Packaging Mac Software for Distribution

This thread has been locked by a moderator.

1.4k

This post is one of a pair of posts, the other one being Creating Distribution-Signed Code for Mac, that replaces my earlier Signing a Mac Product For Distribution post.

For more background on this, see the notes at the top of Creating Distribution-Signed Code for Mac.

Share and Enjoy
—
Quinn “The Eskimo!” @ Developer Technical Support @ Apple
`let myEmail = "eskimo" + "1" + "@" + "apple.com"`

Packaging Mac Software for Distribution

Build a zip archive, disk image, or installer package for distributing your Mac software.

Overview

Xcode is a great tool for creating and distributing Mac apps. Once you’ve written your code you can upload it to the App Store with just a few clicks. However, Xcode cannot do everything. For example:

- Some Mac software products are not apps. You might, for example, be creating a product that includes a daemon.
- Some Mac products include multiple components. Your daemon might include an app to configure it.
- Some Mac products ship outside of the App Store, and so need to be packaged for distribution. For example, you might choose to distribute your daemon and its configuration app in an installer package.
- Some Mac products are built with third-party developer tools.

If your product cannot be built and distributed using Xcode alone, follow these instructions to package it for distribution.

Note If you use a third-party developer tool to build your app, consult its documentation for advice specific to that tool.

To start this process you need distribution-signed code. For detailed advice on how to create distribution-signed code, see Creating Distribution-Signed Code for Mac.

If you ship your product frequently, create a script to automate the distribution process.

Decide on a Container Format

To get started, decide on your container format. Mac products support two distribution channels:

- The Mac App Store, for apps
- Independent distribution, for apps and non-apps, using Developer ID signing

A Mac App Store app must be submitted as an installer package. In contrast, products distributed outside of the Mac App Store use a variety of different container formats, the most common being:

- Zip archive (`.zip`)
- Disk image (`.dmg`)
- Installer package (`.pkg`)

You may choose to nest these containers. For example, you might ship an app inside an installer package on a disk image. Nesting containers is straightforward: Just work from the inside out, following the instructions for each container at each step.

IMPORTANT Sign your code and each nested container (if the container supports signing). For example, if you ship an app inside an installer package on a disk image, sign the app, then create the installer package, then sign that package, then create the disk image, then sign the disk image.

Each container format has its own pros and cons, so choose an approach based on the requirements of your product.

Build a Zip Archive

If you choose to distribute your product in a zip archive, use the `ditto` tool to create that archive:

1. Create a directory that holds everything you want to distribute.
2. Run the `ditto` tool as shown below, where `DDD` is the path to the directory from step 1 and `ZZZ` is the path where `ditto` creates the zip archive.

```
% ditto -c -k --keepParent DDD ZZZ
```

Zip archives cannot be signed, although their contents can be.

Build an Installer Package

If you choose to distribute your product in an installer package, start by determining your installer signing identity. Choose the right identity for your distribution channel:

- If you’re distributing an app on the Mac App Store, use a Mac Installer Distribution signing identity. This is named `3rd Party Mac Developer Installer: TTT`, where `TTT` identifies your team.
- If you’re distributing a product independently, use a Developer ID Installer signing identity. This is named `Developer ID Installer: TTT`, where `TTT` identifies your team.

For information on how to set up these installer signing identities, see Developer Account Help.

Run the following command to confirm that your installer signing identity is present and correct:

```
% security find-identity -v
  1) 6210ECCC616B6A72F238DE6FDDFDA1A06DEFF9FB "3rd Party Mac Developer Installer: …"
  2) C32E0E68CE92936D5532E21BAAD8CFF4A6D9BAA1 "Developer ID Installer: …"
     2 valid identities found
```

The `-v` argument filters for valid identities only. If the installer signing identity you need is not listed, see Developer Account Help.

IMPORTANT Do not use the `-p codesigning` option to filter for code signing identities. Installer signing identities are different from code signing identities and the `-p codesigning` option filters them out.

If your product consists of a single app, use the `productbuild` tool to create a simple installer package for it:

```
% productbuild --sign III --component AAA /Applications PPP
```

In this command:

- `III` is your installer signing identity.
- `AAA` is the path to your app.
- `PPP` is the path where `productbuild` creates the installer package.

The above is the simplest possible use of `productbuild`. If you’re submitting an app to the Mac App Store, that’s all you need. If you have a more complex product, you’ll need a more complex installer package. For more details on how to work with installer packages, see the man pages for `productbuild`, `productsign`, `pkgbuild`, and `pkgutil`. For instructions on how to read a man page, see Reading UNIX Manual Pages.

Build a Disk Image

If you choose to distribute your product in a disk image:

1. Create a directory to act as the source for the root directory of your disk image’s volume.
2. Populate that directory with the items you want to distribute. If you’re automating this, use `ditto` rather than `cp` because `ditto` preserves symlinks.
3. Use `hdiutil` command shown below to create the disk image, where `SSS` is the directory from step 1 and `DDD` is the path where `hdiutil` creates the disk image.
4. Decide on a code signing identifier for this disk image. If you were signing bundled code, you’d use the bundle ID as the code signing identifier. However, disk images have no bundle ID and thus you must choose a code signing identifier for your image. For advice on how to do this, see the *Sign Each Code* section in Creating Distribution-Signed Code for Mac.
5. Use the `codesign` command shown below to sign the disk image, where `III` is your Developer ID Application code signing identity (named `Developer ID Application: TTT`, where `TTT` identifies your team), `BBB` is the code signing identifier you chose in the previous step, and `DDD` is the path to the disk image from step 3.

```
% hdiutil create -srcFolder SSS -o DDD
% codesign -s III --timestamp -i BBB DDD
```

For more information on code signing identities, see the *Confirm Your Code Signing* section in Creating Distribution-Signed Code for Mac.

IMPORTANT Sign your disk image with a code signing identity, not an installer signing identity.

There are various third-party tools that configure a disk image for distribution. For example, the tool might arrange the icons nicely, set a background image, and add a symlink to the Applications folder. If you use such a tool, or create your own tool for this, make sure that the resulting disk image:

- Is signed with your Developer ID Application code signing identity
- Is a UDIF-format read-only zip-compressed disk image (type `UDZO`)

Submit Your App to the Mac App Store

If you’re creating an app for the Mac App Store, submit your signed installer package using either the `altool` command-line tool or the Transporter app. For detailed instructions, see App Store Connect Help > Reference > Upload tools.

Notarize Your Product

If you’re distributing outside of the Mac App Store, notarize the file you intend to distribute to your users. For detailed instructions, see Customizing the Notarization Workflow. Skip the *Export a Package for Notarization* section because you already have the file that you want to submit.

If you’re using nested containers, only notarize the outermost container. For example, if you have an app inside an installer package on a disk image, sign the app, sign the installer package, and sign the disk image, but only notarize the disk image.

The exception to this rule is if you have a custom third-party installer. In that case, see the discussion in Customizing the Notarization Workflow.

Staple Your Product

Once you’ve notarized your product, staple the resulting ticket to the file you intend to distribute. Staple the Ticket to Your Distribution discusses how to do this for an app within a zip archive. The other common container formats, installer packages and disk images, support stapling directly. For example, to staple a tick to a disk image:

```
% xcrun stapler staple FlyingAnimals.dmg
```

Stapling is recommended but not mandatory. However, if you don’t staple a user might find that your product is blocked by Gatekeeper if they try to install or use it while the Mac is offline.

Code Signing Gatekeeper Developer ID

Reply

Posted 7 months ago by eskimo

Add a Comment