Uploading Your App to iTunes Connect

Upload the version and build of your app to iTunes Connect that you want to distribute for testing or submit to the store. Optionally, run iTunes Connect validation tests before uploading your app. Later use iTunes Connect to manage the different versions and builds you upload, described in Viewing Builds in *iTunes Connect Developer Guide*.

iTunes Connect extracts information from your app bundle and associates it with a version of your app in iTunes Connect so perform all these steps before uploading your app.

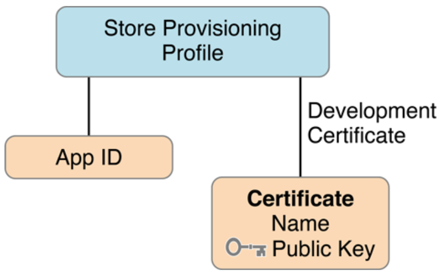
1. Create an iTunes Connect app record.
2. Update the version number and build string.
3. Create an archive of your app.
4. Validate the archive.
5. Upload your app to iTunes Connect.

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

You should submit to the store the last archive you distributed for testing, so before you upload your final candidate, review Apple guidelines and the Xcode project configuration, described in Prepare Your App for Submission.

About Store Provisioning Profiles

A *store provisioning profile* is a distribution provisioning profile that authorizes your app to use certain app services and ensures that you are the one uploading your app. A store provisioning profile contains a single App ID that matches one or more of your apps and a distribution certificate. You configure the App ID indirectly through Xcode to use certain app services. You enable and configure app services by setting entitlements and performing other configuration steps. Some entitlements are enabled for an App ID (a set of apps created by your team), and others are set in the Xcode project. When you upload your app to the store, Xcode signs the app bundle with the distribution certificate referenced in the store provisioning profile.



Creating Your App Record in iTunes Connect

You must have an iTunes Connect app record to validate and upload your app to iTunes Connect. To create your app record, read Creating an App Record before continuing. The bundle ID you enter in iTunes Connect must match the bundle ID in your Xcode project. After you upload your first build, you can't change the bundle ID in iTunes Connect. To change your bundle ID in the Xcode project, read Setting the Bundle ID.

Note: If your app uses encryption, consider providing the required export compliance information in the Xcode project before uploading the build, as described in Adding Export Compliance Information to Your Project (Optional). This may avoid delays later when you attempt to distribute your app using TestFlight or through the store.

Updating the Version and Build Strings

Update the version and build string, as described in Setting the Version Number and Build String, because iTunes Connect uses these to match the build with the app version.

Important: Update the version number when you create a new app version in iTunes Connect. Update the build string before you upload a new build of your app to iTunes Connect. The number represented by the build string should be incremented. For iOS apps, iTunes will recognize that the build string changed and properly sync the new app version to the device. The version number and build string are also used to identify crash reports and `.dSYM` files for apps distributed through TestFlight or the store.

Archiving Your App

You create an archive of your app regardless of the type of distribution method you select. Xcode archives allow you to build your app and store it, along with critical debugging information, in a bundle that's managed by Xcode. For example, if you distribute a build of your app to testers, archiving the debugging information makes it easier to interpret crash reports later.

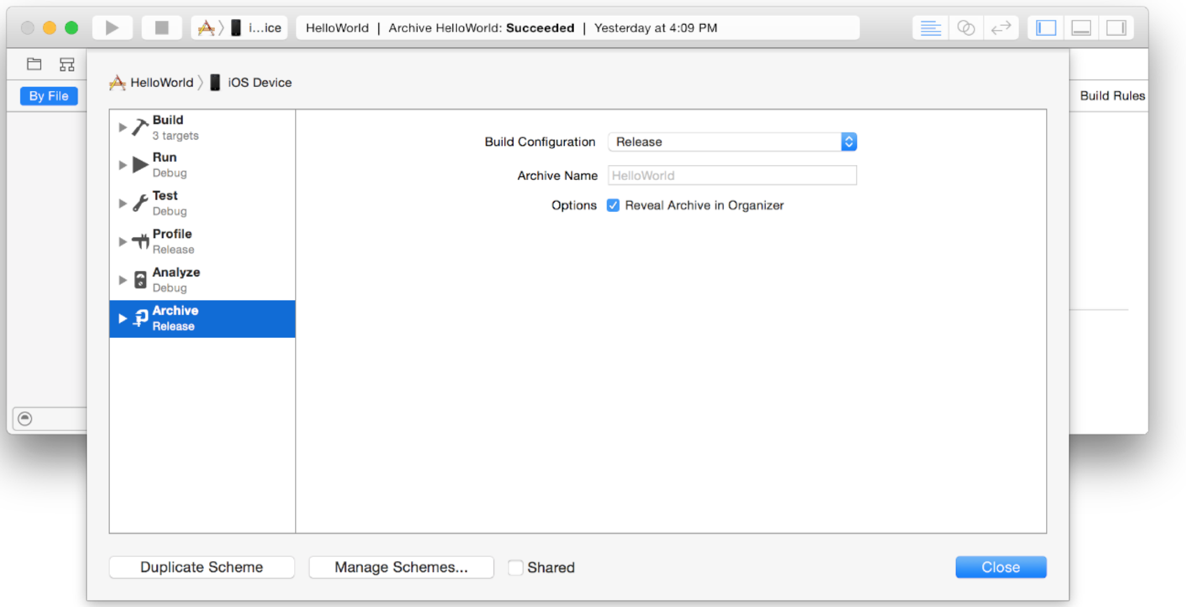
Important: Save the archive for every version and build of an app you distribute. If you upload your app to iTunes Connect, you'll need the archive and associated project to jump from a stack frame in the Crashes log to the source code in the debug navigator, described in Analyzing Crash Reports. You'll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

Reviewing the Archive Scheme Settings

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

To review the Archive scheme

1. In the Xcode project editor, choose Product > Scheme > Edit Scheme to open the scheme editor.
2. Click Archive in the column on the left.


HelloWorld | Archive HelloWorld: **Succeeded** | Yesterday at 4:09 PM

Build Configuration: Release
Archive Name: HelloWorld
Options: ✓ Reveal Archive in Organizer

3. From the Build Configuration pop-up menu, choose Release, and click Close.

Creating an Archive

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

To create an archive

1. For iOS, tvOS, and watchOS apps, choose a generic device or your device name from the Scheme toolbar menu in the project editor.

   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.

2. Choose Product > Archive.

   The Archives organizer appears and displays the new archive.

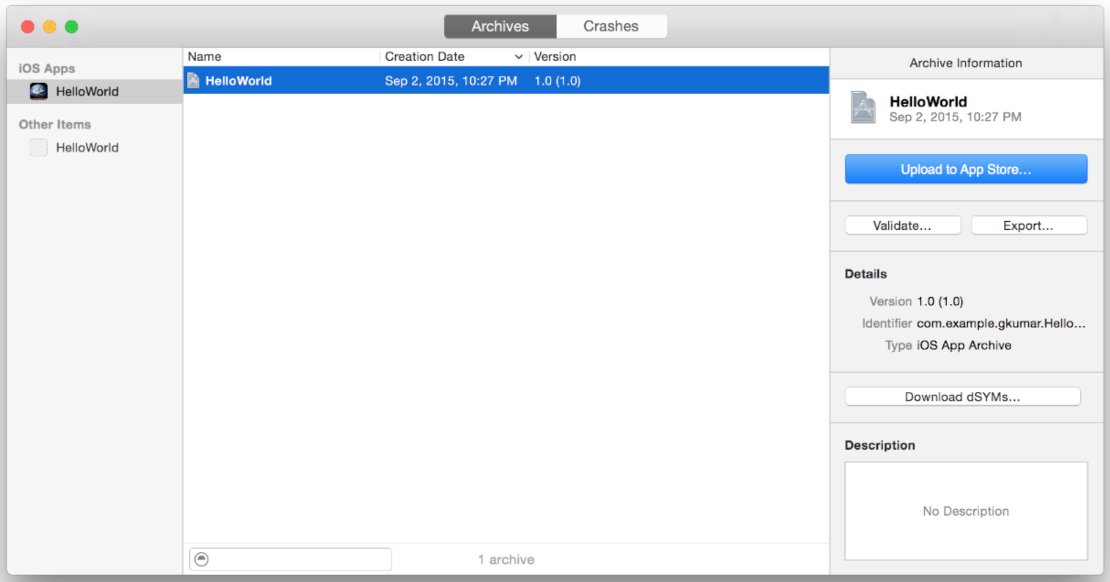
Xcode runs preliminary validation tests on the archive and may display a validation warning in the project editor. If you see a warning, fix the issue and create the archive again.

Running iTunes Connect Validation Tests

To discover issues early, validate your archive before uploading it to iTunes Connect. The validation tests are performed by iTunes Connect, which checks whether your Xcode project is configured correctly for the store. For example, it reports a problem if you don't set required app icons, as described in Setting Individual App Icon and Launch Image Files. The same validation tests are performed again later when you upload your app. However, iTunes Connect performs further validation tests after you upload your app.

To validate an archive

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



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

HelloWorld
Sep 2, 2015, 10:27 PM

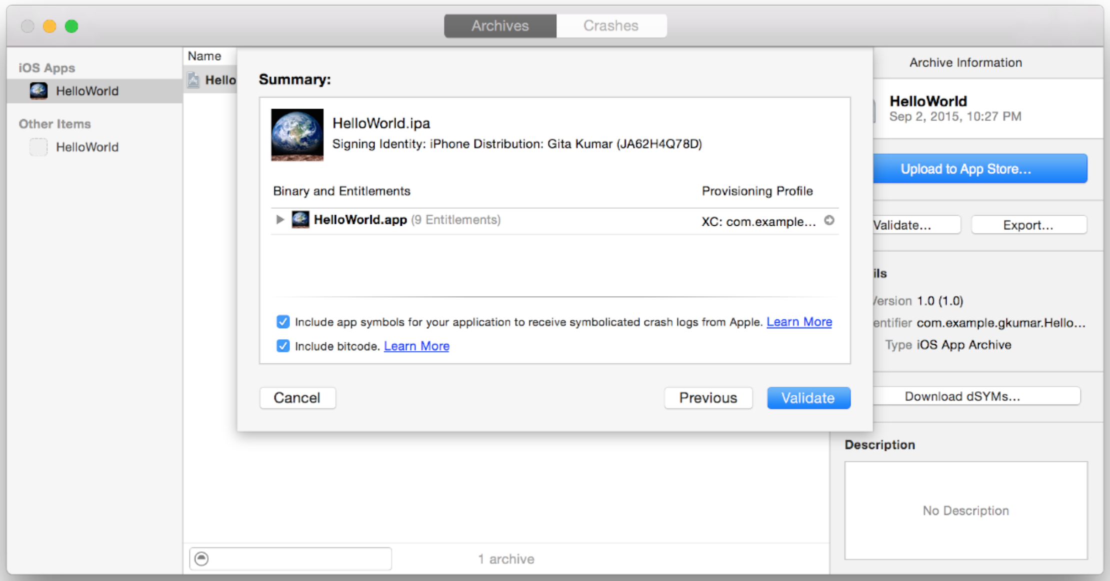
Details
Version 1.0 (1.0)
Identifier com.example.gkumar.Hello...
Type iOS App Archive

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

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

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

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



Summary:

HelloWorld.ipa
Signing Identity: iPhone Distribution: Gita Kumar (JA62H4Q78D)

Binary and Entitlements	Provisioning Profile
HelloWorld.app (9 Entitlements)	XC: com.example…

✓ Include app symbols for your application to receive symbolicated crash logs from Apple. Learn More
✓ Include bitcode. Learn More

HelloWorld
Sep 2, 2015, 10:27 PM

ils
Version 1.0 (1.0)
entifier com.example.gkumar.Hello…
Type iOS App Archive

Description
No Description

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

Fix any validation issues, create a new archive, and validate it again. Validation errors don’t prevent you from uploading your app and distributing it using TestFlight.

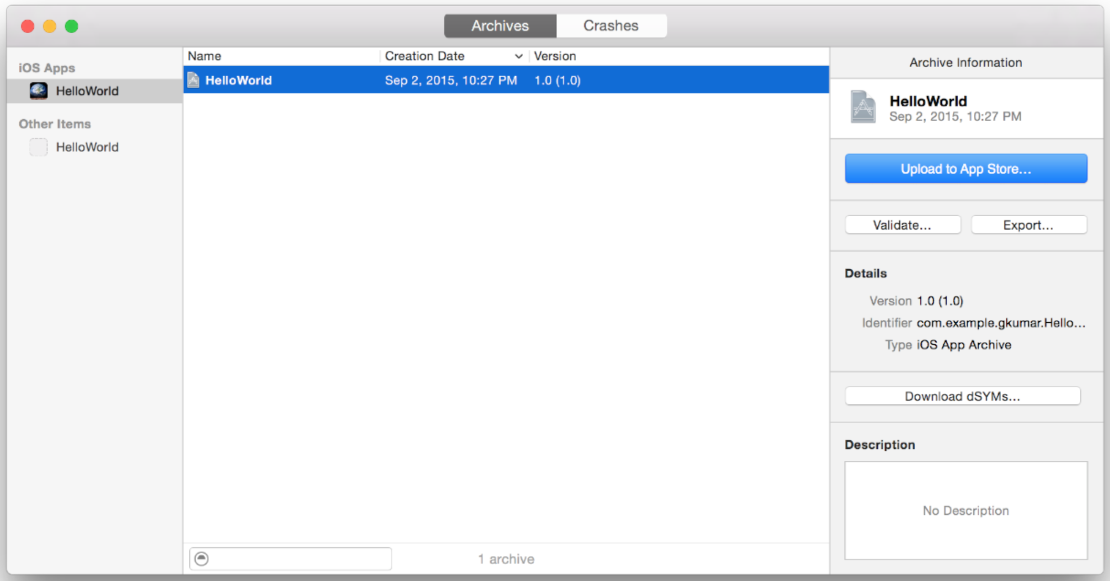
Uploading Your App

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

You can upload the same build you distribute for testing that you later submit to the store. It’s important to test the *exact* same build of your app that you submit to the store. If you include bitcode, described in Bitcode, test the variants that the store builds for you using TestFlight, described in Distributing Your App Using TestFlight (iOS, tvOS, watchOS). For Mac apps, use the same archive to export your app for testing, described in Exporting Your App for Testing (Mac), that you upload to iTunes Connect.

To upload your app to iTunes Connect

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



> **Warning:** If the archive you selected was built with a custom toolchain, the "Upload to App Store" button is disabled. Re-create the archive using a standard toolchain to distribute it through the store or using TestFlight.

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

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

3. In the dialog that appears, review the app, its entitlements, and provisioning profile.
4. To receive crash reports from Apple, check "Include app symbols for your application…".

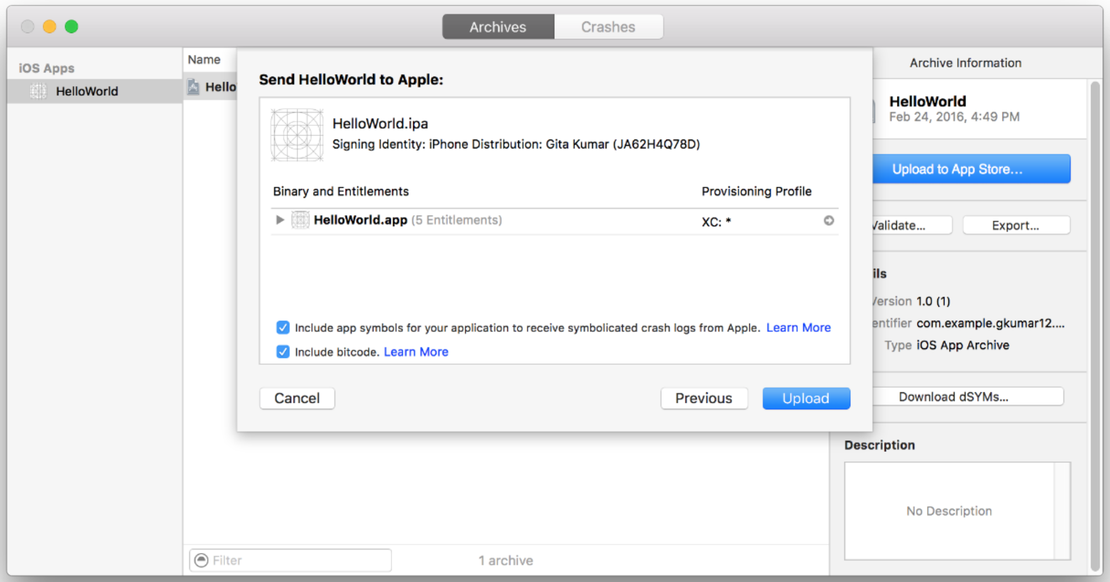
   Later, you can view symbolicated crash logs in the Crashes organizer, described in Analyzing Crash Reports.

5. To include bitcode, ensure that "Include bitcode…" is checked (bitcode is enabled by default).

   Apps you upload to iTunes Connect that contain bitcode will be compiled and linked on the store. Including bitcode allows the store to apply updates to your app without requiring you to create a new version and resubmit it to App Review.

   - For iOS apps, bitcode is optional.
   - For watchOS apps, bitcode is required. If you uncheck this box, only the bitcode for the enclosed iOS app is removed.
   - For tvOS apps, bitcode is required so this checkbox doesn't appear.
   - For Mac apps, bitcode is not supported so this checkbox doesn't appear.

   If you provide bitcode, all apps and frameworks in the app bundle need to include bitcode.



Send HelloWorld to Apple:

HelloWorld.ipa
Signing Identity: iPhone Distribution: Gita Kumar (JA62H4Q78D)

Binary and Entitlements	Provisioning Profile
HelloWorld.app (5 Entitlements)	XC: *

✓ Include app symbols for your application to receive symbolicated crash logs from Apple. Learn More
✓ Include bitcode. Learn More

Cancel | Previous | Upload

Important: To download the dSYMs after you distribute your app using iTunes Connect, select the archive and click Download dSYMs. For more details, read Viewing and Importing Crashes in the Devices Window.

6. Click Upload.

   Xcode uploads the archive to iTunes Connect and iTunes Connect runs validation tests.

7. If issues are found, read the error messages, and click Done.

   If no application record can be found, create an app record in iTunes Connect, and repeat these steps.

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

8. If no issues are found, click Done.

Xcode transmits the archive to Apple, where the build is examined to determine whether it conforms to Apple guidelines. If the build is rejected, correct the problems that were identified and upload a new build. If you successfully upload your app, view the version and build of your app in iTunes Connect, as described in Viewing Binary Details. Use iTunes Connect to select distribution options. For example, distribute your app for testing by inviting internal or external testers, described in TestFlight Beta Testing (Optional). If you are ready to submit your app to App Review, read Submitting Your App to the Store.