Using Apple’s "notarytool" to Notarize Applications

By Brianna Rentfrow, Technical Services Engineer, 4D Inc.

Technical Note 25-01

Table of Contents

Abstract

Cybersecurity is a major concern of both users and companies these days. Apple's solution to making sure that apps outside of the Apple App Store are safe to use involves a process known as "notarization". Notarization allows developers to quickly submit their applications to Apple to ensure they are safe and secure. Apple has now released a new tool called "notarytool" that streamlines the process and makes notarizing applications easier for developers than ever before.

Introduction

Apple's "notarytool" is a fast way for developers to allow Apple to approve their applications, ensuring that they are safe and secure. This technical note will cover how to utilize this tool to streamline the notarization process. Specifically, it will cover first-time set up, the new one-command way to notarize applications, and stapling the applications afterward.

Notarization is required when the 4D Application is built from a MacOS Machine, will be downloaded and opened by another MacOS machine, or is deployed over the internet.

Requirements

- MacOS 11.3 or newer
- 4D version 18 or newer
- Xcode version 13 or newer
- An Apple Developer account
    - Developer ID Application Certificate
    - App Specific Password

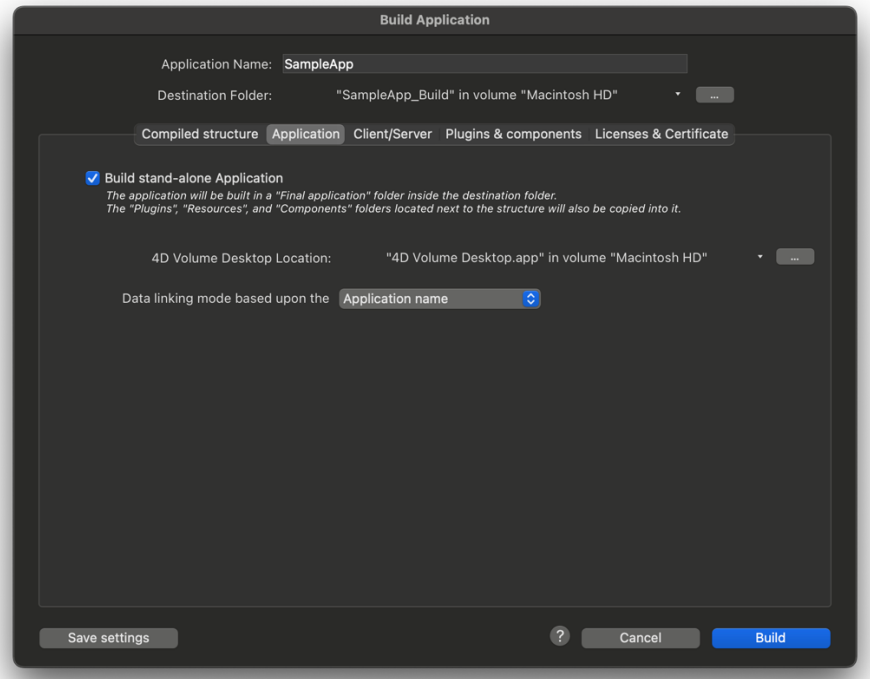
Getting the App Ready for Notarization

Before the developer's application can be notarized, it must be compiled and built.

The first step is to make sure the project compiles. The project can be compiled by navigating to the top and under "Design", clicking "Compiler…". Press the "Compile" button in this new window and if the operation is successful, the program is ready to be built.

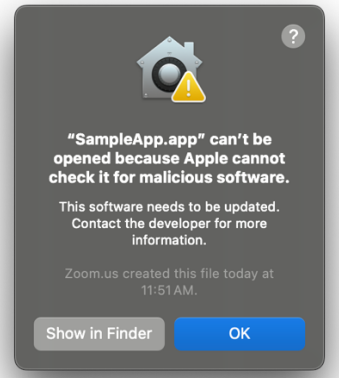
To build the application, one can navigate back to "Design" and select "Build Application".

- Select the checkbox for what type of file the developer wants to build. For example, Application > Build Standalone Application. Make sure anything needed in this area is filled out. For applications it will need the developer to select the 4D Volume Desktop location.



- Head to the apple developer account and either create a new certificate or download one of the ones the developer already has. Double click to add it to the machine's keychain. Copy the exact full name of the certificate.
- Go back to the build application window and go to the tab Licenses and Certificate. Check off the "Sign Application" checkbox and paste the certificate name into the box at the bottom.
- From there the developer should be able to build the application

There should now be an application file. At this point if a user downloads the .app file, it will not open and instead give the message: ""appname.app" can't be opened because Apple cannot check it for malicious software."



If the system is on Mac OS15 Sequoia, the message will be slightly different, instead reading ""appname.app" Not Opened Apple could not verify "appname.app" is free of malware that may harm your Mac or compromise your privacy." It will give the options to either close out of the window with "Done" or "Move to Trash".

First Time Setup

To make the notarization process as quick as possible, some setup is required. Specifically, this section will cover setting up a profile on the system's keychain that may be called every time the developer wants to notarize an application.

The next step is to set up the Apple developer profile on the system's keychain. This command will only need to be set up the one time and anytime the developer needs to notarize an app, it will be as easy as simply referring the notarytool to this keychain profile rather than needing to put in the app-specific password and log in information every time.

- Within the command line, insert the following command.

```
xcrun notarytool store-credentials --apple-id "EMAIL" --team-id "TEAMID"
```

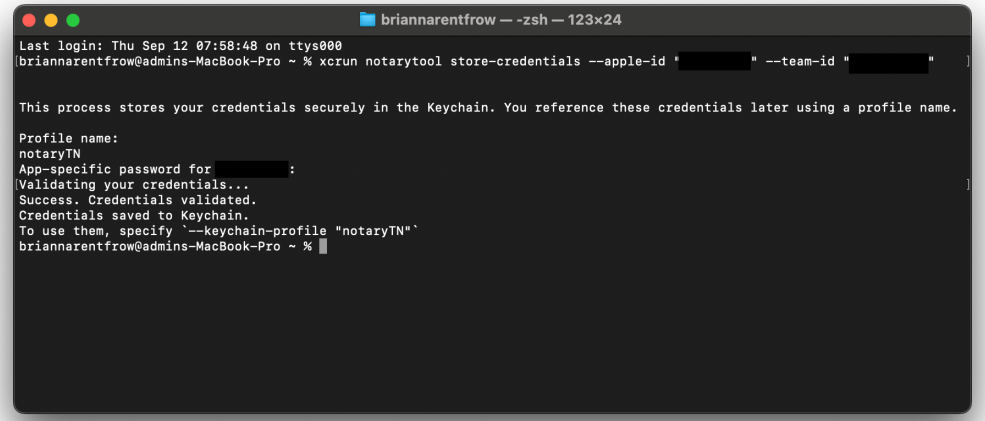
Note: *"Email" is replaced with the apple developer email and "TEAMID" is replaced with the team id that can be found on the* ***apple developer account****.*

- Pressing enter will allow the developer to put in a name for the profile. This is the name the developer will refer to every time. In this example, the name "notaryTN" is used.
- Next it will ask for an app-specific password which the developer can paste from their **Apple account**.
- The following steps show how to obtain an App-Specific Password:
    - Navigate to https://account.apple.com/ and login
    - On the page that opens, click on the box labeled "App-Specific Passwords".
    - From there, the developer can revoke and add new App-Specific Passwords.

Note: *App-Specific Passwords cannot be viewed from the website, only revoked or added.*

The profile is now set up for use. It can be accessed from now on with the command

```
--keychain-profile "notaryTN"
```



Adding Files to an Application before Notarization

Note: *This section may not apply to all applications. It only applies if the application requires additional files that cannot be placed in locations that 4D automatically signs.*

Some applications may require a file or folder to be added that was not part of the regular format, thus not included in the initial build. To do so, the developer may complete the following steps before moving onto the notarization instructions. If working with an application that does not require additional file, these steps may be disregarded.

The first step is to Compile and Build the application as normal.

Next, add the file where it is needed in the project.

Then create a brand new 4D Project. Within this project's Resources folder, create a new shell script file (.sh) with the following contents:

```
IFS=$'\n'; set -f
	for f in $(find "${product_path}" -type f \( -perm +111 -o -name "*.dylib" -o -name "*.js" -o -name "*.json" -o -name "*.html" -o -name "*.so" \) | sort -r )
	do xattr -c "${f}";
	codesign --verbose --options=runtime --timestamp --force --sign "${sign}" --keychain ${keychain} --entitlements "${entitlements_path}" "${f}";
	done
	for f in $(find "${product_path}" -type d \( -name "*.app" -o -name "*.framework" -o -name "*.bundle"  -o -name "*.plugin" -o -name "*.kext" \) | sort -r )
	do xattr -c "${f}";
	codesign --verbose --options=runtime --timestamp --force --sign "${sign}" --keychain ${keychain} --entitlements "${entitlements_path}" "${f}";
	done
	unset IFS; set +f
```

If the developer already has an entitlement file, insert it inside of the new project's Resources. Otherwise, the 4D Entitlements file can be used.

Go to 4D.app and right click > Show Package Contents. Go to the Resources folder and find 4D.entitlements. This file can be copied into the new project's resources.

Create a 4D method inside the newly created project with the following contents. Within the variable $sign, feel free to insert the certificate name that is in the computer's keychain. Please ensure the entitlements matches the name of the one in the Resources file. This code assumes the app is on the developer's desktop. Also ensure the script file name matches. This example implies the script is named "script.sh".

```
//name of your certificate
$sign:="CERTNAME (CERTNUMBERS)"
//your keychain profile
$keychain:="notaryTN"
//Location of the .app file (this example is an app called appname.app that is on the desktop)
$product_path:=Folder(fk desktop folder).folder("appname.app").path
//entitlements file. I use the one from 4d
$entitlements_path:=File(File("/RESOURCES/4D.entitlements").platformPath; fk platform path).path

SET ENVIRONMENT VARIABLE("sign"; $sign)
SET ENVIRONMENT VARIABLE("keychain"; $keychain)
SET ENVIRONMENT VARIABLE("product_path"; $product_path)
SET ENVIRONMENT VARIABLE("entitlements_path"; $entitlements_path)

$script:=File(File("/RESOURCES/script.sh").platformPath; fk platform path).path

var $in; $out; $err : Text
LAUNCH EXTERNAL PROCESS("bash "+$script; $in; $out; $err)

ALERT("Done")
```

Once this is run and the alert says "Done", the application should be successfully signed and ready for notarization and stapling.

Notarization

Notarizing Applications and Components

Now that the profile is set up, the following steps for notarization will be simple to do each time the developer needs to notarize an application. First, zip the application. To notarize the application, the following command can be executed in the terminal.

```
xcrun notarytool submit "[ZIP FILE PATH]" --keychain-profile "notaryTN" --wait
```

Note: *"[ZIP FILE PATH]" is to be replaced with the path to the zip that contains the signed application. Do not include the brackets. The file can be dragged to the terminal window to get the correct path.*

Once this command is entered, the developer will see an upload progress percentage.

A status "In progress…" adding more dots to let the developer know that the process is still going. This will show until the notarization process is completed with a status whether it is accepted or not.

Notarizing Compiled Databases

Compiled databases require an extra step to be properly signed. The file "lib4d-arm64.dylib" does not get signed properly by 4D when the application is built. To get around this, the developer must sign this library manually. The library can be found in the "Libraries" folder of the Compiled Database. The following command can be put in the terminal.

```
codesign -s [IDENTITY] -o library [Path to lib4d-arm64.dylib]
```

A codesign identity, which is connected to the developer account and the certificate, is required. To find the codesign identities on the machine, run this command in the terminal:

```
security find-identity -v -p codesigning
```

The identity is just the generated alphanumeric string before the name of the identity.

After this, the compiled database should be able to be signed normally the same as applications and components. Simply re-zip the entire folder and submit that for notarization.

Notarization Log Files

It is recommended to check the log whether the notarization has been accepted or not which can be done using the following command with the ID that it provides.

```
xcrun notarytool log "[ID_RECEIVED]" --keychain-profile "notaryTN" developer_log.json
```

If the application has not been accepted, the log should contain the reason as to why. The log may also contain warnings the user may want to look out for.

Note: *The log can be named whatever the developer prefers, this example having it be named "developer_log.json". This is the path so the developer may choose to place the .json file wherever is easily accessed.*

Stapling Applications

The last step to ensuring the application is ready is stapling. When an application is notarized, a ticket saying that it has been notarized is generated and stored on Apple's servers. Developers can 'staple' this ticket directly to the application so that Gatekeeper doesn't need to search online for this ticket, ensuring the ticket can be found regardless of internet connection.

Note: *Some types of applications cannot be stapled. For example, 4D Components which are the file format ".4dbase" are unable to be stapled. The only valid file formats for stapling are ".app", ".dmg", and ".pkg".*

- Once the application is notarized, unzip the zip file.
- Run the following command on your .app file

```
xcrun stapler staple [PATH]
```

- Upon success, the message "The staple and validate action worked!" will appear. Now the app can be run the first time it is downloaded without requiring access to the internet.

Conclusion

This technical note described how to sign, notarize, and staple applications. Due to Apple's security measures, the company requires their systems to check all applications to make sure they are safe and secure. This information should allow developers to deploy applications meant for MacOS machines to run.