Distributing Your Xojo App Using Xcode

Jim McKay
Owner
piDog Software

Benefits

- Simple CodeSigning and Entitlements Setup
- Built-in Notarization
- Export for distribution or submit to the AppStore!
- Crash Logs show up in your Organizer.
- Profile your code with Instruments.app
- When Apple changes something…

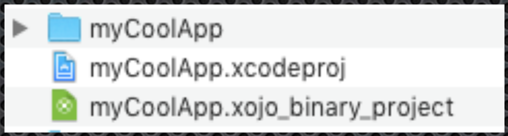
Requirements

- Xojo
- Xcode
- Apple Developer Account
- Signing Certificates
- Register Bundle IDs with Apple
- Auto Increment Version in Xojo!

Initial Setup

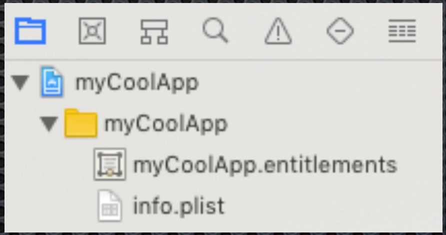
1. Create a new Xcode project.
2. Select the Cocoa App template.
3. Enter your project info.
4. Save the new project to the Desktop.
5. Close the Xcode project.
6. Drop the contents of the folder next to your Xojo project file.
7. Delete the Xcode project folder.

New Friends


myCoolApp
myCoolApp.xcodeproj
myCoolApp.xojo_binary_project

Clean Up Xcode

1. Open the .xcodeproj file
2. Delete everything in the folder in the navigator except the entitlements and plist.
3. Things should look pretty simple.


myCoolApp
myCoolApp
myCoolApp.entitlements
info.plist

Products Location

(Optional)

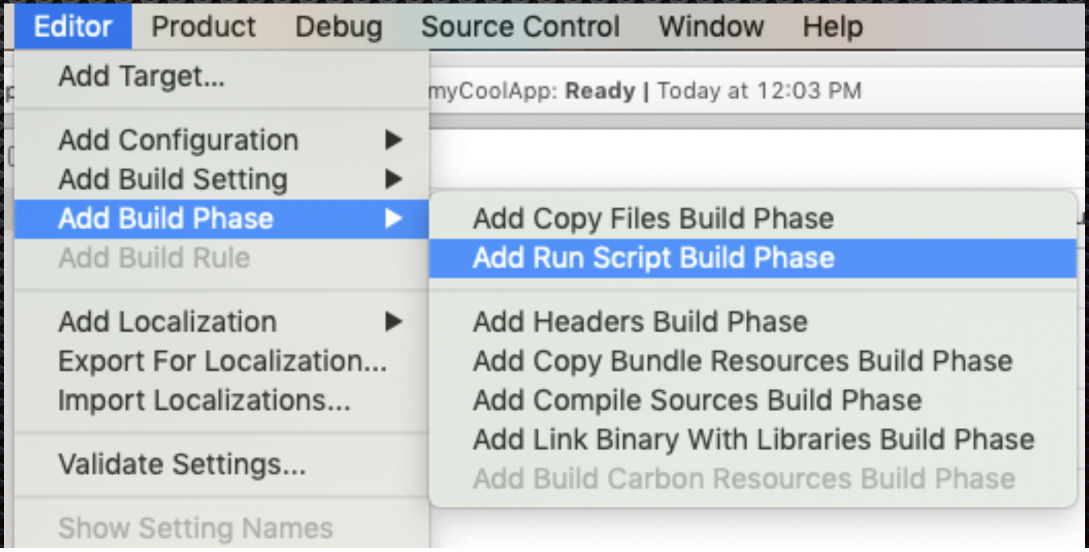
1. Select the **File->Project Settings** menu
2. Click the **Advanced** button
3. Select **Custom** / **Absolute** for the Build Location
4. Click the folder icon next to **Products** and navigate to the **OS X 64** folder in your Xojo Builds folder.
5. Create a new folder. I'll call it Xcode.
6. You can place the other folders where you like, but selecting the same folder is fine.

Codesign/Entitlements

1. With your app target selected, go to **Build Settings**.
2. In the search box, enter “signing flags”
3. Add **—deep** and **—force** to the flags.
4. Go to the Capabilities section and set any sandboxing/entitlements and hardened runtime setting you require.
5. Back in the **General** section, be sure your team is selected under **signing**.

Build Setup

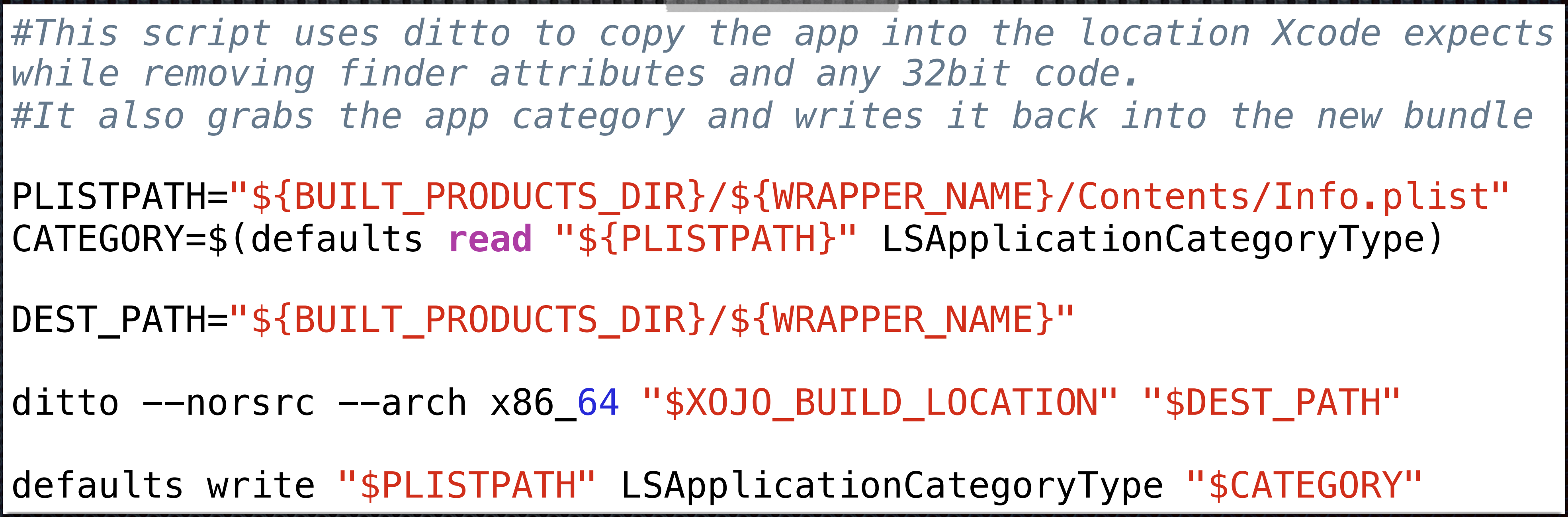
1. Go to the **Build Phases** section.
2. Clear it out. Remove the Compile/Link/Copy steps.
3. Ignore the grim warnings.
4. Select **Editor->Add Build Phase->Add Run Script Build Phase**.


Editor
Product
Debug
Source Control
Window
Help
myCoolApp: Ready | Today at 12:03 PM
Add Target...
Add Configuration
Add Build Setting
Add Build Phase
Add Build Rule
Add Localization
Export For Localization...
Import Localizations...
Validate Settings...
Show Setting Names
Add Copy Files Build Phase
Add Run Script Build Phase
Add Headers Build Phase
Add Copy Bundle Resources Build Phase
Add Compile Sources Build Phase
Add Link Binary With Libraries Build Phase
Add Build Carbon Resources Build Phase

Add the Script

This script uses ditto to copy the app into the location Xcode expects while removing finder attributes and any 32bit code.

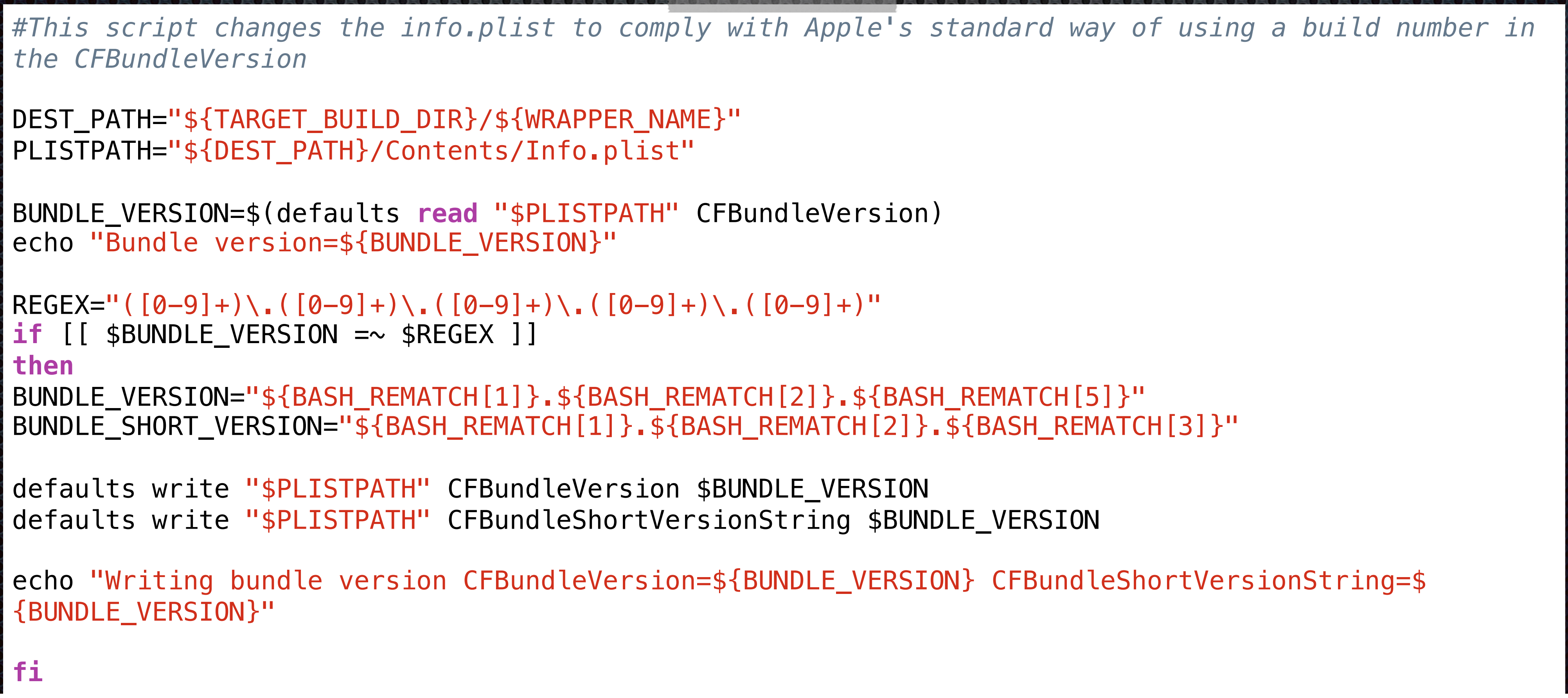
It also grabs the app category and writes it back to the new bundle.



```
#This script uses ditto to copy the app into the location Xcode expects
while removing finder attributes and any 32bit code.
#It also grabs the app category and writes it back into the new bundle

PLISTPATH="${BUILT_PRODUCTS_DIR}/${WRAPPER_NAME}/Contents/Info.plist"
CATEGORY=$(defaults read "${PLISTPATH}" LSApplicationCategoryType)

DEST_PATH="${BUILT_PRODUCTS_DIR}/${WRAPPER_NAME}"

ditto --norsrc --arch x86_64 "$XOJO_BUILD_LOCATION" "$DEST_PATH"

defaults write "$PLISTPATH" LSApplicationCategoryType "$CATEGORY"
```

Optional Script

This script updates the info.plist to comply with Apple's standard way of using a build number in the CFBundleVersion



```
#This script changes the info.plist to comply with Apple's standard way of using a build number in
the CFBundleVersion

DEST_PATH="${TARGET_BUILD_DIR}/${WRAPPER_NAME}"
PLISTPATH="${DEST_PATH}/Contents/Info.plist"

BUNDLE_VERSION=$(defaults read "$PLISTPATH" CFBundleVersion)
echo "Bundle version=${BUNDLE_VERSION}"

REGEX="([0-9]+)\.([0-9]+)\.([0-9]+)\.([0-9]+)\.([0-9]+)"
if [[ $BUNDLE_VERSION =~ $REGEX ]]
then
BUNDLE_VERSION="${BASH_REMATCH[1]}.${BASH_REMATCH[2]}.${BASH_REMATCH[5]}"
BUNDLE_SHORT_VERSION="${BASH_REMATCH[1]}.${BASH_REMATCH[2]}.${BASH_REMATCH[3]}"

defaults write "$PLISTPATH" CFBundleVersion $BUNDLE_VERSION
defaults write "$PLISTPATH" CFBundleShortVersionString $BUNDLE_VERSION

echo "Writing bundle version CFBundleVersion=${BUNDLE_VERSION} CFBundleShortVersionString=$
{BUNDLE_VERSION}"

fi
```

Back to Xojo

We'll automate the build processes with a script!

Add this as a post-build script for release and builds will automatically show up in the origanizer

```
dim setAppPath as string = "export
XOJO_BUILD_LOCATION="""+CurrentBuildLocationNative+"/"+CurrentBuildAppName+".app"";"

call doshellcommand(setAppPath + "cd ""$PROJECT_PATH""; xcodebuild clean > build_log.txt")
call doshellcommand(setAppPath + "cd ""$PROJECT_PATH""; xcodebuild >> build_log.txt")
call doshellcommand(setAppPath + "cd ""$PROJECT_PATH""; xcodebuild archive >> build_log.txt")
```

Archive and Organize

Select **Window->Organizer** to see the result!

Where Now?

From the Organizer you can:

- Submit to the Mac App Store
- Export for direct distribution
- Verify with Apple
- View Crash logs submitted by users

Running in the Sand(box)

We need a “Scheme” to be able to run a script **after** the bundle is built and signed.

1. Go to **Product->Scheme->New Scheme**
2. We’ll call it **Debug**
3. Select our myCoolAppDebug product.

Add the Script

1. Select the new Scheme and edit scheme
2. Add a **Run Script** to **Post Actions** of **Build**.
3. This one's pretty simple…

```
BUILD_PATH="$BUILT_PRODUCTS_DIR/$WRAPPER_NAME"

rm -r "$XOJO_BUILD_LOCATION"
mv "$BUILD_PATH" "$XOJO_BUILD_LOCATION"
```

Allow Debugging

For the debug app to connect to Xojo, it needs to open a connection.

1. Add a new Entitlements file to the Xcode Project

   Copy/Paste in the Finder is the simplest way.

2. Go to **Capabilities**.
3. Turn on **Sandbox**.
4. Check **Outgoing Connections**
5. Copy those settings to the new entitlements file.
6. Set the new Entitlements for Debug in **Project Settings**

Add a script

Now we'll need a script that puts the debug app back in place for Xojo to be able run it.

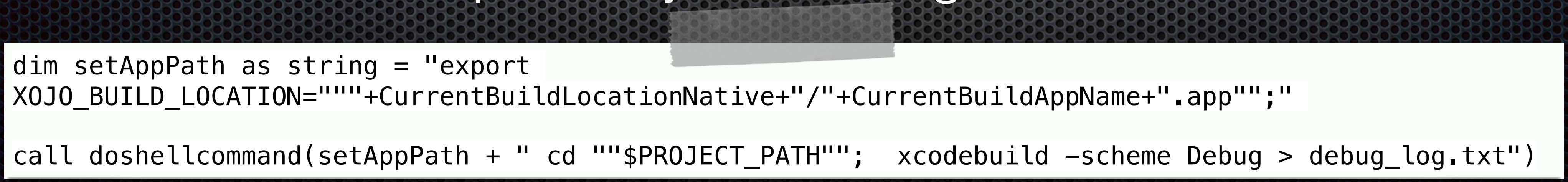
1. Go to **Product->Scheme->Edit Scheme**
2. Go to the **Build->Post-actions** section.
3. Click the **+** and select **New Run Script Action**.
4. Add a script to move the app back for debugging.

```
BUILD_PATH="$BUILT_PRODUCTS_DIR/$WRAPPER_NAME"

rm -r "$XOJO_BUILD_LOCATION"
mv "$BUILD_PATH" "$XOJO_BUILD_LOCATION"
```

Back to Xojo (again)

Ask Xcode to build our debug app with the new scheme.

Add a Build Script to Xojo for debug.



```
dim setAppPath as string = "export 
XOJO_BUILD_LOCATION="""+CurrentBuildLocationNative+"/"+CurrentBuildAppName+".app"";"

call doshellcommand(setAppPath + " cd ""$PROJECT_PATH"";  xcodebuild -scheme Debug > debug_log.txt")
```

Run the app in Xojo and check the debug app for entitlements!

What about iOS?

- Xojo already handles iOS!
- Dump ApplicationLoader!
- Automatic certificate handling.
- Automate Archiving.
- Keep up with Apple!

Just a few differences…

- iOS bundles have their plist at the top level.
- Stripping code is not needed and can break things.

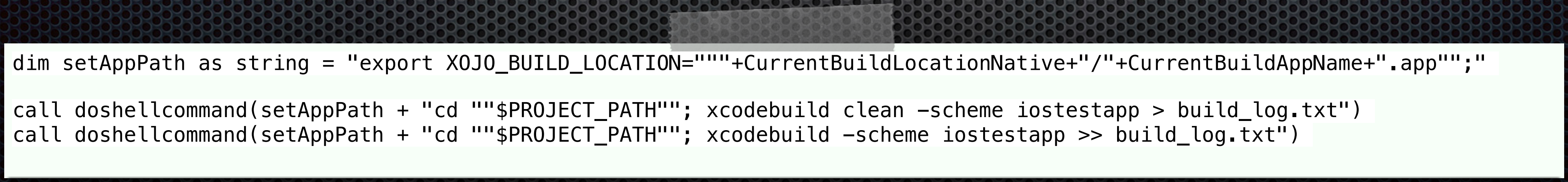
Add the Script

New Build Script for iOS

```
#This script uses ditto to copy the app into the
location Xcode expects while removing finder
attributes.

DEST_PATH="${BUILT_PRODUCTS_DIR}/${WRAPPER_NAME}"

ditto --norsrc "$XOJO_BUILD_LOCATION" "$DEST_PATH"
```

Add a Build Script

Have Xojo run the Xcode clean / build / archive for us.



```
dim setAppPath as string = "export XOJO_BUILD_LOCATION="""+CurrentBuildLocationNative+"/"+CurrentBuildAppName+".app"";"

call doshellcommand(setAppPath + "cd ""$PROJECT_PATH""; xcodebuild clean -scheme iostestapp > build_log.txt")
call doshellcommand(setAppPath + "cd ""$PROJECT_PATH""; xcodebuild -scheme iostestapp >> build_log.txt")
```

Run the app in Xojo and check the organizer in Xcode!

Xcode Project Templates

Visit:

https://bitbucket.org/pidog/xcode-templates-for-xojo

Q & A

Jim McKay

jim@pidog.com

Give us feedback on this session in the XDC app!