Article

# Notarizing Your App Before Distribution

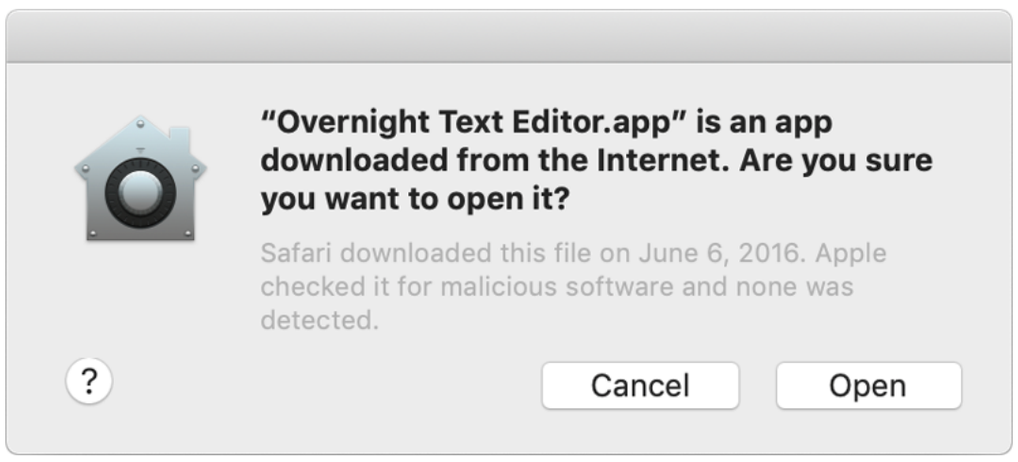
Give users even more confidence in your software by submitting it to Apple for notarization.



## Overview

Notarization gives users more confidence that the Developer ID-signed software you distribute has been checked by Apple for malicious components. Notarization is not App Review. The Apple notary service is an automated system that scans your software for malicious content, checks for code-signing issues, and returns the results to you quickly. If there are no issues, the notary service generates a ticket for you to staple to your software; the notary service also publishes that ticket online where Gatekeeper can find it.

When the user first installs or runs your software, the presence of a ticket (either online or attached to the executable) tells Gatekeeper that Apple notarized the software. Gatekeeper then places descriptive information in the initial launch dialog to help the user make an informed choice about whether to launch the app.




You can notarize several different types of software deliverables, including:

- macOS apps
- Non-app bundles, such as kernel extensions
- Disk images (UDIF format)
- Flat installer packages

Notarization also protects your users if your Developer ID signing key is exposed. The notary service maintains an audit trail of the software distributed using your signing key. If you discover unauthorized versions of your software, you can work with Apple to revoke the tickets associated with those versions.

**Important**

Beginning in macOS 10.14.5, all new or updated kernel extensions and all software from developers new to distributing with Developer ID must be notarized in order to run. In a future version of macOS, notarization will be required by default for all software.

## Prepare Your Software for Notarization

Notarization requires Xcode 10 or later. Building a new app for notarization requires macOS 10.13.6 or later. Uploading and stapling an app requires macOS 10.12 or later.

Apple's notary service requires you to adopt the following protections:

- Enable code-signing for all of the executables you distribute.
- Enable the Hardened Runtime capability for your executable targets, as described in Enable hardened runtime.
- Use a “Developer ID” application, kernel extension, or installer certificate for your code-signing signature. (Don't use a Mac Distribution or local development certificate.) For more information, see Create, export, and delete signing certificates.
- Include a secure timestamp with your code-signing signature. (The Xcode distribution workflow includes a secure timestamp by default. For custom workflows, include the `--timestamp` option when running the `codesign` tool.)
- Don’t include the `com.apple.security.get-task-allow` entitlement with the value set to any variation of `true`. If your software hosts third-party plug-ins and needs this entitlement to debug the plug-in in the context of a host executable, see Avoid the Get-Task-Allow Entitlement.
- Link against the macOS 10.9 or later SDK.

Apple recommends that you notarize all of the software that you’ve distributed, including older releases, and even software that doesn’t meet all of these requirements or that is unsigned. Apple’s notary service uses a variety of methods, including telemetry, to determine which of the above rules to relax for preexisting software. For more information, see Notarize Your Preexisting Software.

**Important**

Some preexisting software might not run properly after being successfully notarized. For example, Gatekeeper might find code signing issues that a relaxed notarization process didn’t enforce. Always review the notary log for any warnings, and test your software before distribution.

## Add the Entitlements Needed by Plug-ins

When you enable the extra security enforced by the hardened runtime, as notarization requires, this impacts both your app and any plug-ins that your app hosts. Plug-ins don’t declare their own entitlements. Instead, they inherit the entitlements of the host process. Therefore, a host app must include all the entitlements that prospective plug-ins require, even when the plug-ins are notarized separately.

For example, if a plug-in employs deep integration with the host executable via C function

pointer overrides, or uses a JavaScript engine for custom workflows, the host executable must declare the `Allow Unsigned Executable Memory Entitlement` or `Allow Execution of JIT-compiled Code Entitlement`, respectively. In some cases, a plug-in fails to even load if the host executable lacks the proper entitlement.

Also include resource access entitlements, like the Address Book or Location access entitlements, and the related purpose strings, that support your app's plug-ins. For example, if a Print Dialog Extension (PDE) that provides fax services wants to access a user's contact list, the host executable must declare the `Address Book Entitlement` and include the `NSContactsUsageDescription` purpose string in its Information Property List for the plug-in to operate.

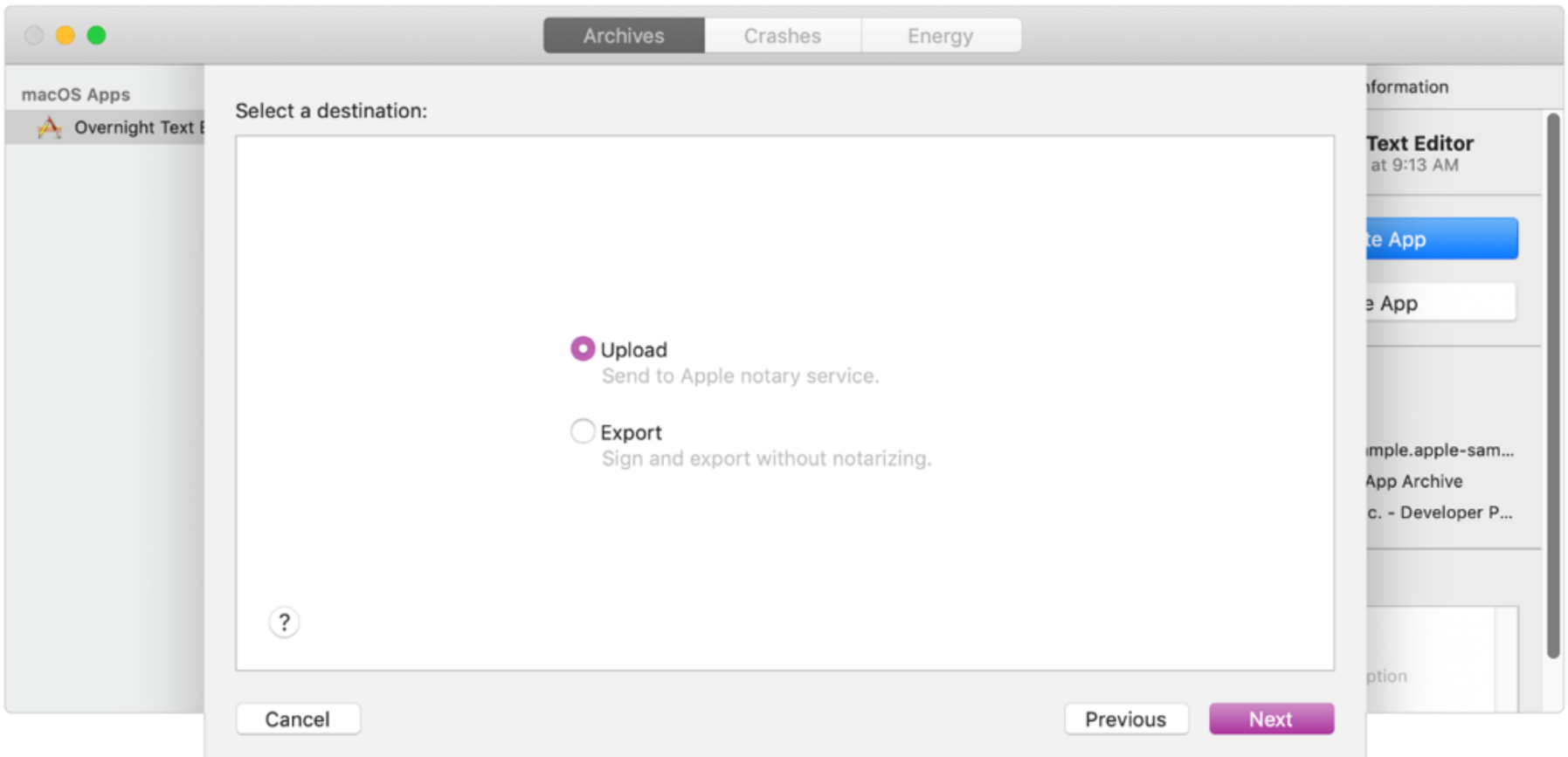
For a complete list of hardened runtime entitlements, see Hardened Runtime Entitlements. For information about usage strings, see Accessing Protected Resources.

> **Important**
>
> On macOS 10.14.x, for executables using the hardened runtime, PDEs load only if the host executable has the `Disable Library Validation Entitlement`.

## Notarize Your App Automatically as Part of the Distribution Process

Before distributing your app directly to customers, your Account Holder must sign the app with your Developer ID. Xcode's Organizer window includes a workflow for generating a distributable version of your app. In Xcode 10 and later, this workflow includes an option to notarize your app automatically. To notarize your app using this workflow, do the following:

1. Open your Xcode project.
2. Create an archive of your app.
3. Open Xcode's Organizer window.
4. In the Archives tab, select the archive you created.
5. Click Distribute App to view the distribution options.
6. Choose Developer ID for your method of distribution.
7. Click Next.
8. Choose Upload to send your archive to the Apple notary service.
9. Click Next.




When you click Next, Xcode uploads your archive to the notary service. When the upload is complete, the notary service begins the scanning process, which usually takes less than an hour. While the notary service scans your software, you can continue to prepare your archive for distribution. For example, you can export the archive and perform any final testing that you require prior to making your software available to customers.

When the notarization process finishes, Xcode downloads the ticket and staples it to your archive. At that point, export your archive again to receive a distributable version of your software that includes the notary ticket.

For more information about how to use the Xcode UI to upload your software, see Upload a macOS app to be notarized.

## Notarize Your Preexisting Software

Notarizing your preexisting software lets Gatekeeper warn users when they try to run it. It also helps the notary service distinguish your legitimate software from variants that have been tampered with. You can notarize an existing disk image, installer package, or ZIP archive containing your app.

To notarize your preexisting software, do the following:

1. Make Xcode 10 your active Xcode installation. (If you're not sure whether Xcode 10 is the active installation, use the `xcode-select` command-line to make it active. For information about how to use this tool, see the man page for it, as described in Reading UNIX Manual Pages.)
2. Upload your software to the Apple notary service, as described in Upload Your App to the Notarization Service.
3. Staple the returned ticket to your existing software, as described in Staple the Ticket to Your Distribution.

> **Note**
>
> You don’t need to rebuild or re-sign your software before submitting it for notarization, but you must use Xcode 10 to perform the notarization steps. Submit everything you’ve previously released, as well as your most recent version, to protect users who continue to

use older versions of your software.

For tips on how to resolve issues that can occur during notarization, see Resolving Common Notarization Issues.

## Add a Notarization Step to Your Build Scripts

If you use an automated build system, you can integrate the notarization process into your existing build scripts. The `altool` and `stapler` command-line tools (included with Xcode) allow you to upload your software to the Apple notary service, and to staple the resulting ticket to your executable.

For information about how to incorporate notarization into your custom build scripts, see Customizing the Notarization Workflow.

## Topics

### Notarization

- Customizing the Notarization Workflow

  Notarize your app from the command line to handle special distribution cases.

- Resolving Common Notarization Issues

  Handle common problems reported in the notarization log file, or that arise during ticket stapling.

## See Also

## Secure Code

### Code Signing Services

Examine and validate signed code running on the system.

### Preparing Your App to Work with Pointer Authentication

Test your app against the arm64e architecture to ensure that it works seamlessly with enhanced security features.

### App Sandbox Entitlements

Manage access to system resources and user data in macOS apps to contain damage if an app becomes compromised.

### Hardened Runtime Entitlements

Manage security protections and resource access for your macOS apps.