

How to Sign and Notarize a New Build for the Mac

I recently learned how to sign and notarize our build for the Mac Mini and will be documenting both the one-time setup that I had to do, as well as the steps that must be taken with each new build.

Many of these steps began as outlined in [this blog](#), though with extensive changes along the way. If all you're looking for is how to make the next build for the Mac, please skip down to the "Notarize and Staple a Build" section.

Initial Setup

All of this setup has already occurred, and you probably won't ever need to do any of it again. These instructions are here just so that we can reproduce these actions if that is ever made necessary.

Update the Keychain certificates

In order to sign and notarize our build file, we're going to need to use two Keychain Certificates.

1. Open the Keychain Access app, open the "All Items" tab
2. Search for "J82X76G6Y8" (our "team identifier")
3. Among the results you are looking for these two certificates
 - Developer ID Application: eFilecabinet, Inc. (J82X76G6Y8)
 - Developer ID Installer: eFilecabinet, Inc. (J82X76G6Y8)
4. You need to verify that both of these are still current (not expired). You also need to make sure that there aren't any expired duplicates of these certificates. If there are, then right-click on the expired duplicates and delete them.

Configure Visual Studio to use the Hardened Runtime

For our application to be notarized, it is also necessary for it to be compiled using the Hardened Runtime.

1. Right-click on the RubexClient project in the Solution side-panel of Visual Studio and select Properties. Then open the Build->Mac Signing view
 - Check "Enable Hardened Runtime"

- In the "Custom entitlements" field enter: Entitlements.plist
- Open the Entitlements.plist and add a necessary permission.
 - If you are doing this in Visual Studio, click "Add new entry" and then enter the following in each column
 - Property: com.apple.security.cs.allow-jit
 - Type: Boolean
 - Value: Yes
 - If you are doing this in a text editor, inside of the <dict></dict> section add the following:

```
<key>com.apple.security.cs.allow-jit</key>
<true/>
```

Configure Visual Studio to sign the .dmg build

Now Visual Studio needs to be configured to sign the package as part of the build process. For this you will need the name of the *Installer* certificate from the last step, as well as the identifier for our application (currently "com.efilecabinet.rubex").

1. In the Visual Studio project find and open the spec.json file
2. On the same level as the title, background, icon-size, contents, and window, add a new section as follows (with the current signing-identity and identifier as defined above):

```
"code-sign": {
  "signing-identity": "Developer ID Application: eFilecabinet, Inc. (J82X76G6Y8)",
  "identifier": "com.efilecabinet.rubex"
}
```

3. Next, right-click on the RubexClient project in the Solution side-panel of Visual Studio and select Properties. Then open the Build->Mac Signing view
 - Check "Sign the application bundle" at the top
 - For the "Identity" directly beneath that checkbox select "Developer ID: eFilecabinet, Inc. (J82X76G6Y8)"
 - Check "Sign the installer package" at the bottom
 - For the "Identity" directly beneath that checkbox select "Developer ID Installer: eFilecabinet, Inc. (J82X76G6Y8)"
4. Now when you build the project, towards the end of the process you should see a line that says:
 - [20/21] Signing image... [OK]

Generate Application-Specific Password for notarytool

Now that we have a signed build using the hardened runtime, we can configure our notarization tools. For this, we use the command-line utility "notarytool." First, though, we must authenticate the tool with an application-specific password. The current application-password is stored in LastPass, under the Notes section of the Mobile Dev Apple Account item. If you ever need to create

a new password, then:

1. Go to the [Apple ID](#) website
2. Enter the sign-in credentials under the Mobile Dev Apple Account item in LastPass
3. You will probably need to enter a second-factor code. For this you'll want to be signed into the Development Mac Mini so you can see the sign-in attempt alert, approve it, and get the 6-digit code to complete your sign in
4. Under the Sign-in and Security section select App-Specific Passwords
5. Press the + button, enter a memorable name, and press the Create button
6. You will *ONLY* be able to see the password one time on the next screen, so be sure to save that somewhere for reference

Authorize notarytool

Now we can use this password to authorize the notarytool.

1. In a terminal on the mac run:

```
xcrun notarytool store-credentials --apple-id "mobiledev@efilecabinet.net" --team-id "J82X76G6Y8"
```

Note the 10-digit team id at the end that we got from our certificates. Also, the Apple ID is the same one that we used to sign into the mobile dev account.

2. You will then be asked for a "Profile name." Choose something meaningful and write it down for future use. Our current profile name is "notary-credentials"
3. You will then be asked for the password that goes with your account (mobiledev@efilecabinet.net). This does *not* mean the password that you used to sign into the Apple Id account. It means the Application-Specific Password that you generated at the end of the last section.
4. It should process for a minute and then tell you that it has validated your credentials and saved them to Keychain. You will now be able to reference that Keychain item using the "Profile name" that you provided on step 2.

In the following section it refers to the keychain profile as "notary-credentials." Obviously, you would change that to whatever profile name you set in this section.

Notarize and Staple a Build

1. Increment the "Bundle versions string" in the Info.plist of the Visual Studio project
2. Ensure that your Visual Studio project is set to build for Release
3. Select Build->Rebuild Solution

4. Open a terminal and navigate to the build location:
dev/UtopiaClientApplications/UtopiaMacDesktopClient/RubexClient/bin/Release/
5. Notarize the built .dmg file with the following command:

```
xcrun notarytool submit RubexInstaller.dmg --keychain-profile "notary-credentials" --wait
```

It will run for a while, just wait for it to complete with a success message.

6. Now you need to "staple" that notarization to the .dmg file. You do that with this command:

```
xcrun stapler staple RubexInstaller.dmg
```

7. Finally, check that everything is correct by running this command

```
spctl --assess -vv --type install RubexInstaller.dmg
```

This should give a response message the .dmg file is "accepted," that the source is a "Notarized Developer ID," and the origin should be our developer id.

You may now give the signed and notarized RubexInstaller.dmg to QA for testing and then SecDevOps for release.

Revision #5

Created 22 March 2023 22:50:24 by Abe Austin

Updated 1 May 2023 18:34:19 by Abe Austin