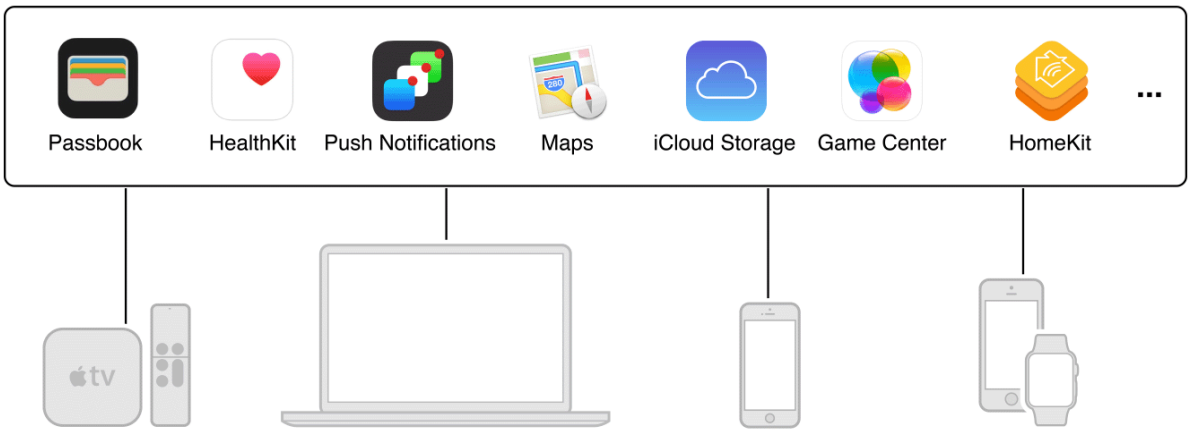
Adding Capabilities

Certain app services—such as Game Center and In-App Purchase—are available only to members of the Apple Developer Program who distribute their apps through the store. These services require additional configuration in your Xcode project, your developer account, and sometimes iTunes Connect. App services that don't require iTunes Connect configuration are also available to Apple 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 your developer account and iTunes Connect. Push notifications require additional setup in your developer account.

This chapter describes all the steps that you perform to access app services from your app. For the list of the capabilities available for your type of app, see Supported Capabilities.

About Entitlements

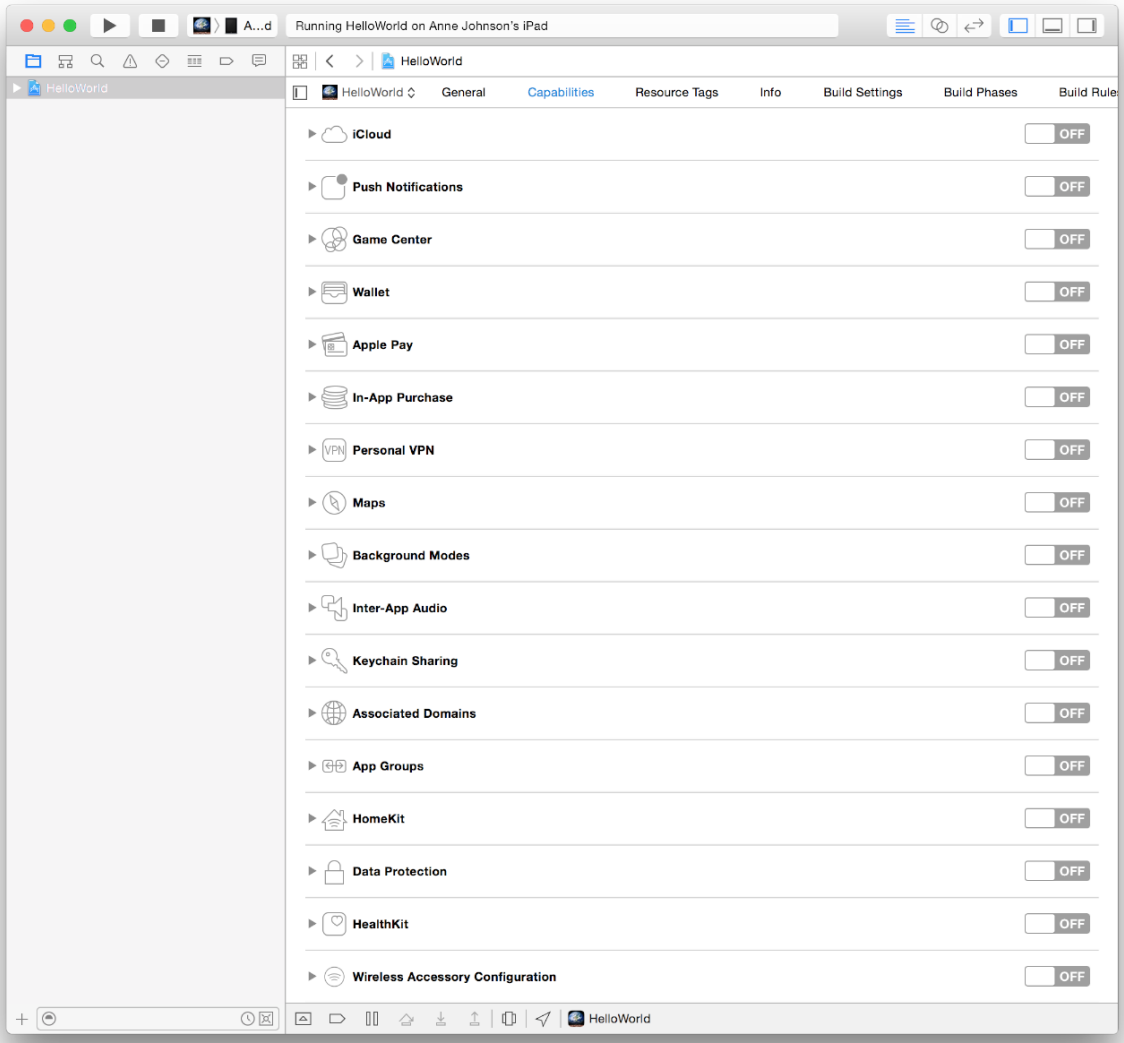
An **entitlement** is a single right granted to a particular app, tool, or other executable that gives it additional permissions above and beyond what it would ordinarily have. The term *entitlement* is most commonly used in the context of a sandbox, and to a lesser degree for an App ID. Regardless of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

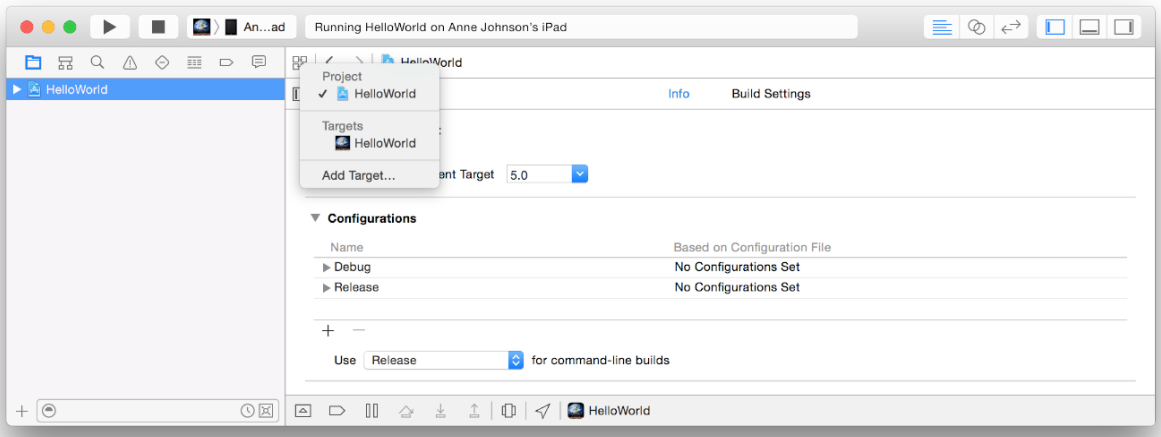
All of the options discussed in this chapter are located in the Capabilities pane in the project editor for your target.

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, and for iOS, tVOS, or watchOS apps, connect a 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. Otherwise, read Troubleshooting for how to resolve issues as they occur.

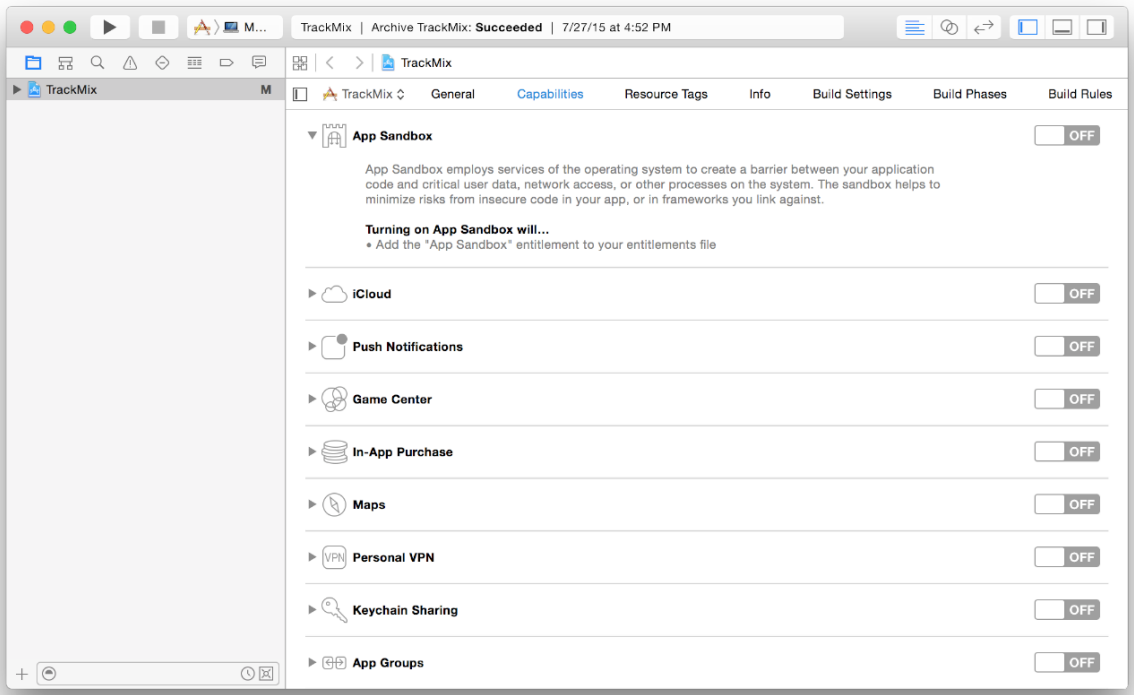
Configuring App Sandbox (Mac)

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.

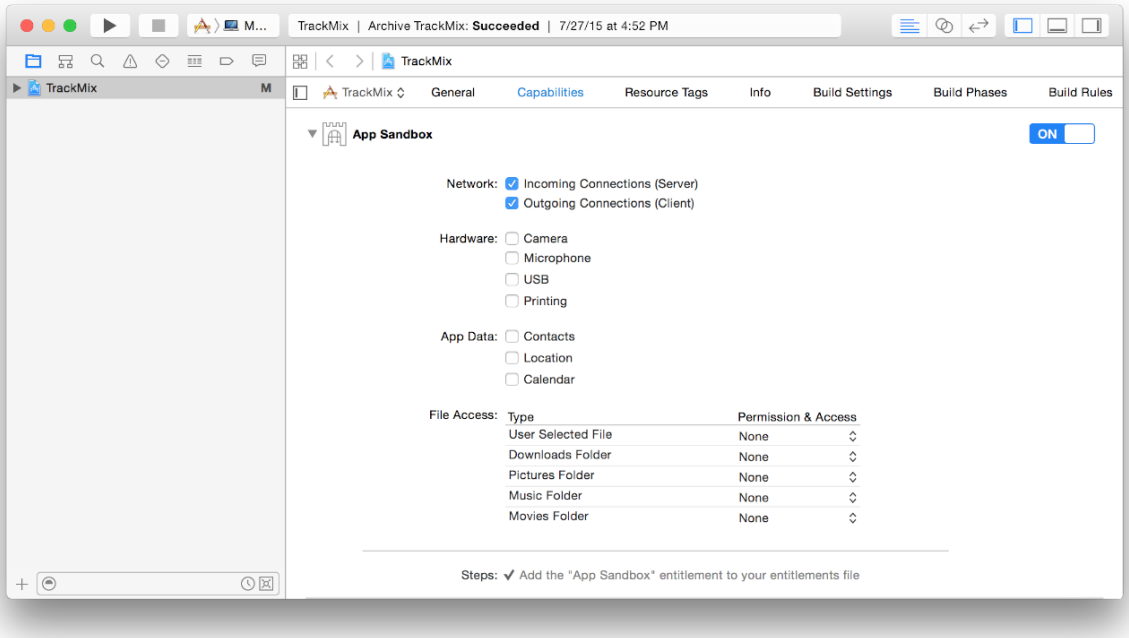
To configure App Sandbox

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

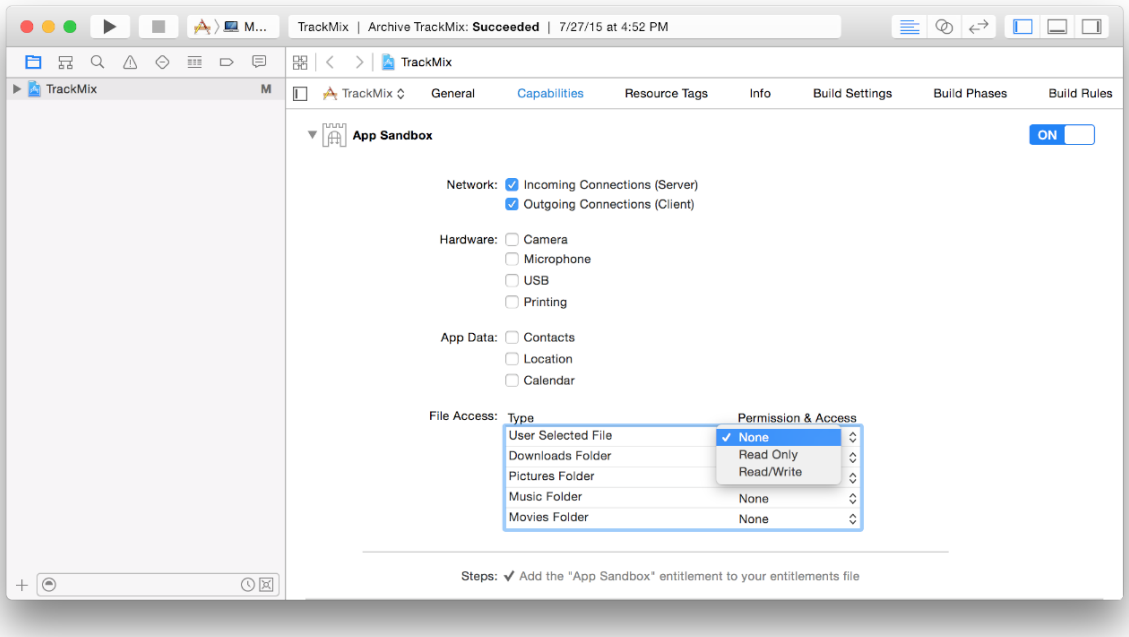


   Xcode adds an entitlements file to your project and automatically enters default values for some entitlements. Xcode also enables the App Sandbox entitlement.

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.

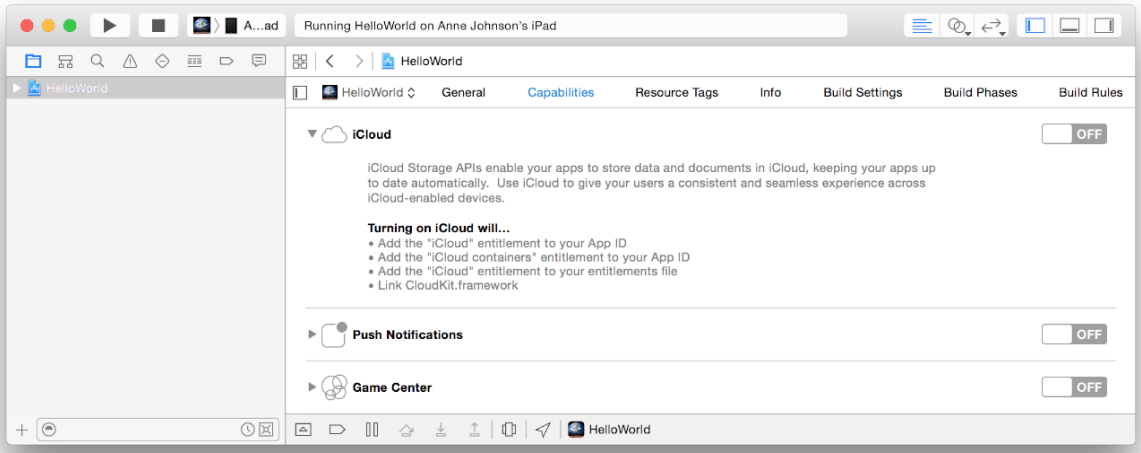
Adding iCloud Support

iCloud storage allows you to share user or app data among multiple instances of your app running on different devices. 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.

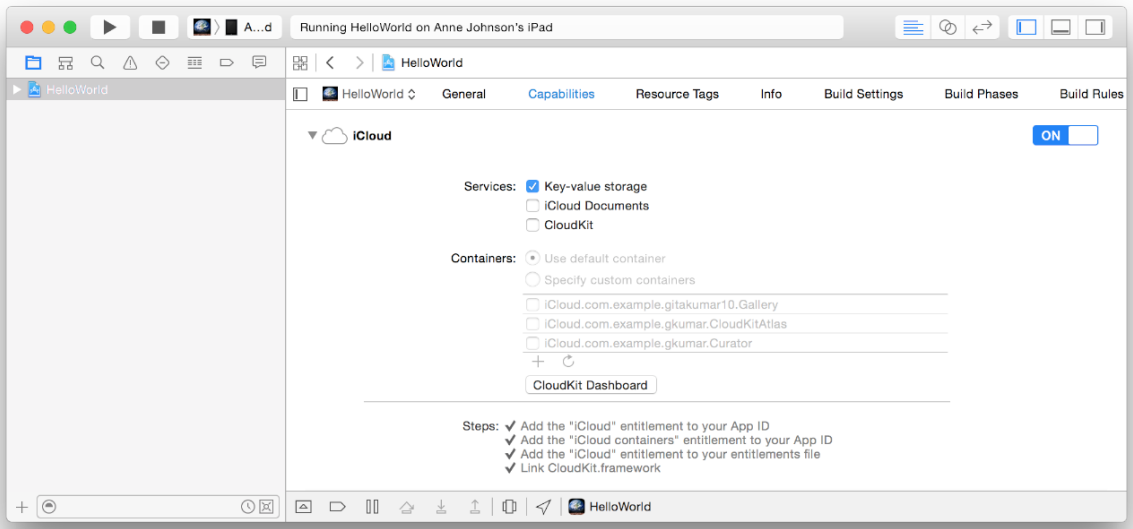
Enabling iCloud

Before you choose and configure iCloud services, you enable iCloud in Xcode. To enable iCloud, click the switch in the iCloud section. 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*.

> **Important:** Only a team agent or admin can configure Document Storage.

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.

> **Important:** Only a team agent or admin can configure CloudKit.

Specifying Custom Containers

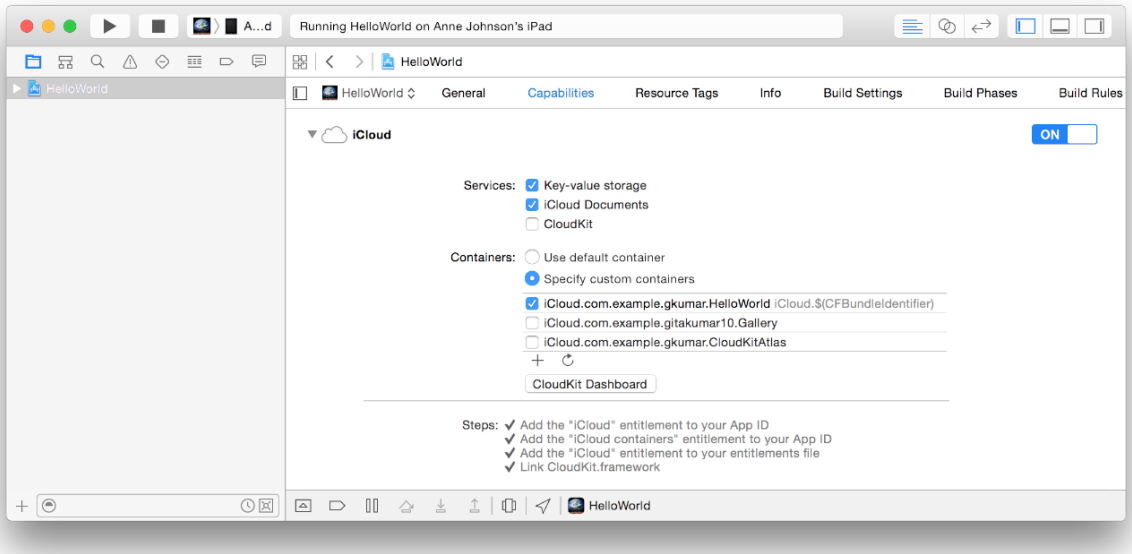
For document storage and CloudKit, the default container ID is `iCloud.$(CFBundleIdentifier)` which matches the explicit App ID. Optionally, add one or more custom containers and share them between apps. You can select an existing container ID used by another app or create a new one.

> **Important:** Only a team agent or admin can specify custom containers.

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.



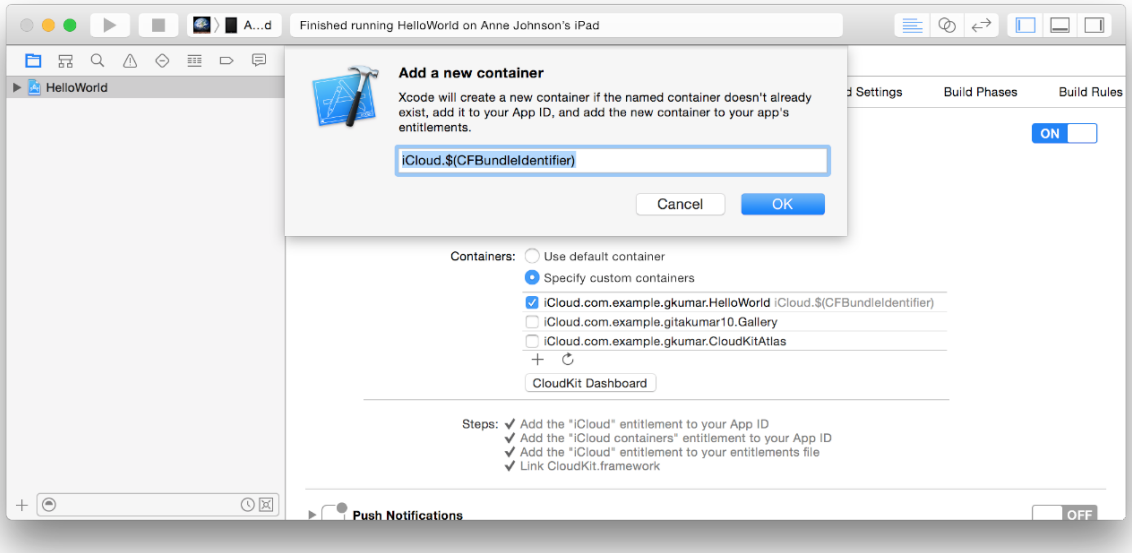
If an existing container ID is not sufficient, create another container ID for the app.

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 your developer account.

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

Configuring Push Notifications

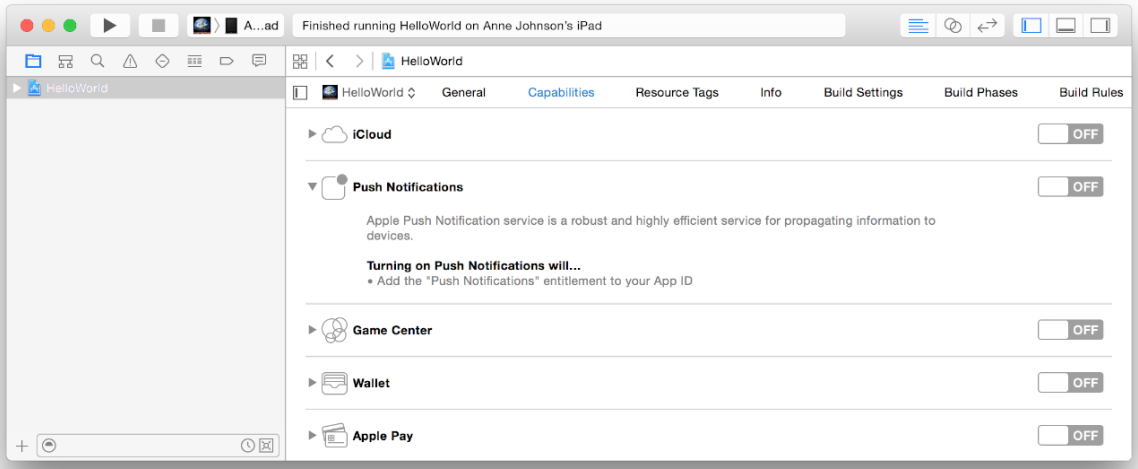
Apple Push Notification service (APNs) allows an app that isn't running in the foreground to notify the user that it has information for the user. Use Xcode to enable push notifications. If necessary, Xcode creates an explicit App ID with the APNs entitlement enabled and a team provisioning profile using the explicit App ID. You then generate and download a corresponding client SSL certificate—this step fully enables push notifications—and later, install the client SSL certificate and key on your server.

Enabling Push Notifications

First enable push notifications in your Xcode project.

To enable push notifications

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



In your developer account, the Push Notifications service will appear as `Configurable` (not `Enabled`) until you create a client SSL certificate.

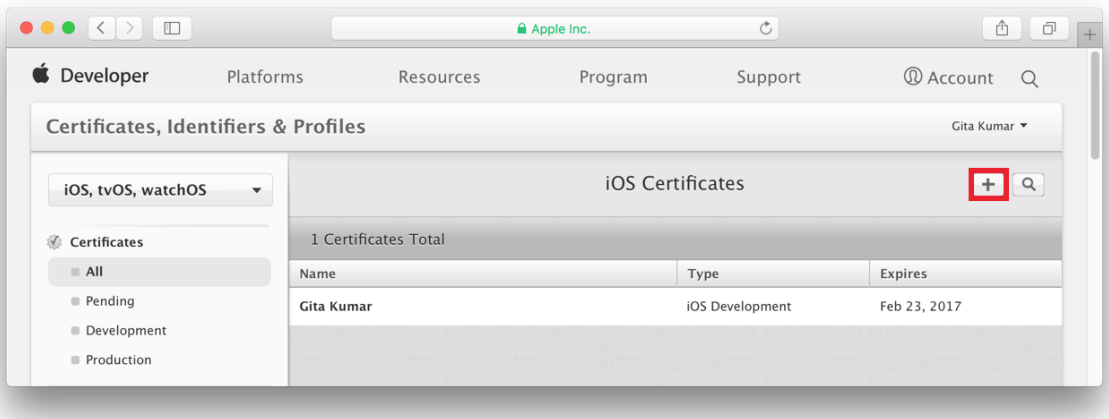
Creating a Universal Push Notification Client SSL Certificate

You use your developer account to generate a push notification *client SSL certificate* that allows your notification server to connect to the APNs. Each App ID is required to have its own client SSL certificate. The client SSL certificate that is generated is a universal certificate that allows your app to connect to both the development and production environments.

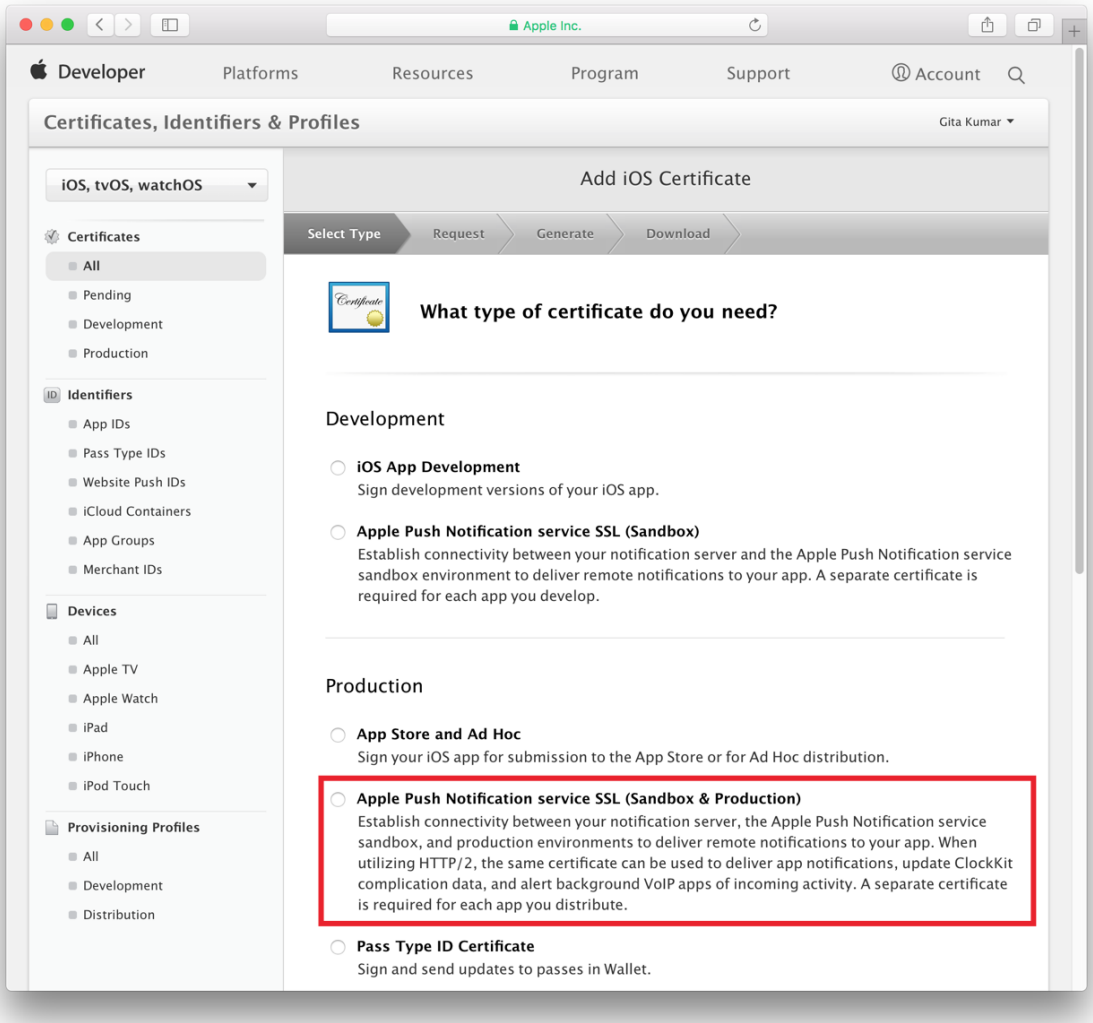
> **Important:** Only a team agent or admin can generate Apple Push Notification service SSL certificates.

To generate a universal client SSL certificate

1. Go to Certificates, Identifiers & Profiles and for Mac apps, choose OS X from the pop-up menu on the left.
2. Under Certificates, select All.
3. Click the Add button (+) in the upper-right corner.



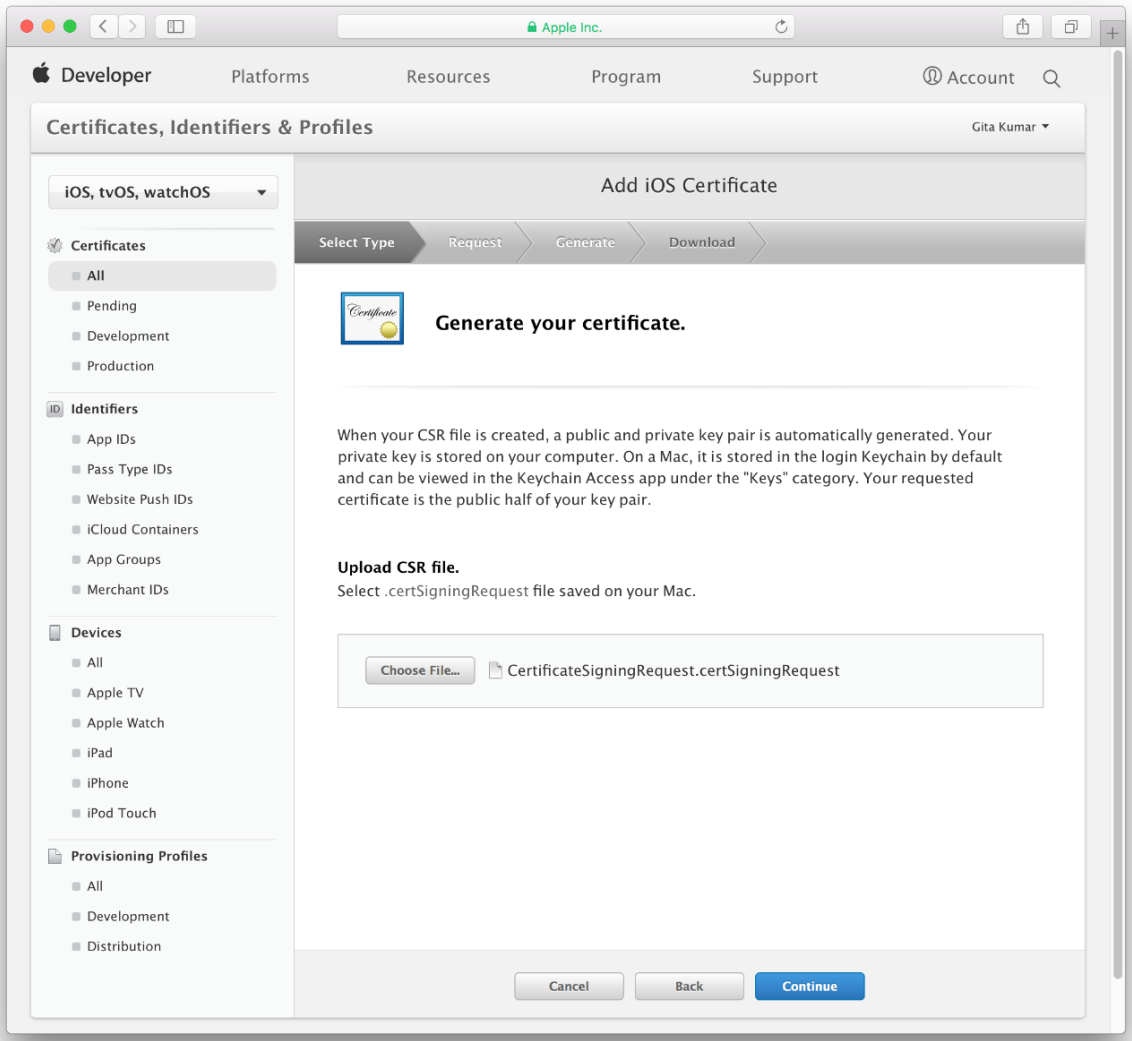
4. Under Production, select the “Apple Push Notification service SSL (Sandbox & Production)” checkbox, and click Continue.



5. Choose an App ID from the App ID pop-up menu, and click Continue.

   Choose the explicit App ID that matches your bundle ID.

6. Follow the instructions on the next webpage 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 Continue.



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

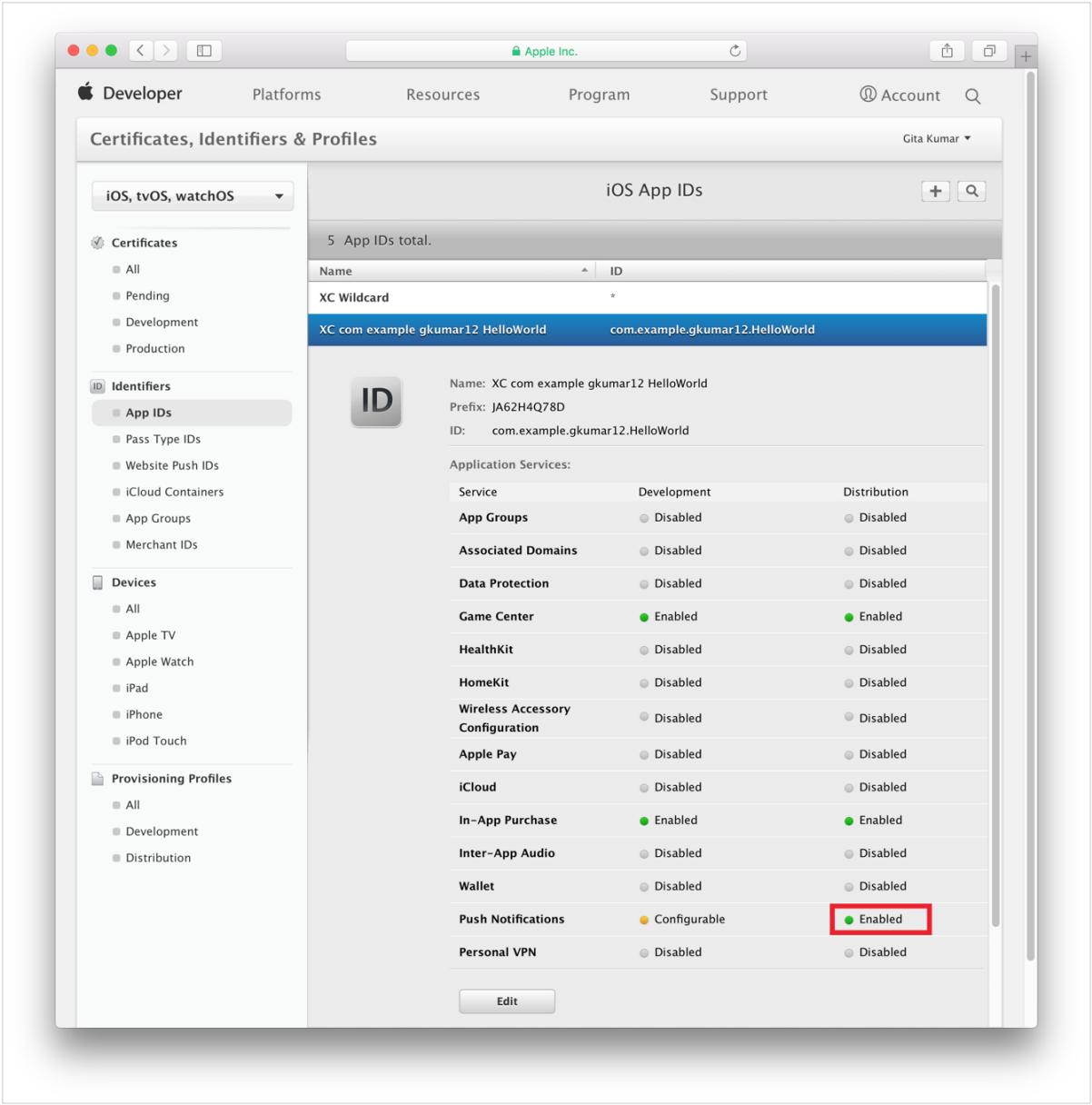
Verifying Your Steps

In your developer account, the status of the Push Notifications service for the specified environment (Development or Distribution) changes from `Configurable` to `Enabled`.

To verify the App ID settings

1. Go to Certificates, Identifiers & Profiles and for Mac apps, choose OS X from the pop-up menu on the left.
2. Under Identifiers, select App IDs.
3. Select the explicit App ID that matches the bundle ID.

   A green circle followed by “Enabled” appears in the Push Notifications row and the Distribution column.



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*.

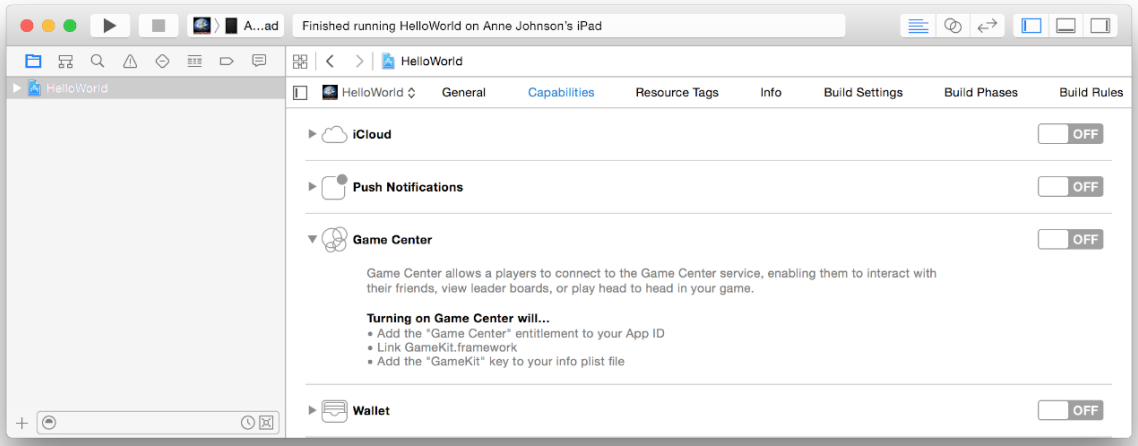
Enabling Game Center (iOS, tvOS, Mac)

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.



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).

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.

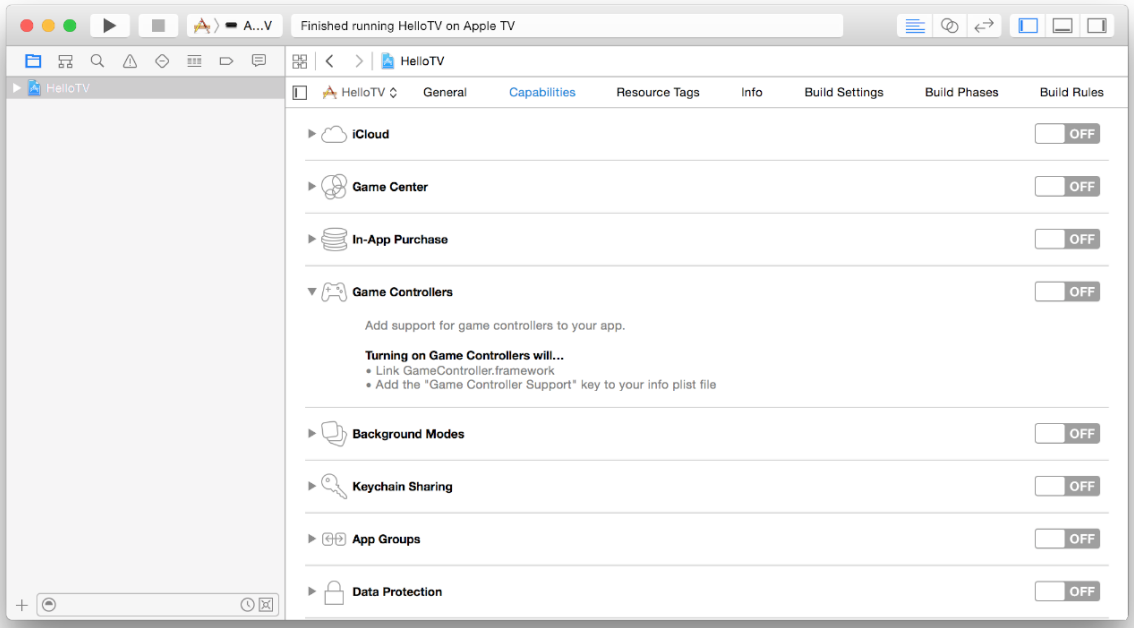
Enabling Game Controllers (tvOS)

Game controllers provide physical controls to trigger actions in your game.

To use the Game Controller framework, enable Game Controllers in Xcode.

To enable Game Controllers

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



For how to configure game controllers and write Game Controller framework code, read *Game Controller Programming Guide*.

Configuring Wallet (iOS, WatchKit Extension)

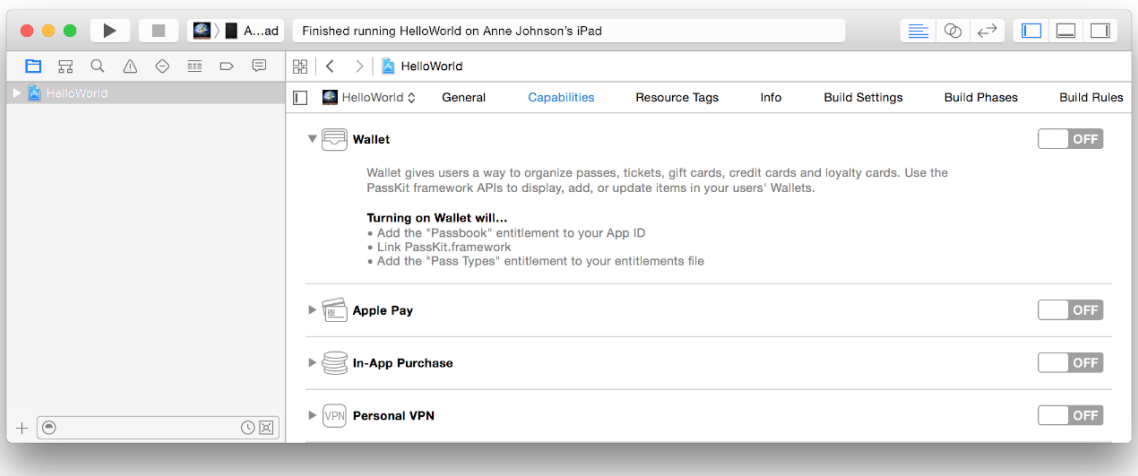
Wallet presents digital representations of information—such as a coupon, ticket for a show, or boarding pass—that allow users to redeem a real-world product or service. You can use Wallet in several ways:

- 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 *Wallet Developer Guide*.
- To let users add passes to Wallet from your app, use the PassKit framework. You don't need to set Wallet entitlements to do this.
- To access the user's passes in your app, follow the steps below.

First, you enable Wallet in your Xcode project.

To enable Wallet

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

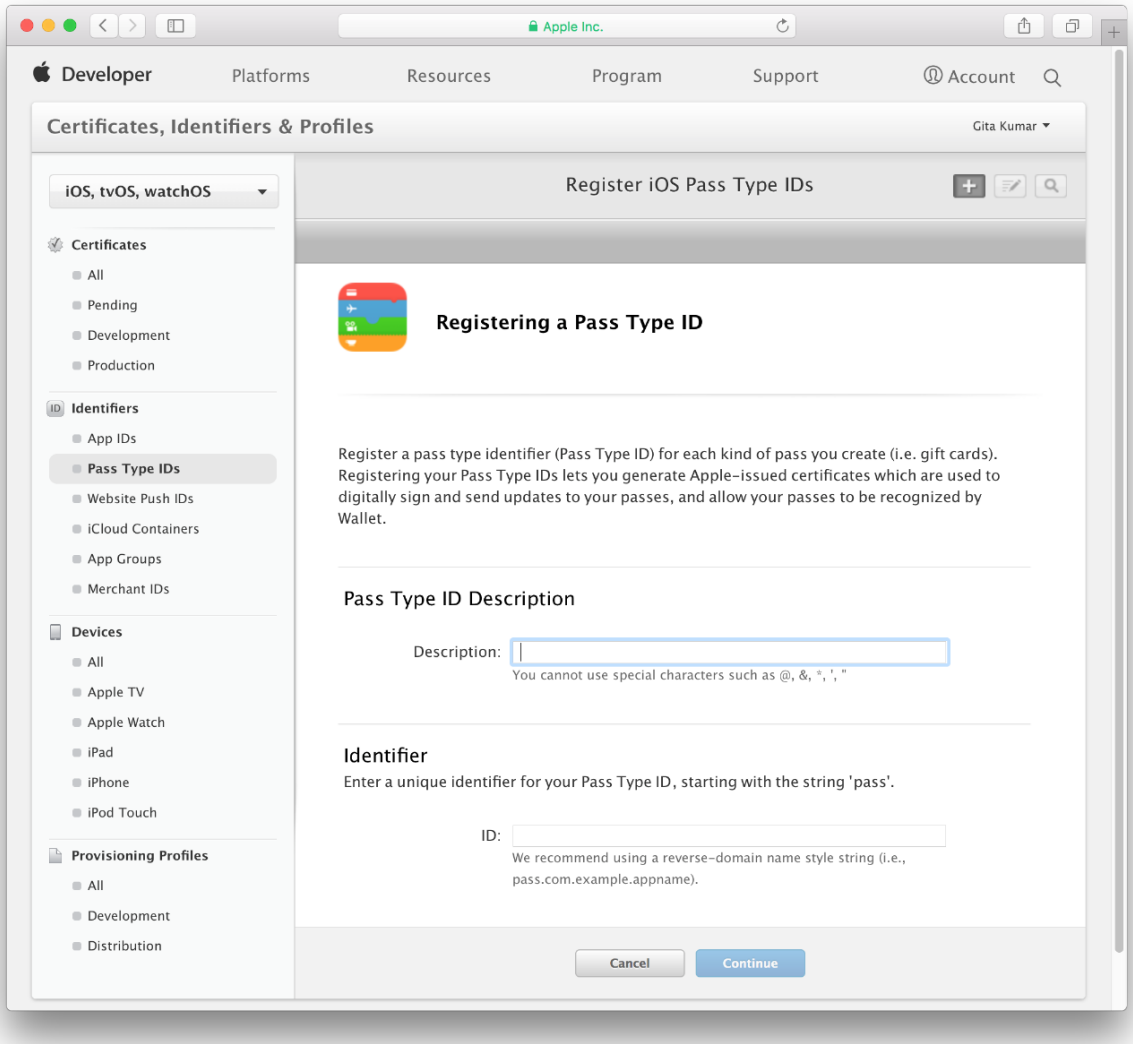


Xcode automatically provisions your app to use Wallet and adds the PassKit framework to your project.

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. Go to Certificates, Identifiers & Profiles and for Mac apps, choose OS X from the pop-up menu on the left.
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.



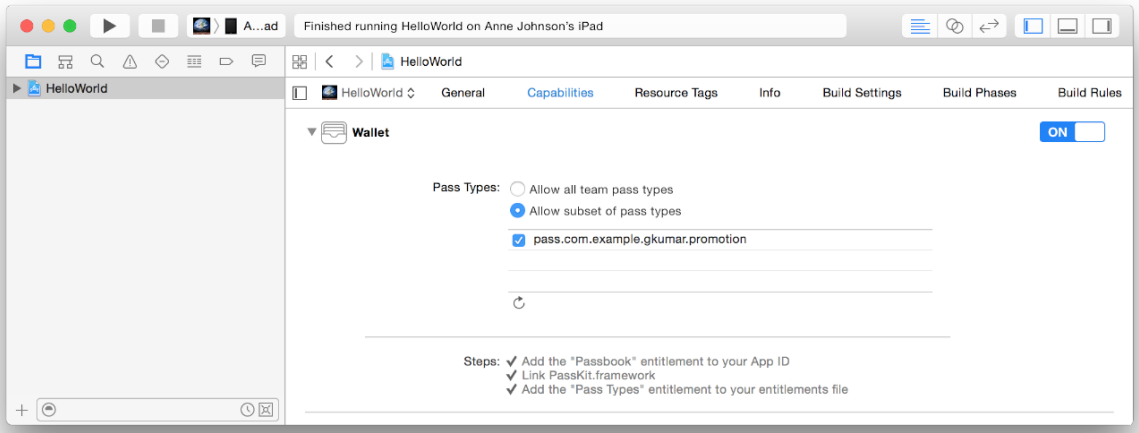
5. Review the settings, and click Register.
6. Click Done.

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 Wallet disclosure triangle.
2. Select "Allow subset of pass types."

   If there are no pass type identifiers in your developer account, 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.

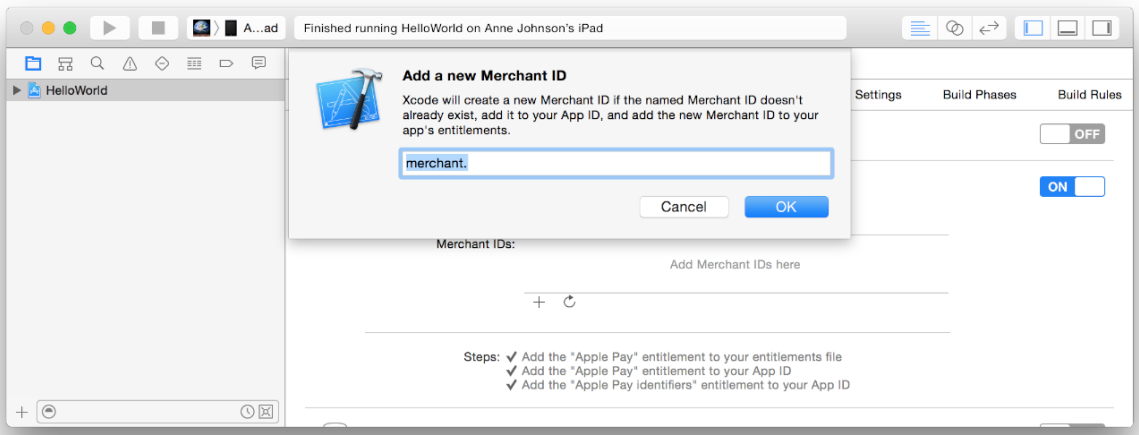
To use a pass type identifier in your app, read Setting the Pass Type Identifier and Team ID in *Wallet Developer Guide*.

Configuring Apple Pay (iOS, WatchKit Extension)

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



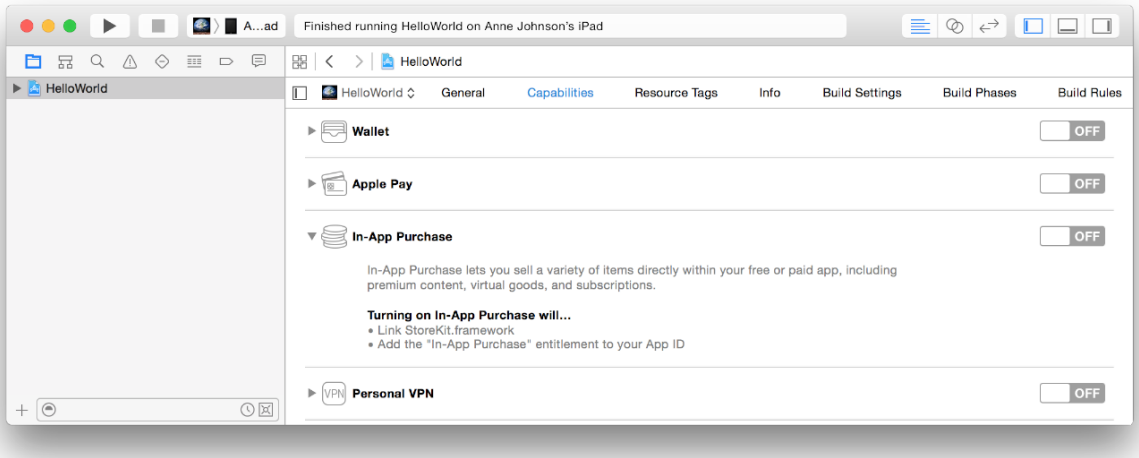
To learn more about Apple Pay, read *Apple Pay Programming Guide* and *PassKit Framework Reference*.

Enabling In-App Purchase (iOS, tvOS, Mac)

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.



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

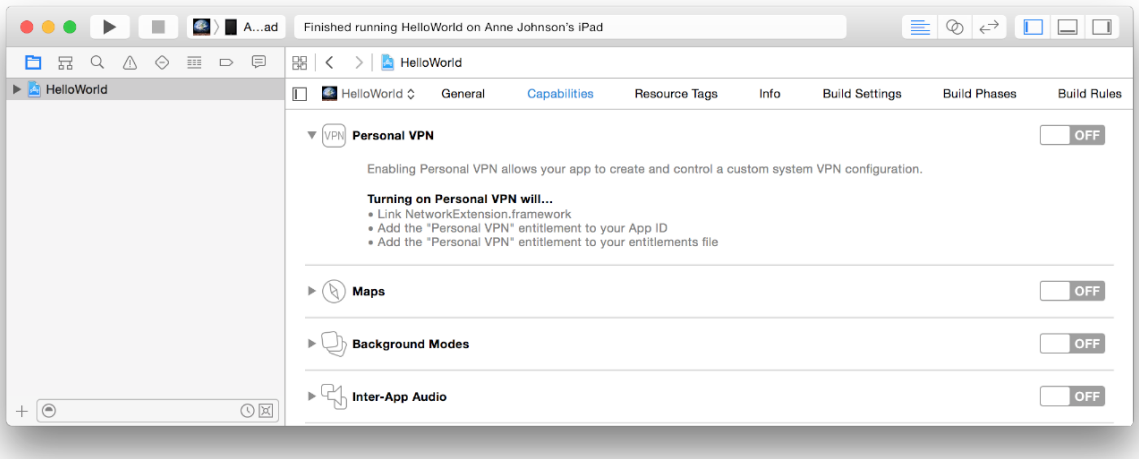
For how to write your In-App Purchase code, read *In-App Purchase Programming Guide*. To create an app record and enter the explicit App ID in iTunes Connect, read Adding New Apps in *iTunes Connect Developer Guide*. To create and upload in-app purchases, read *In-App Purchase Configuration Guide for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

Enabling Personal VPN (iOS, Mac)

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.



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, read *Network Extension Framework Reference*.

Configuring Maps (iOS, WatchKit Extension, Mac)

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. You use Xcode to enable the Maps service. For iOS routing apps, you use iTunes Connect to upload a geographic coverage file.

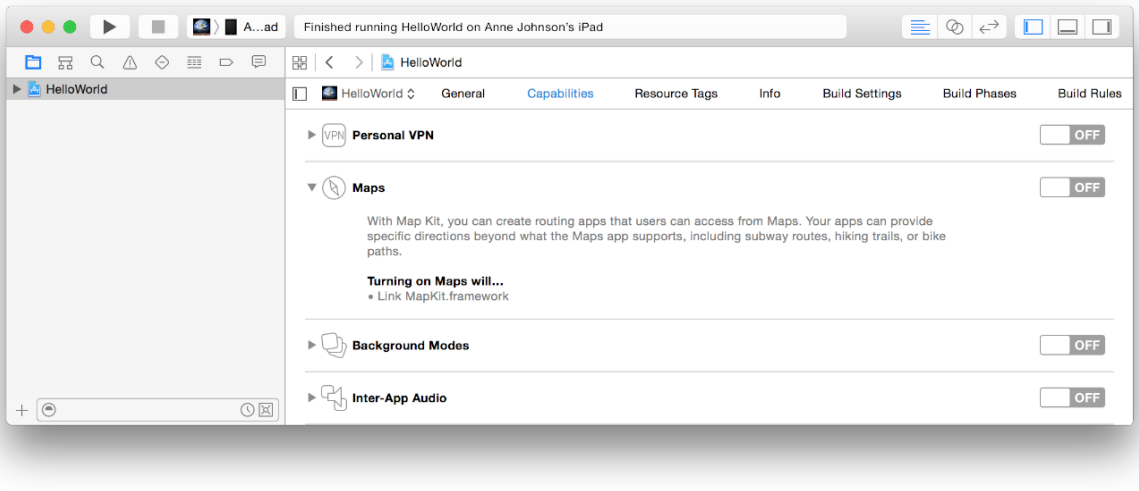
For how to write your MapKit framework code, read *Location and Maps Programming Guide*.

Enabling Maps in Xcode

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

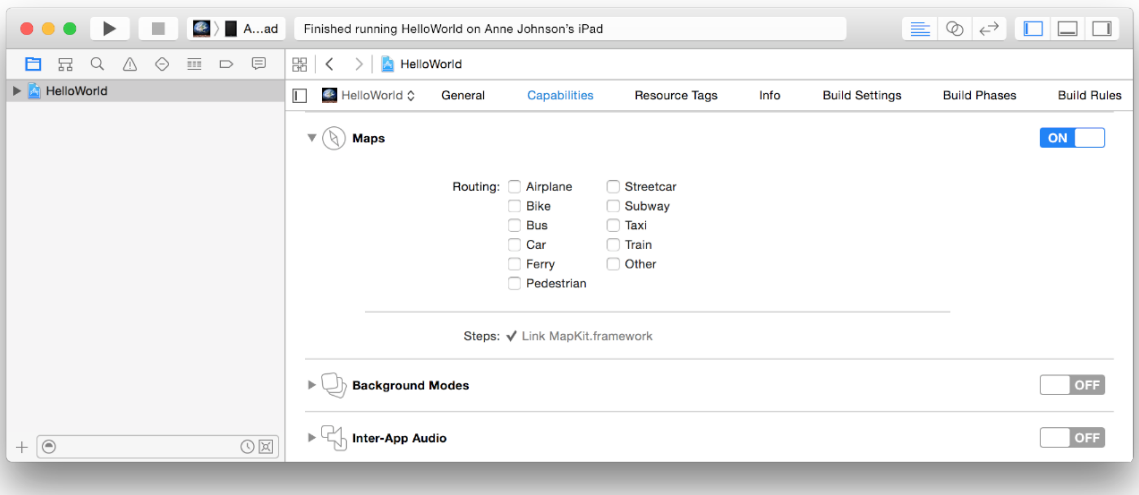
To enable Maps and select modes

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



2. 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.



Configuring a Routing App (iOS, WatchKit Extension)

You perform additional steps to configure an app that provides point-to-point directions for other apps. Before continuing, review the tasks that you perform to configure a 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 build 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 build, as described in Uploading the Geographic Coverage File to iTunes Connect.

Submitting a Build to the Store

To upload a build to iTunes Connect, follow the steps in Submitting Your App to the Store.

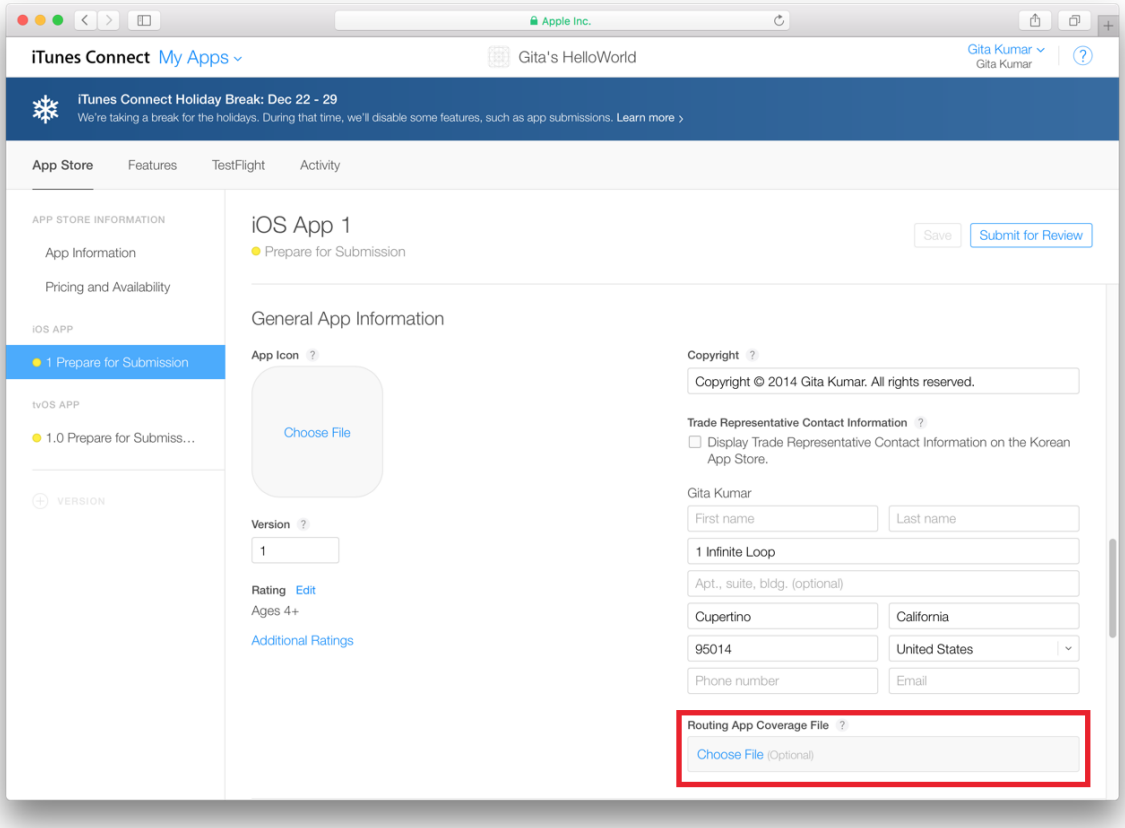
Uploading the Geographic Coverage File to iTunes Connect

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.
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.

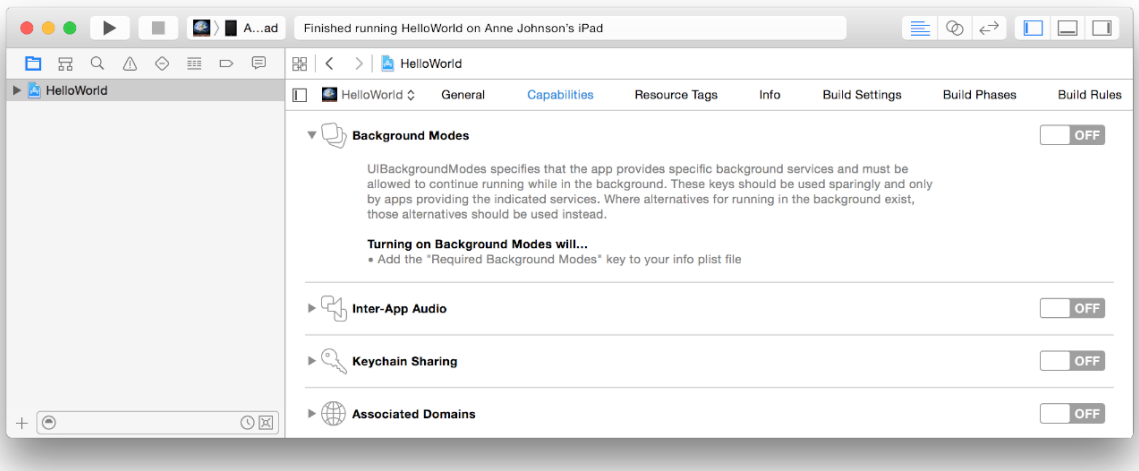
   An error message appears if the file isn't formatted correctly or has the wrong file extension.

Configuring Background Modes (iOS, tvOS, WatchKit App)

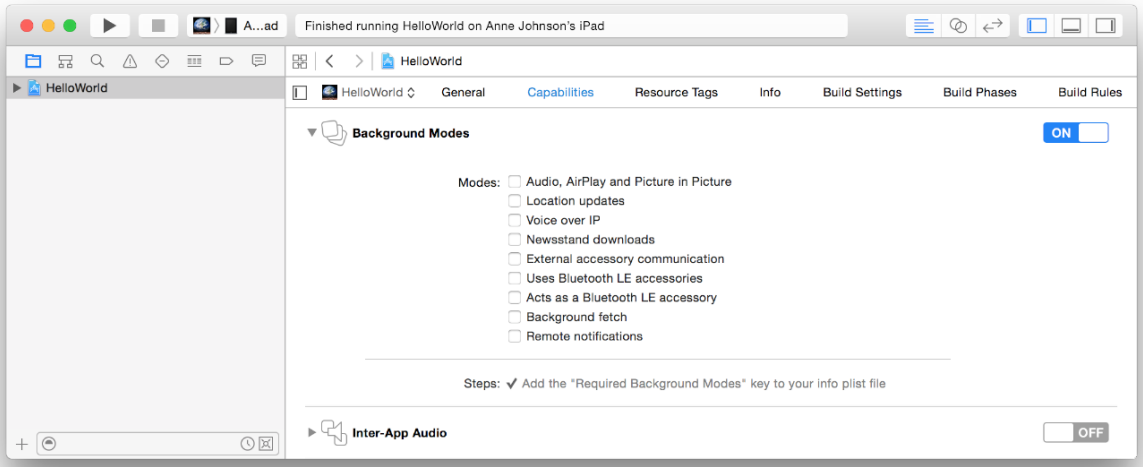
Enabling background modes allows your app to continue running in the background.

To enable background modes

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



2. Optionally, select the supported modes from the checkboxes below.



Xcode adds the background modes to the information property list.

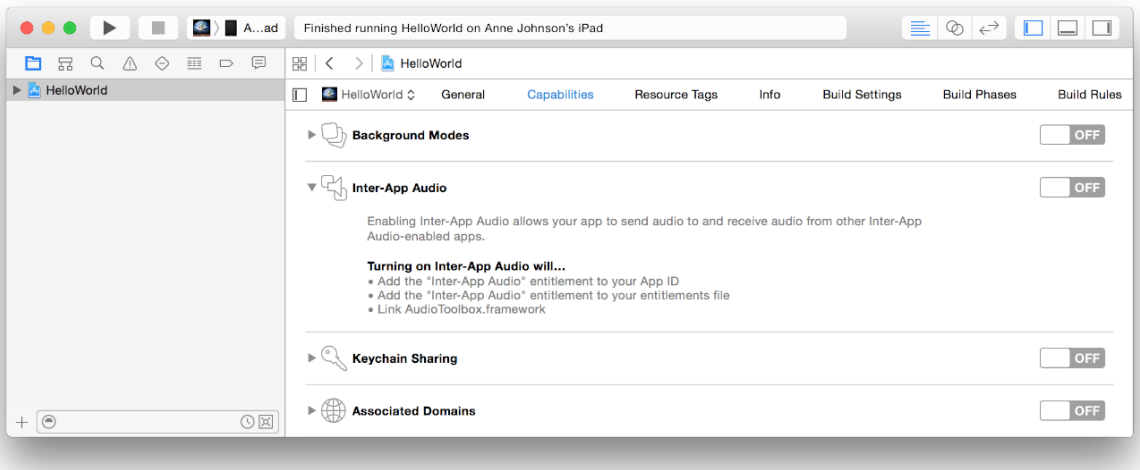
For guidance on selecting background modes, read Background Execution in *App Programming Guide for iOS*.

Enabling Inter-App Audio (iOS)

Inter-app audio allows your 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.



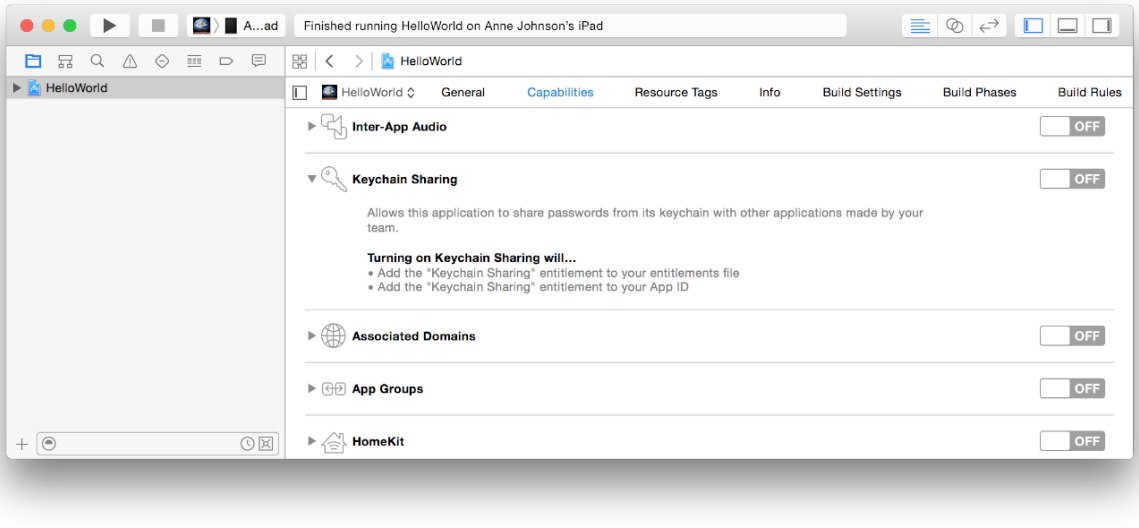
Xcode automatically provisions your app to use inter-app audio and adds the Audio Toolbox framework to your project. For how to write Audio Toolbox framework code, read *Audio Toolbox Framework Reference*.

Configuring 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.

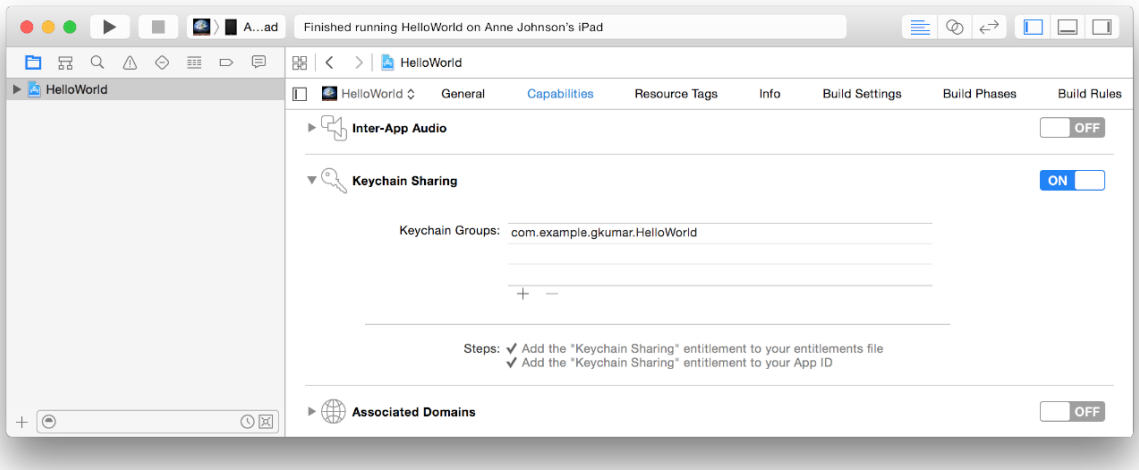


Xcode adds the `keychain-access-groups` key to the entitlements file.

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.



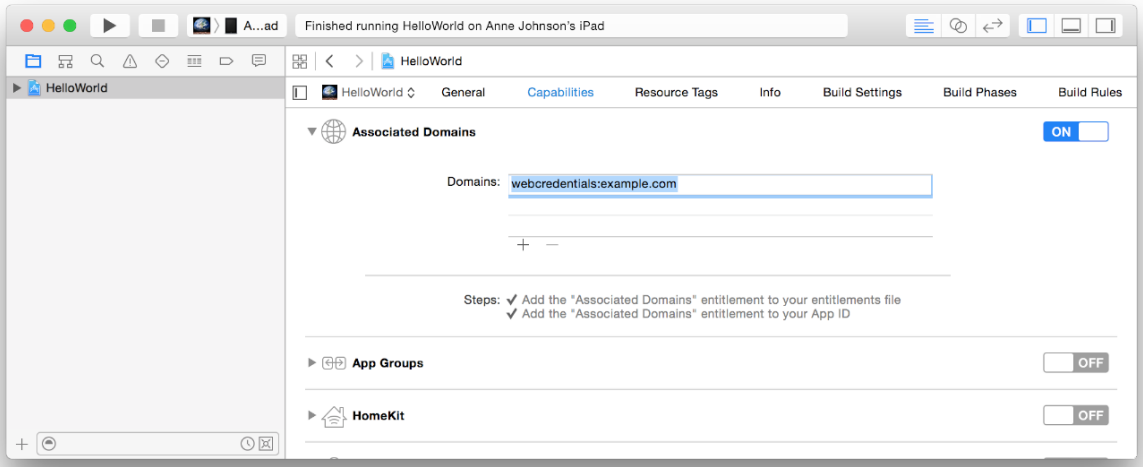
To delete a keychain access group, select it in the Keychain Groups area and click the Delete button (–).

Configuring Associated Domains (iOS, WatchKit Extension)

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

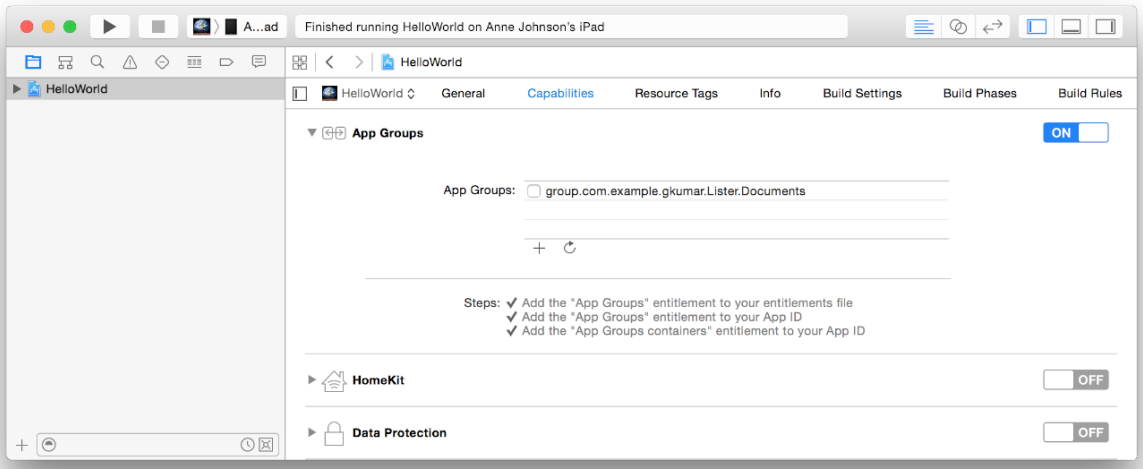
1. If necessary, click the Refresh button below the table to download container IDs from your developer account.
2. In the row of the container ID, select the checkbox to use the container and deselect the checkbox to not use the container.

   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.



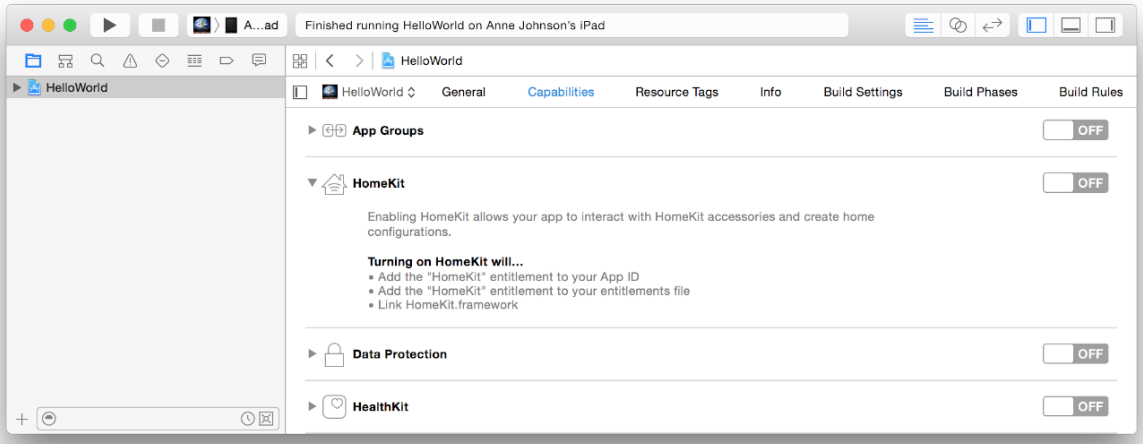
2. In the dialog that appears, enter a container ID in the text field and click OK.

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

Adding HomeKit (iOS, WatchKit Extension)

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.

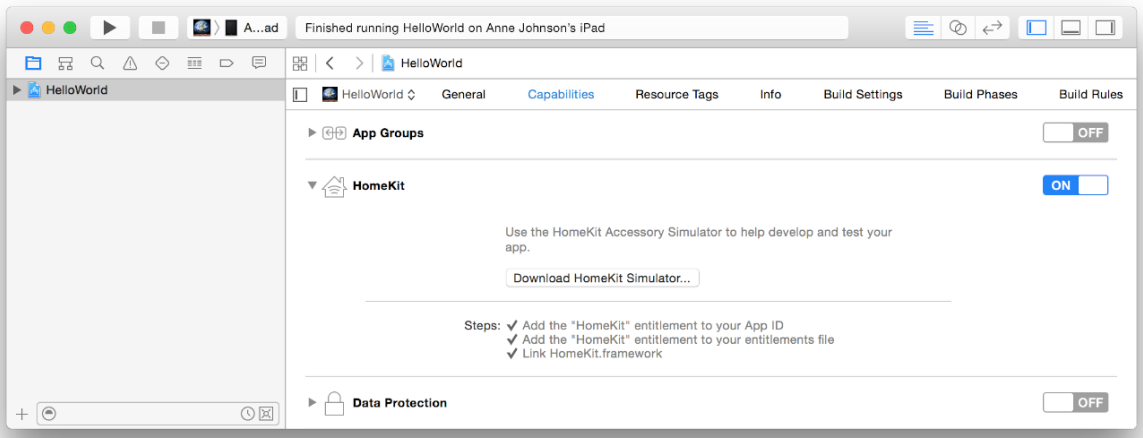


HomeKit Accessory Simulator is not distributed with Xcode.

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.

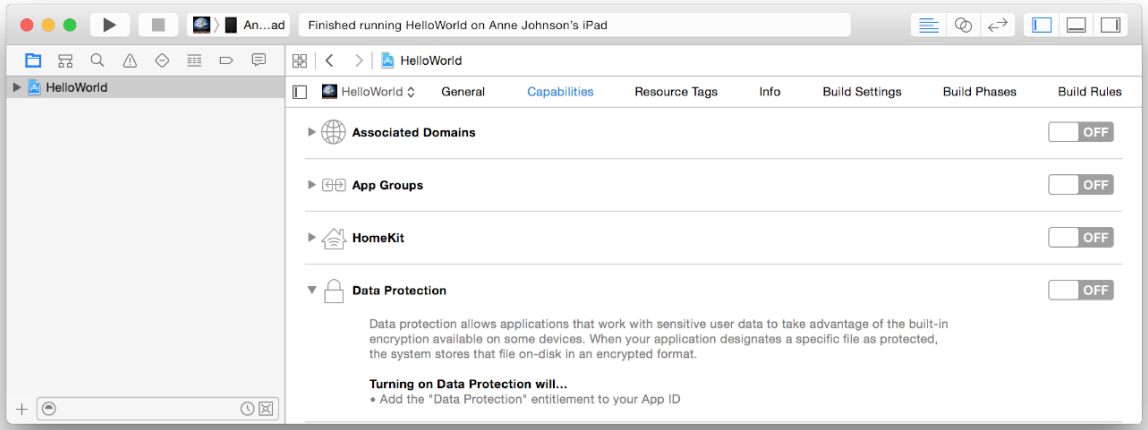
For information about the HomeKit framework, see *HomeKit Framework Reference*.

Enabling Data Protection (iOS, WatchKit Extension, tvOS)

Data protection adds a level of security to files stored on disk by your app in the app's container. 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

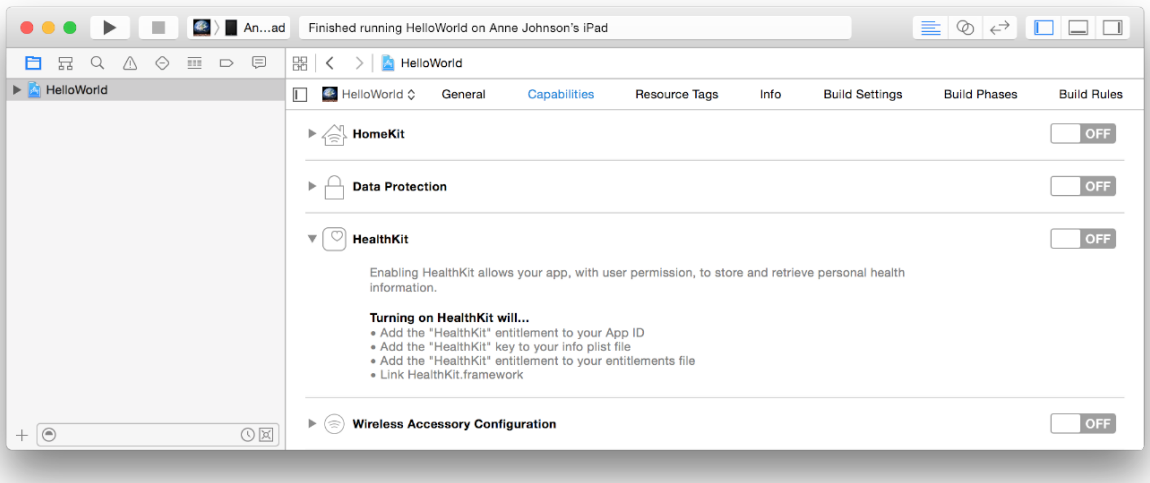
1. In the Capabilities pane, if Data Protection isn't enabled, click the switch in the Data Protection section.



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*. For files stored in shared containers (described in Configuring App Groups), set the level of protection programmatically.

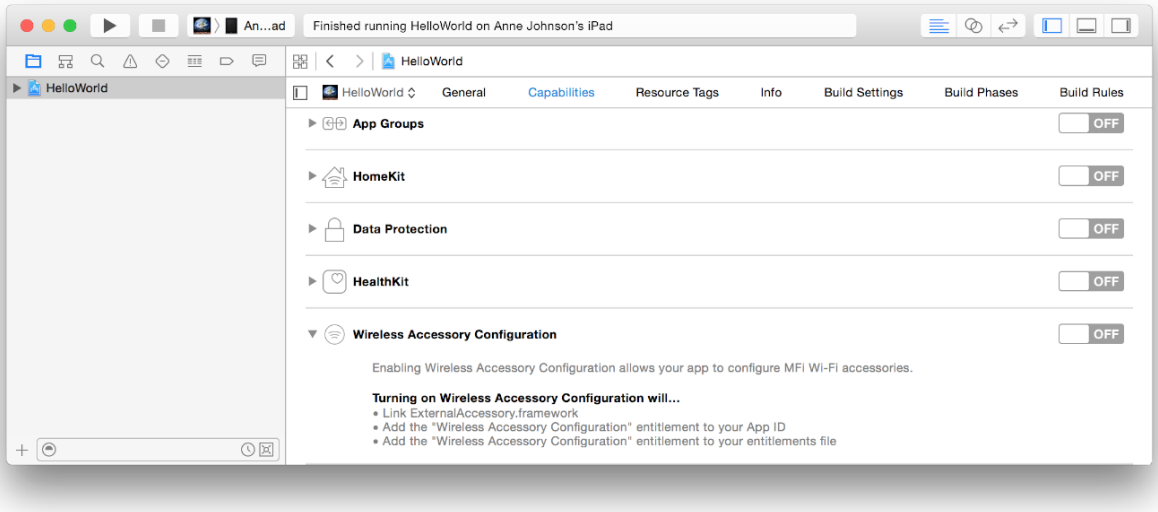
Adding HealthKit (iOS, WatchKit Extension)

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)

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.



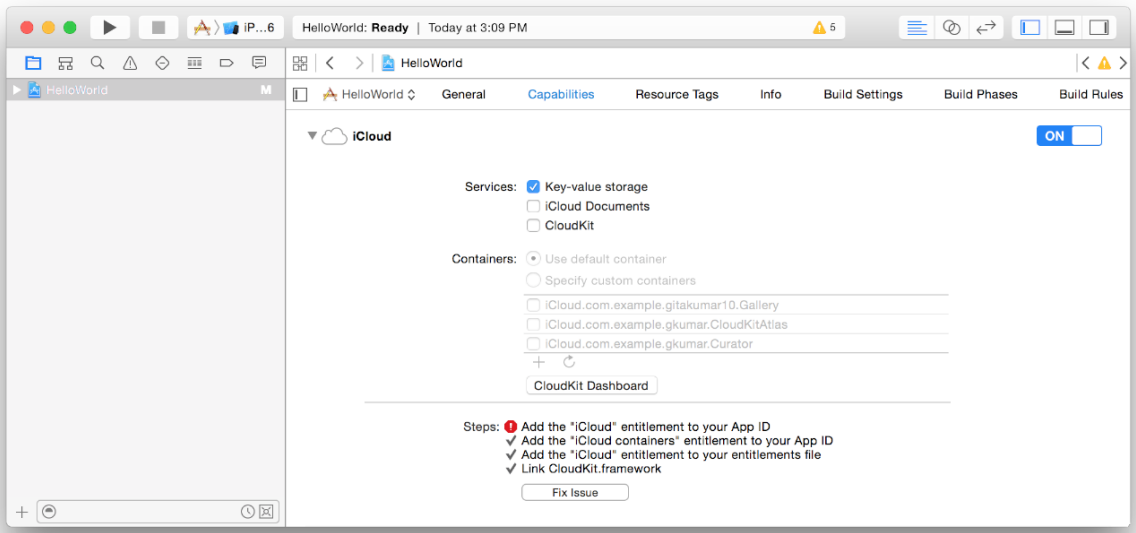
For information about the External Accessory framework, read *External Accessory Programming Topics*.

Configuring Newsstand (iOS, WatchKit Extension)

Newsstand enables an 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, tvOS, and watchOS apps, a 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 your developer account and iTunes Connect.