Exporting Your App for Testing (iOS, tvOS, watchOS)

Before uploading your app to iTunes Connect, optionally distribute it for testing on registered devices using an ad hoc provisioning profile or team provisioning profile. These distribution methods allow you to test variants of your app that are built locally by Xcode. Testers don't need to be team members or iTunes Connect users to run the app, but their devices need to be registered in your developer account. You can register up to 100 devices per product family per year that your team uses for development and testing. Choose one of these methods if you can afford to use a portion of these devices for testing and can collect device IDs from your testers.

Note: If you are a member of the Apple Developer Enterprise Program, you don't have access to iTunes Connect, so use this method for beta testing your apps.

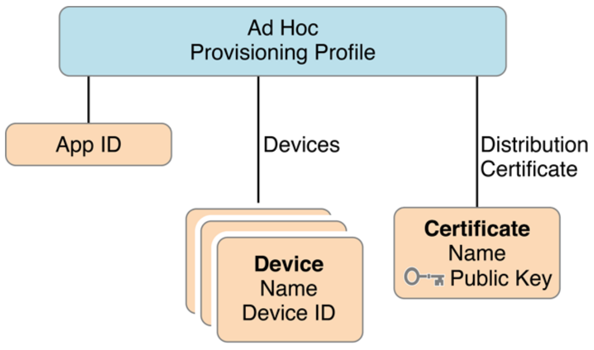
These are the steps to export your app for testing:

1. Register all test devices.
2. Archive your app.
3. Export the archive using either an ad hoc provisioning profile or team provisioning profile to code sign your app.
4. Install the app on test devices.
5. Solicit crash reports from testers.

About Ad Hoc Provisioning Profiles

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

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



Each device in an ad hoc provisioning profile is identified by its *unique device ID (UDID)*. The devices you register and add to a provisioning profile are stored in your developer account. Each individual or organization can register up to 100 devices per product family per membership year for development and testing. You can register 100 devices of each type per year. For iOS apps, you can register 100 iPad, 100 iPhone, and 100 iPod Touch devices.

Registering Test Devices

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

Testers don’t need to install Xcode to locate the device ID. For iOS apps, testers can get their device ID using iTunes. Send the instructions in Locating iOS Device IDs Using iTunes (iOS, tvOS) to testers and ask them to send their device IDs to you, or follow these steps to collect your own device IDs. For Apple TV devices, send the instructions in Locating Device IDs Using System Information (iOS, tvOS, Mac) to testers. They need to connect the Apple TV to their Mac to get the device ID.

In your developer account, register one or more devices, as described in Registering Devices Using Your Developer Account.

Archiving Your App

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

To create an archive

1. In the Xcode project editor, choose a generic device—Generic iOS Device, Generic tvOS Device, or Generic iOS Device + watchOS Device—or your device name from the Scheme toolbar menu.

   You can’t create an archive of a simulator build. If a device is connected to your Mac, the device name appears in the Scheme toolbar menu. When you disconnect the device, the menu item changes to the generic device name.

2. Choose Product > Archive.

   The Archives organizer appears and displays the new archive.

Xcode runs preliminary validation tests on the archive and may display a validation warning in the project editor. If you see a warning, fix the issue and create the archive again.

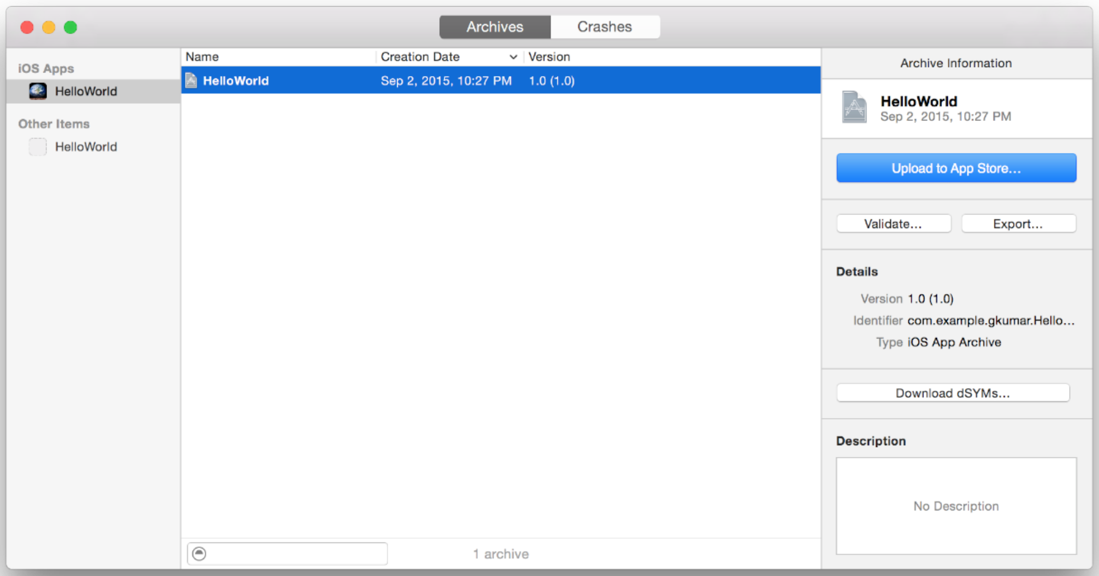
Exporting Your App for Testing Outside the Store

Because testers don’t have Xcode to run your app, you create an *iOS App* file (a file with an `.ipa` filename extension) that they can then use to install your app on their device. Use this method to test a universal app that runs on all supported devices or test device variants that the store distributes later to users.

Note: tvOS binaries are also iOS App files with an `.ipa` filename extension.

To create an iOS App file for testing

1. Open the Archives organizer (choose Organizer from the Window menu), and select the archive.



Name	Creation Date	Version
HelloWorld	Sep 2, 2015, 10:27 PM	1.0 (1.0)

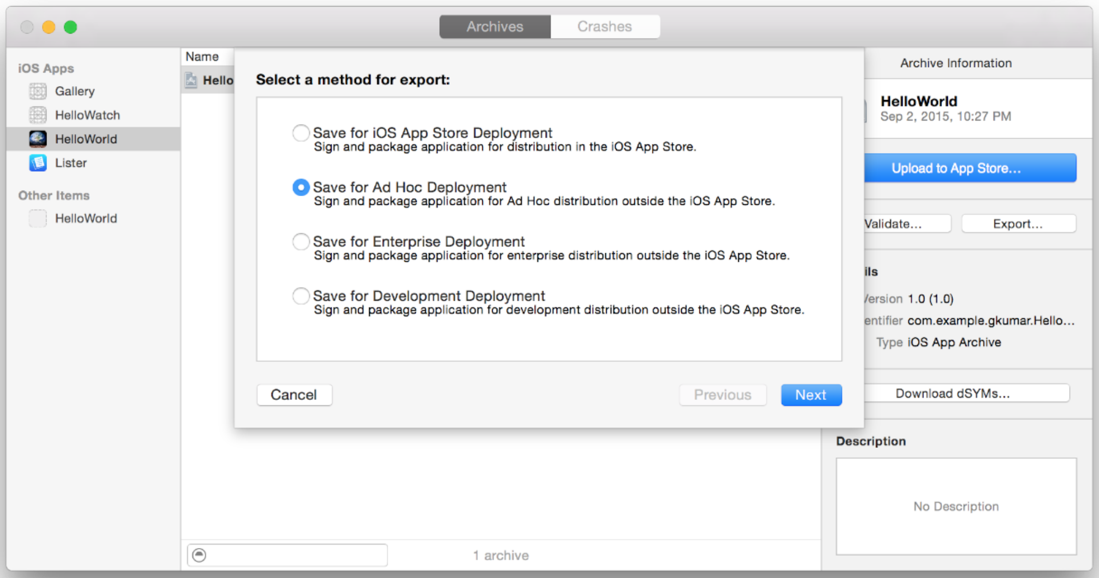
HelloWorld
Sep 2, 2015, 10:27 PM

Details
Version 1.0 (1.0)
Identifier com.example.gkumar.Hello...
Type iOS App Archive

2. Click the Export button, select an export option, and click Next.

   To distribute your app to users with designated devices, select “Save for Ad Hoc Deployment.” The app will be code signed with the distribution certificate.

   To distribute your app for internal testing, select “Save for Development Deployment.” The app will be code signed with your development certificate.

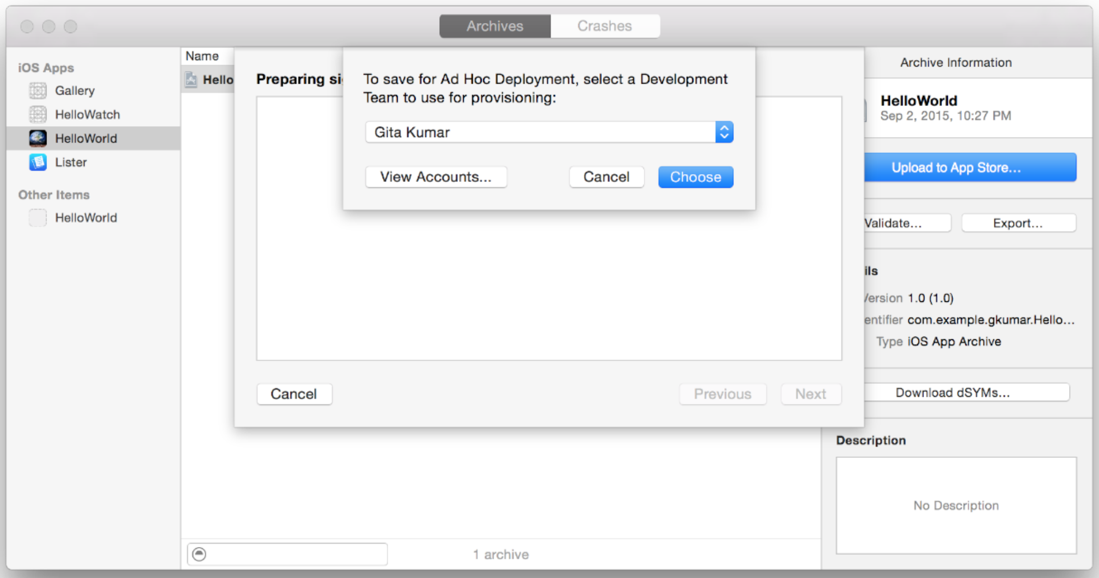


Select a method for export:

- Save for iOS App Store Deployment
  Sign and package application for distribution in the iOS App Store.
- [✓ Save for Ad Hoc Deployment]
  Sign and package application for Ad Hoc distribution outside the iOS App Store.
- Save for Enterprise Deployment
  Sign and package application for enterprise distribution outside the iOS App Store.
- Save for Development Deployment
  Sign and package application for development distribution outside the iOS App Store.

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

   If necessary, Xcode creates the needed signing identity and provisioning profile for you.

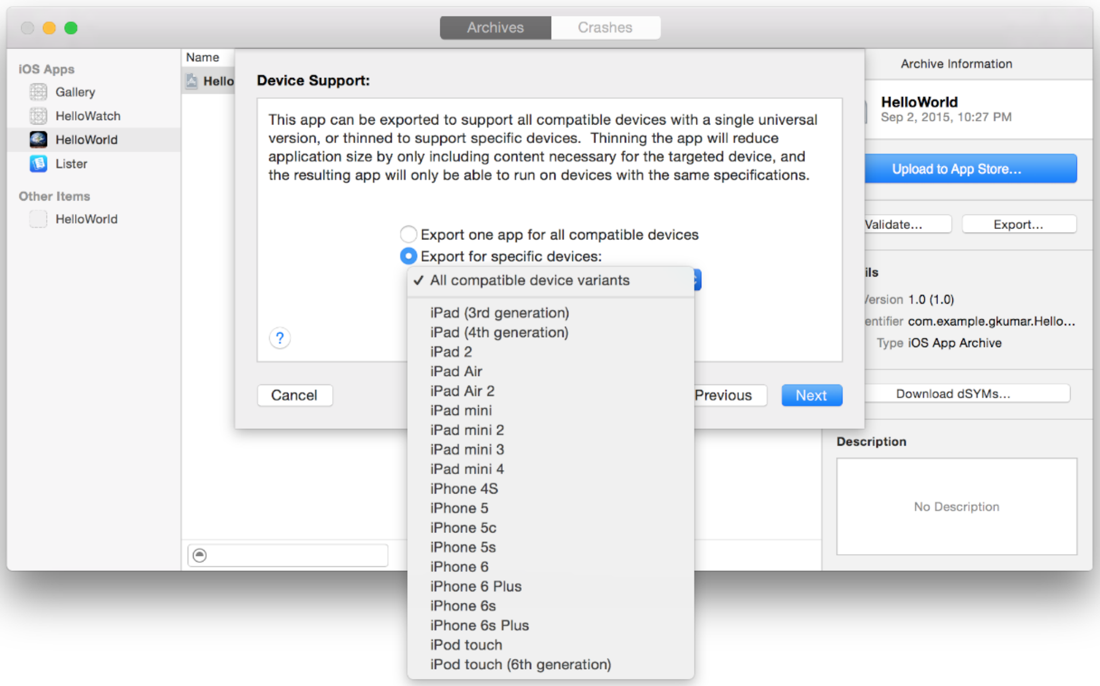


To save for Ad Hoc Deployment, select a Development Team to use for provisioning:

Gita Kumar

4. In the Device Support dialog, choose whether to export the universal app or a variant for a specific device, and click Next.

   - If you want to run the app on any supported device, select “Export one app for all compatible devices.”
   - If you want to test all device variances, select “Export for specific devices” and choose “All compatible device variants” from the pop-up menu.
   - If you want to test a specific device variant, select “Export a thinned app for a specific device” and choose the device family from the pop-up menu.


Device Support:

This app can be exported to support all compatible devices with a single universal version, or thinned to support specific devices. Thinning the app will reduce application size by only including content necessary for the targeted device, and the resulting app will only be able to run on devices with the same specifications.

Export one app for all compatible devices [✓ Export for specific devices:]

✓ All compatible device variants

iPad (3rd generation)
iPad (4th generation)
iPad 2
iPad Air
iPad Air 2
iPad mini
iPad mini 2
iPad mini 3
iPad mini 4
iPhone 4S
iPhone 5
iPhone 5c
iPhone 5s
iPhone 6
iPhone 6 Plus
iPhone 6s
iPhone 6s Plus
iPod touch
iPod touch (6th generation)

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

   The ad hoc provisioning profile should begin with the text `XC Ad Hoc:`. The team provisioning profile should begin with the text `[platform] Team Provisioning Profile: [App ID]` (see Team Provisioning Profiles in Depth).

6. Review the build options, and click Next.

   - If you use on-demand resources, check "Include manifest for over-the-air installation."

     If you distribute your app outside of the store, you need to host the on-demand resources yourself. The manifest file is an XML plist used by a device to find, download, and install apps from your web server.

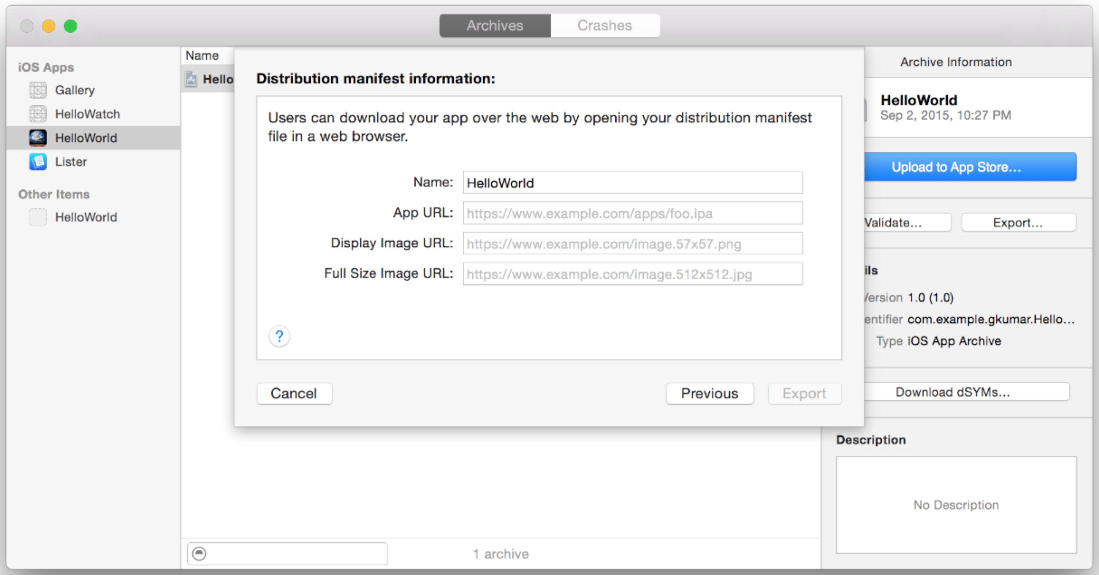
   - If you want to test a build created from bitcode, check "Export from bitcode."

     Bitcode is an intermediate representation of your compiled app that allows the store to build your binaries, described in Bitcode. You can test binaries created from bitcode using Xcode, but should also test the binaries created by the store using TestFlight, described in Distributing Your App Using TestFlight (iOS, tvOS, watchOS).

7. If you request a manifest file, enter details about your web server in the "Distribution manifest information" dialog that appears.

   Enter the following information:

   - **Name.** The name of the app displayed during download and installation.
   - **App URL.** A fully qualified HTTPS URL for the iOS App file.
   - **Display Image URL.** A fully qualified HTTPS URL for an app icon that is displayed during download and installation. The image file must be 57 x 57 pixels and in PNG format.
   - **Full Size Image URL.** A fully qualified HTTPS URL for a larger image that is displayed in iTunes. The image file must be 512 x 512 pixels and in PNG format.



Distribution manifest information:

Users can download your app over the web by opening your distribution manifest file in a web browser.

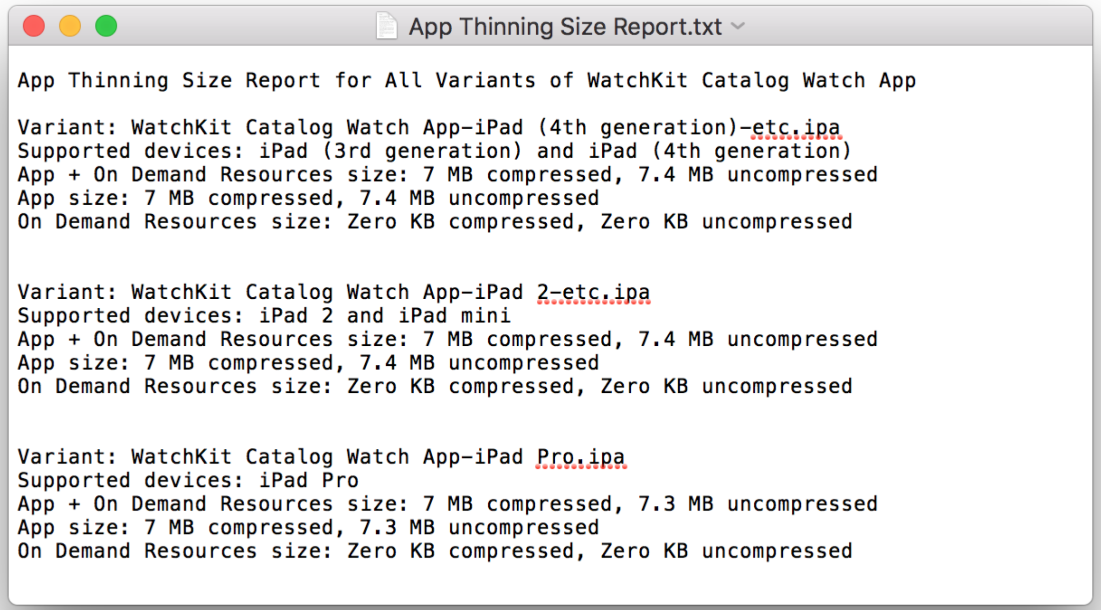
Name: HelloWorld
App URL: https://www.example.com/apps/foo.ipa
Display Image URL: https://www.example.com/image.57x57.png
Full Size Image URL: https://www.example.com/image.512x512.jpg

Version 1.0 (1.0)
Identifier com.example.gkumar.Hello...
Type iOS App Archive

Cancel Previous Export

8. Enter a filename and location for the file, and click Export.

Xcode creates a folder containing the iOS App files you specified. If you request a manifest file, a `manifest.plist` file appears in the same location as the iOS App file. If you request variances, app thinning metrics are also included in the folder. The `App Thinning Size Report` file contains a summary of the sizes of all variants and the `app-thinning.plist` file contains more details.



```
App Thinning Size Report for All Variants of WatchKit Catalog Watch App

Variant: WatchKit Catalog Watch App-iPad (4th generation)-etc.ipa
Supported devices: iPad (3rd generation) and iPad (4th generation)
App + On Demand Resources size: 7 MB compressed, 7.4 MB uncompressed
App size: 7 MB compressed, 7.4 MB uncompressed
On Demand Resources size: Zero KB compressed, Zero KB uncompressed


Variant: WatchKit Catalog Watch App-iPad 2-etc.ipa
Supported devices: iPad 2 and iPad mini
App + On Demand Resources size: 7 MB compressed, 7.4 MB uncompressed
App size: 7 MB compressed, 7.4 MB uncompressed
On Demand Resources size: Zero KB compressed, Zero KB uncompressed


Variant: WatchKit Catalog Watch App-iPad Pro.ipa
Supported devices: iPad Pro
App + On Demand Resources size: 7 MB compressed, 7.3 MB uncompressed
App size: 7 MB compressed, 7.3 MB uncompressed
On Demand Resources size: Zero KB compressed, Zero KB uncompressed
```

Installing Your App on Test Devices Using Xcode

You can install iOS App files on devices using Xcode.

To install an app on a device using Xcode

1. Connect the device to your Mac.

2. In Xcode, choose Window > Devices and select the device under Devices.
3. In the Installed Apps table, click the Add button (+) below the table.
4. In the dialog that appears, choose the iOS App file and click Open.

For more tasks you can perform in the Devices window, read Managing Apps on Devices.

Installing Your App on Test Devices Using iTunes (iOS, watchOS)

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

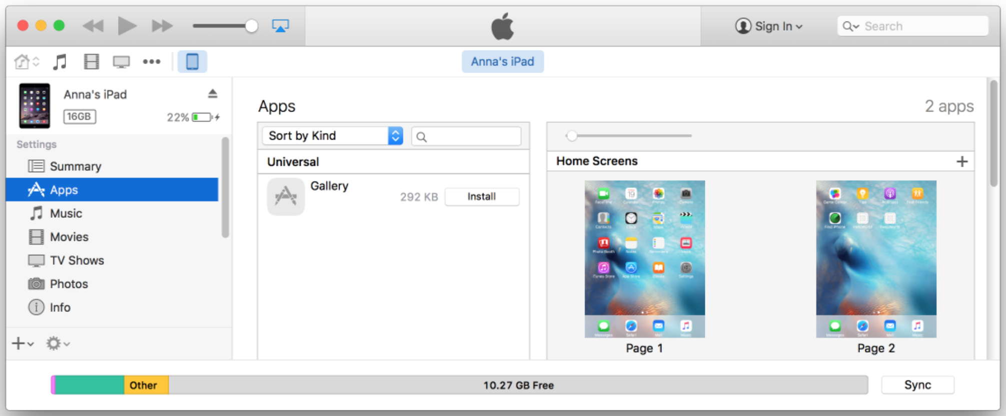
To install the app on an iOS device using iTunes

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

   If possible, don't use a Mac that you use for development. For watchOS apps, connect an iPhone paired with an Apple Watch.

2. Double-click the iOS App file that you created earlier.
3. In iTunes, click the device in the upper-left corner of the window.
4. Click the Apps button.

   The app appears in the iTunes Apps list.



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

   An Install or Remove button appears adjacent to the app.

6. If an Install button appears, click it.

   The button text changes to Will Install.

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

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

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

Installing Your App on Test Devices Using Apple Configurator 2 (iOS, watchOS, tvOS)

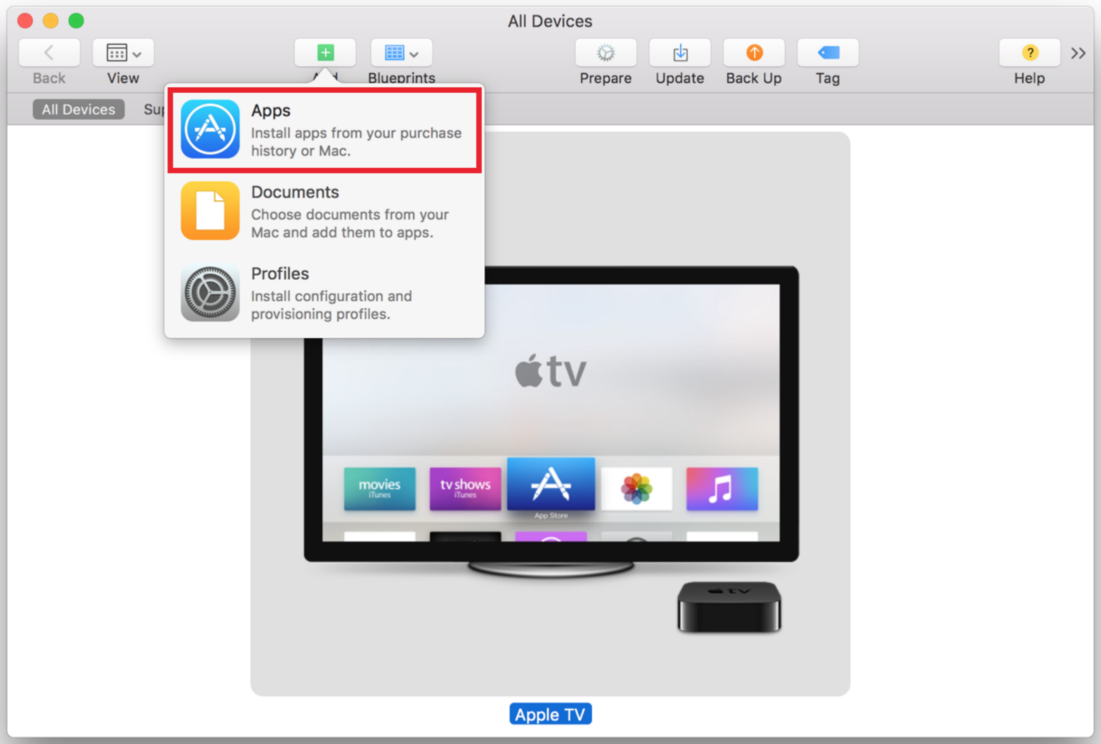
Apple Configurator 2 is a free app from the Mac App Store that makes it easy to install iOS and tvOS apps on connected test devices. For tvOS apps, this is the only way to install an iOS App file on an Apple TV without using Xcode.

To install the app on a device using Apple Configurator 2

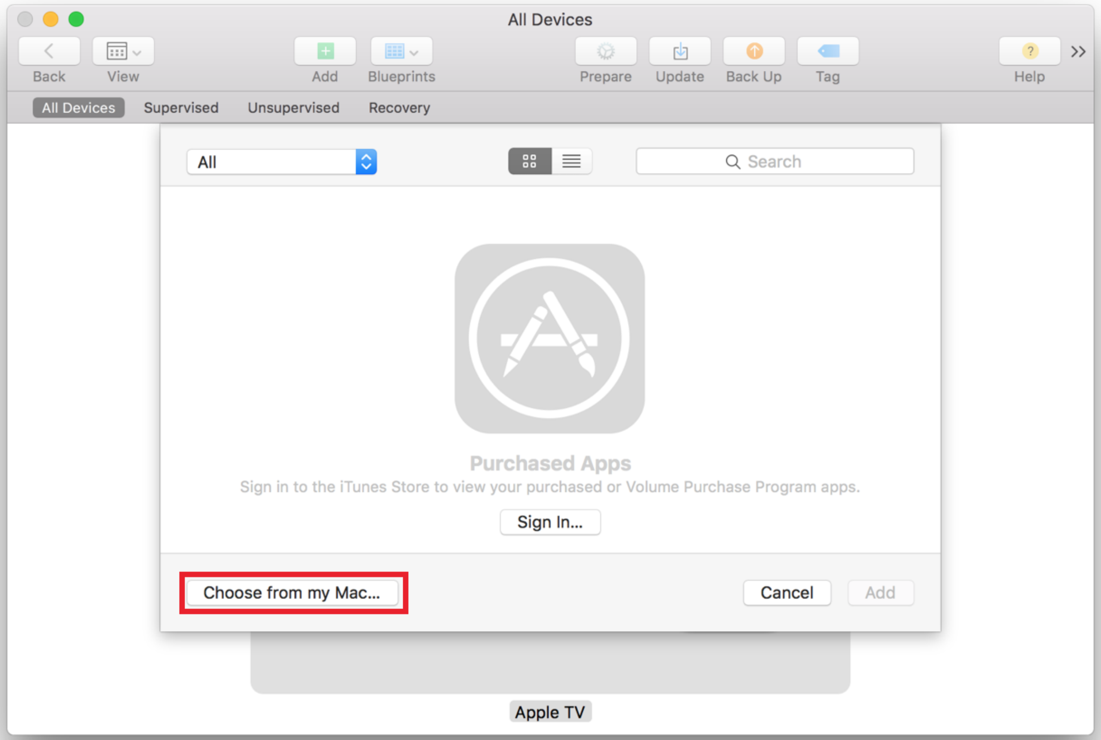
1. Connect the testing device to a Mac running Apple Configurator 2.

   If possible, don't use a Mac that you use for development. For watchOS apps, connect an iPhone paired with an Apple Watch.

2. Select the device, click the Add button (+), and select Apps from the pop-up menu.


All Devices
Back
View
Add
Blueprints
Prepare
Update
Back Up
Tag
Help
All Devices
Apps
Install apps from your purchase history or Mac.
Documents
Choose documents from your Mac and add them to apps.
Profiles
Install configuration and provisioning profiles.
Apple TV

3. In the dialog that appears, click "Choose from my Mac".



4. Choose the iOS App file that you created earlier and click Add.

Distributing Your App Using Xcode Server

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

Copying App Sandbox Data

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

Soliciting Crash Reports from Testers

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

Note: If you use TestFlight to distribute your prerelease app, described in Distributing Your App Using TestFlight (iOS, tvOS, watchOS), Apple collects and aggregates crash logs into crash reports for you.

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

To send crash reports from a Mac

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

   iTunes downloads the crash reports to your Mac.

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

To send crash reports from Windows

1. In the Windows search field, enter the crash log directory for your operating system, replacing `<user_name>` with your Windows user name.
   - For crash log storage on Windows, type:

     ```
     C:\Users\<user_name>\AppData\Roaming\Apple
     computer\Logs\CrashReporter/MobileDevice
     ```

   - For crash log storage on Windows XP, type:

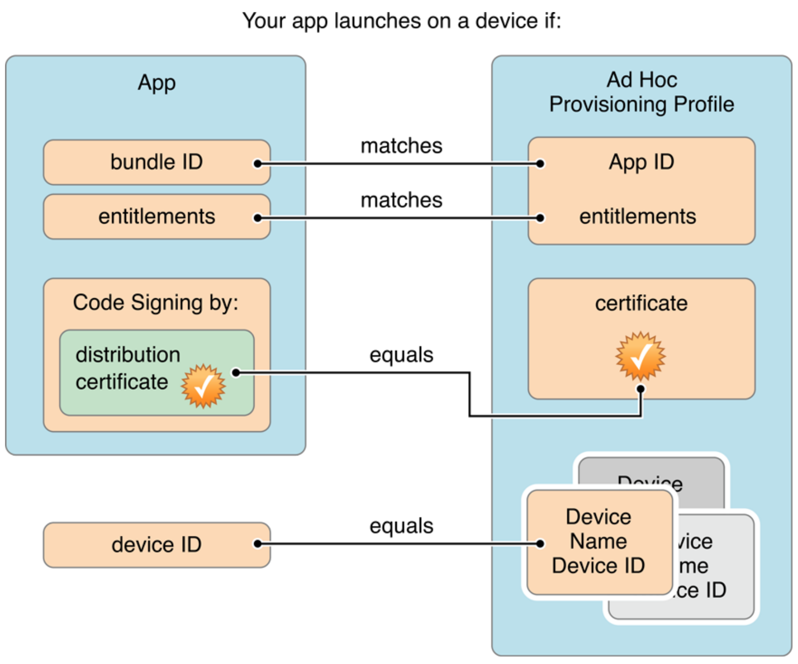
     ```
     C:\Documents and Settings\<user_name>\Application Data\Apple
     computer\Logs\CrashReporter
     ```

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

To learn how to interpret the reports that you receive from testers, read Viewing and Importing Crashes in the Devices Window.

Ad Hoc Provisioning Profiles in Depth

You signed the iOS App file using the distribution certificate specified in the ad hoc provisioning profile. The ad hoc provisioning profile was included in the app bundle when you built and archived the app. Then you installed the app on a test device. The app successfully launches if the app's bundle ID matches the App ID, the signature matches the distribution certificate, and the device is in the device list of the ad hoc provisioning profile.



Recap

In this chapter, you learned how to distribute your app on designated test devices using ad hoc provisioning profiles. You also received instructions to send to ad hoc testers to install the beta version of your app and send crash reports to you.
