i>
HomeKit For iOS apps, provides integration between accessories that support Apple's Home Automation Protocol and iOS devices.	<i>HomeKit Framework Reference</i>
HealthKit For iOS apps, provides a structure that apps can use to share health and fitness data.	<i>HealthKit Framework Reference</i>
Push Notifications (Apple Push Notification service) Allows an app that isn't running in the foreground to notify the user that it has information for the user.	Configuring Push Notifications <i>Local and Remote Notification Programming Guide</i>
Newsstand For iOS apps, enables an app to organize a user's magazine and newspaper app subscriptions into a folder.	Newsstand for Developers Newsstand Icons in <i>iOS Human Interface Guidelines</i>

Recap

In this chapter, you learned how to enable some app services that Apple provides for apps submitted to the store. By examining your team provisioning profiles and App IDs in Member Center, you gained a deeper understanding of how provisioning works. You also learned where to find more information on these app services.

Next Steps

In this document you learned the early steps you perform to provision and code sign your app during development. Your next step is to fully configure and write code for the app services that you enabled in [Enabling App Services](#) (page 44). You might also explore additional and alternative workflows depending on the size of your team and choice of platform. When you're ready to distribute your app to testers or customers, read *App Distribution Guide*.

Where to Go from Here

Refer to these chapters in *App Distribution Guide* for additional tasks that you perform throughout the lifetime of your app.

To learn how to	Read
For a company, add team members, assign roles, approve their certificates, and register their devices	Managing Your Team in <i>App Distribution Guide</i>
Add iTunes Connect users to your account	Adding iTunes Connect Users in <i>App Distribution Guide</i>
Perform final configuration steps before distributing your app	Configuring Your Xcode Project for Distribution in <i>App Distribution Guide</i>
For iOS apps, distribute your app for beta testing	Beta Testing Your iOS App in <i>App Distribution Guide</i>
Upload your app to iTunes Connect for approval	Submitting Your App in <i>App Distribution Guide</i>
For Mac applications, sign your application using a Developer ID certificate	Distributing Applications Outside the Mac App Store in <i>App Distribution Guide</i>

Before you submit your app to the store, read the appropriate human interface guidelines for your platform and, most important, read the guidelines for submitting your app to the store.

To learn about	iOS	Mac
The user interface guidelines	<i>iOS Human Interface Guidelines</i> App Store Review Guidelines for iOS Apps	<i>OS X Human Interface Guidelines</i> <i>App Store Review Guidelines for Mac Apps</i>

Document Revision History

This table describes the changes to *App Distribution Quick Start*.

Date	Notes
2014-10-20	Applied minor edits.
2014-10-16	Updated for Xcode 6.1.
2014-09-17	Updated for Xcode 6.0.
2014-03-10	New document that teaches you how to create signing identities and provisioning profiles when developing 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 technologies and run on designated devices during development. This profile consists of a name, multiple development certificates, multiple devices, and an App ID.

device Used to refer to a Mac computer—or to an iPad, iPhone, or iPod—when no further distinction between them is needed.

device ID A way of uniquely identifying an iOS or Mac device.

distribution certificate A type of signing certificate used to distribute an app and allow it to launch on a device without the assistance of Xcode. A distribution certificate identifies a team, not a team member.

distribution provisioning profile A type of provisioning profile that authorizes an app to run on devices without the assistance of Xcode and allows them to use certain technologies. A distribution provisioning profile is used to submit an app to the App Store or Mac App Store. Mac has one type of distribution provisioning profile, and iOS has two.

entitlement A single right granted to a particular app, tool, or other executable that gives it additional permissions beyond what it would ordinarily have.

explicit App ID An App ID that matches a single bundle ID, in contrast to a wildcard App ID, which can match one or more bundle IDs.

Game Center Apple's social gaming network that allows players to connect to the service and exchange information with other players.

Gatekeeper The OS X feature that enables users to choose to disallow the launching of applications that 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 Store Package A type of OS X file that, when double-clicked, installs an app in iTunes, where it can be synced to an iOS device.

iOS Dev Center An Apple developer center that provides all the resources needed to develop iOS 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 a team. The Team ID is used as the prefix for an [App ID](#).

team member A person on a development team who has the fewest privileges. A team member can sign apps during development after that request is approved by a team admin. Compare [team agent](#); [team admin](#).

team provisioning profile The development provisioning profile that Xcode creates and manages for you. The team provisioning profile contains all of a team's development certificates, its registered devices, and the wildcard App ID, which Xcode also creates.

wildcard App ID An App ID that matches one or more bundle IDs used by a development team. Compare [explicit App ID](#).

Xcode iOS Wildcard App ID The [wildcard App ID](#) that Xcode manages for iOS developers.

Xcode Mac Wildcard App ID The [wildcard App ID](#) that Xcode manages for Mac developers.



Apple Inc.
Copyright © 2014 Apple Inc.
All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, electronic, photocopying, recording, or otherwise, without prior written permission of Apple Inc., with the following exceptions: Any person is hereby authorized to store documentation on a single computer or device for personal use only and to print copies of documentation for personal use provided that the documentation contains Apple's copyright notice.

No licenses, express or implied, are granted with respect to any of the technology described in this document. Apple retains all intellectual property rights associated with the technology described in this document. This document is intended to assist application developers to develop applications only for Apple-branded products.

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Apple, the Apple logo, Cocoa, iPad, iPhone, iPod, iPod touch, iTunes, Keychain, Mac, OS X, Passbook, Safari, Sand, and Xcode are trademarks of Apple Inc., registered in the U.S. and other countries.

.Mac, iCloud, and iTunes Store are service marks of Apple Inc., registered in the U.S. and other countries.

App Store and Mac App Store are service marks of Apple Inc.

iOS is a trademark or registered trademark of Cisco in the U.S. and other countries and is used under license.

APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS DOCUMENT IS PROVIDED "AS IS," AND YOU, THE READER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.

IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT, ERROR OR INACCURACY IN THIS DOCUMENT, even if advised of the possibility of such damages.

Some jurisdictions do not allow the exclusion of implied warranties or liability, so the above exclusion may not apply to you.