Installing and Building Apps on a Mac

Keyboard App Builder:
Installing and Building Apps on a Mac

Last updated: 25 March 2025

The latest version is available at
http://software.sil.org/keyboardappbuilder/resources/

and on the Help menu of Keyboard App Builder.

Contents

1. Introduction

This document provides information on how to install Keyboard App Builder and build apps on an Apple macOS system. Creating an Android app on a Mac is essentially the same process as it is for Windows or Linux.

Keyboard App Builder only builds apps for Android. There is currently no iOS app.

2. Installing Keyboard App Builder

To install the Keyboard App Builder program files:

1. Download the current Mac installer file (dmg) from the KAB website:

   http://software.sil.org/keyboardappbuilder/download/

2. Double click on the file within **Finder** to open the disk image that contains the Keyboard App Builder application.

3. Copy the **Keyboard App Builder** application to your **Application** folder. This can be done by dragging the Keyboard App Builder icon from the disk image window to the shortcut of the Applications folder in the same window.

3. Installing Prerequisites for Android

If you want to build Android apps, you need to install the following components on your computer:

1. Java Development Kit (JDK)
2. Android Software Development Kit (SDK)

3.1. Java Development Kit (JDK)

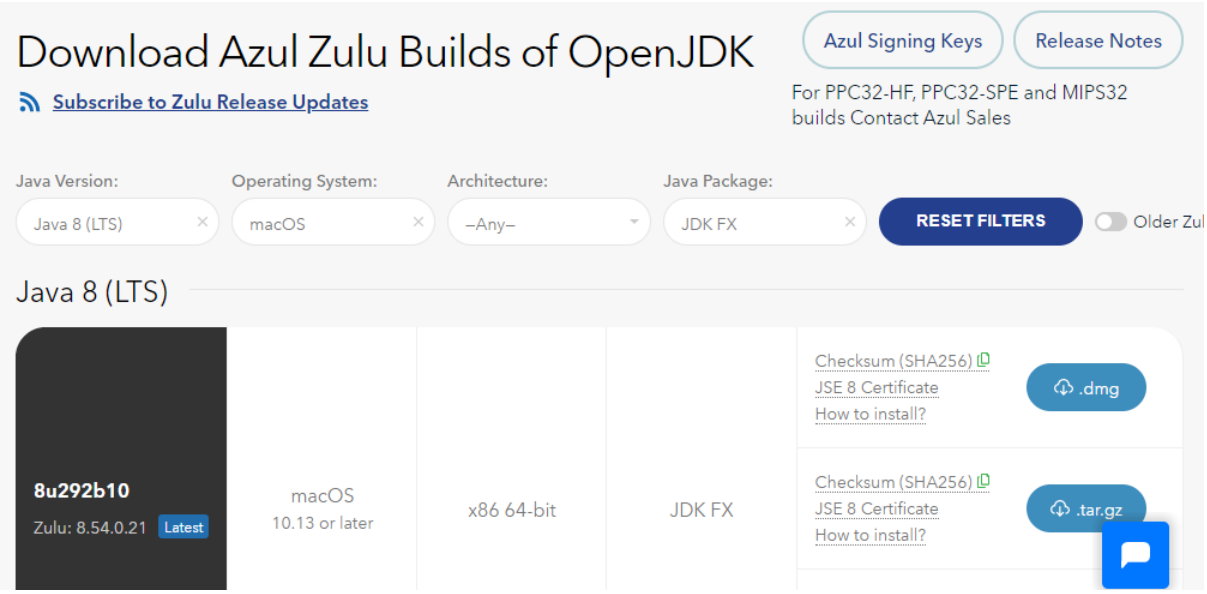
You will need version 8 of the Java Development Kit (JDK) to build apps. We recommend you use Zulu, which is a free distribution of OpenJDK from Azul.

1. Go to the **Download Zulu Builds of OpenJDK** website:

   https://www.azul.com/downloads/?version=java-8-lts&os=macos&package=jdk-fx

   There are many downloads on this page, but the above link will filter the ones you see (Java Version: Java 8 LTS; Operating System: MacOS; Java Package: Java FX).

2. Scroll down the page until you see the downloads under the heading **Download Azul Zulu Builds of OpenJDK**:



3. You have a choice between two different architectures: **x86 64-bit** and **ARM 64-bit**. You can find the processor type of your machine by clicking on the Apple symbol at the top left of your screen and selecting **About This Mac**. If you have one of the newer Macs with an M1 chip, choose ARM, otherwise you will need x86 (Intel).

   Once you have identified your device architecture, you have a choice between a **dmg** file, a **tar.gz** file and a **zip** file. **Download the .dmg file** since it comes with its own installer program.

   The file you download will have a filename something like this:

   zulu8.54.0.21-ca-fx-jdk8.0.292-macosx_x64.msi

4. **Double-click the file in Finder** and follow the instructions in the installation wizard to install it. By default, the installer will install the JDK to the following folder:

   /Library/Java/JavaVirtualMachines/zulu-8.jdk/Contents/Home

> **Important**: If you change the JDK install folder to something other than the default folder, you will need to remember the location of the folder so you can tell Keyboard App Builder where to find the JDK.

3.2. Installing Android Software Development Kit (SDK)

The Android Software Development Kit (SDK) is needed for building Android apps. There are two ways of installing the Android SDK:

1. **Online: Download the Android SDK packages from the internet:**
   Use the Android SDK Installation wizard to download and install the command line tools and three additional packages. This method will require an internet connection.

   See 3.2.1 for more details.

2. **Offline: Copy the Android SDK files from someone else:**
   If you know someone who has already downloaded and installed the Android SDK, you can copy all the files from them.

   This method is especially useful in a training workshop where several people need to install the SDK but have limited internet bandwidth.

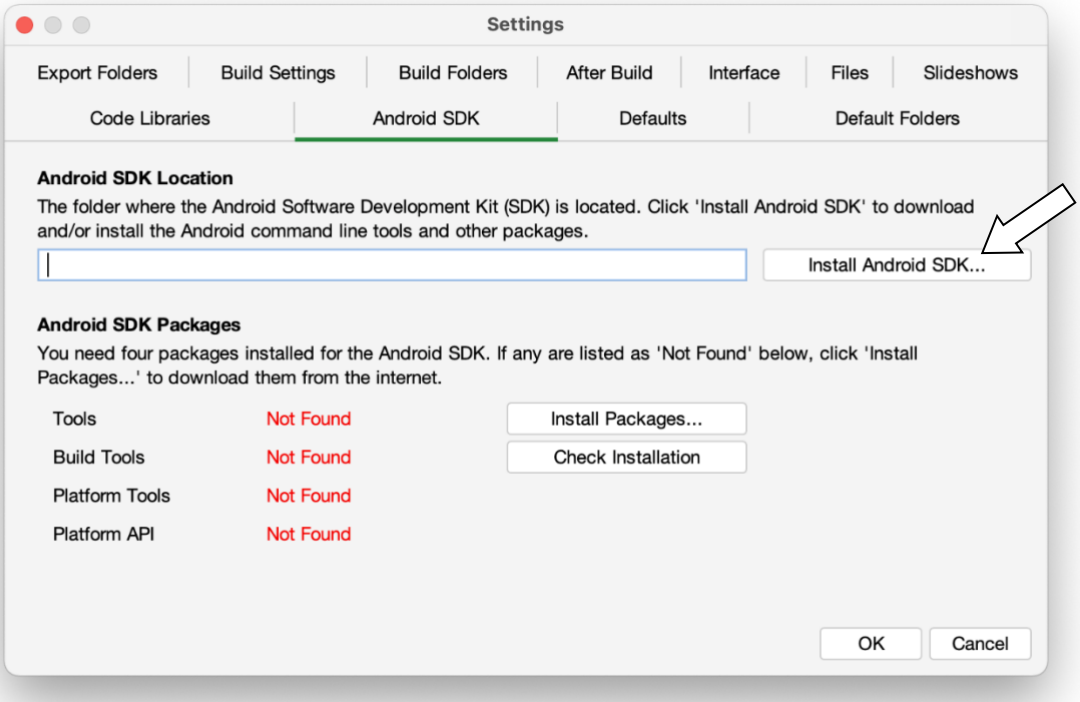
   See 3.2.2 for more details.

3.2.1. Downloading the Android SDK packages from the internet

To install the Android SDK from the internet:

1. Launch **Keyboard App Builder**.
2. Select **Keyboard App Builder** ➢ **Preferences** from the main menu.
3. Go to the **Android SDK** tab, which is the second tab.

4. Click the **Install Android SDK** button.



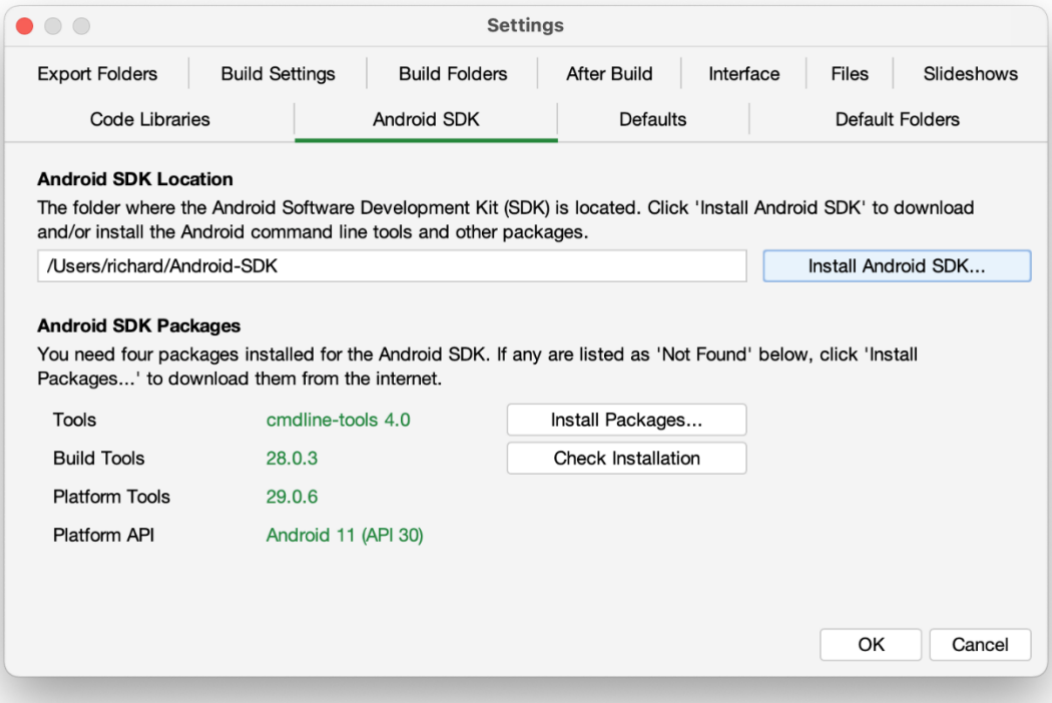
5. Follow the instructions on each page of the **Install Android SDK** wizard to download each of the Android SDK packages and install them.

   When you are asked to specify a target folder, a good place is:
   /Users/*your-name*/Android-SDK.

   Four packages will be downloaded and installed:

   - Command line tools,
   - Build Tools,
   - Platform Tools, and
   - Platform API.

   If the installation was successful, you will see the version numbers displayed in green.



If any of the Build Tools, Platform Tools or Platform API is listed as “Not Found” (displayed in red), click the **Install Packages** button to install them.

Click the **Check Installation** button to confirm that all the packages have been installed correctly.

3.2.2. Copying the Android SDK files from someone else

If you know someone who has already downloaded and installed the Android SDK and is successfully building apps with it, you can copy all of their Android SDK files to a folder on your computer.

You need to look for the top-level Android SDK folder, such as **/Users/*user-name*/Android-SDK**, and copy the whole folder and its contents to your computer. A location such as **/Users/*your-name*/Android-SDK** is good. If it makes it easier, you can zip the folders and then unzip them onto your computer.

Note that there is no setup program to run. Copying the files from one computer to another is sufficient.

Tip: A typical Android SDK folder can be quite large (over 1 GB, depending on which additional packages have been installed). To build an app, you do not actually need all of the

Android SDK files. If you want to cut down the number of files, here is a list of the essential and optional folders:

Android SDK Folder	Required for building apps?
cmdline-tools (or tools)	Yes
build-tools	Yes (you only need the sub-folder for the latest version)
platforms	Yes (you only need android-30 for now)
platform-tools	Yes
add-ons	No
docs	No
emulator	No, unless you want to use an emulator
extras	No
licenses	Yes
sources	No
system-images	No, unless you want to use an emulator
temp	No

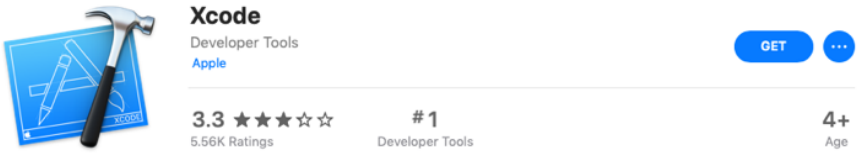
4. Installing Prerequisites for iOS

If you want to build iOS apps and upload them to the Apple App Store, you need to install the following components:

1. **Xcode**
2. **Transporter**

4.1. Install Xcode

Xcode is required to build iOS Apps. To install Xcode, simply search for Xcode in the Mac App Store and install it. Open Xcode at least once to agree to the licensing restrictions and install components.



4.2. Verify Xcode Installation

To verify that you have successfully installed Xcode and that it will be correctly used by Scripture App Builder, please open Terminal and run the following command to print the path to the active developer directory: `xcode-select -p`

```
$ xcode-select -p
/Applications/Xcode.app/Contents/Developer
```

If you have had Xcode Command-line Tools installed previously, it might still be pointing to that installation directory and Scripture App Builder will not work correctly.

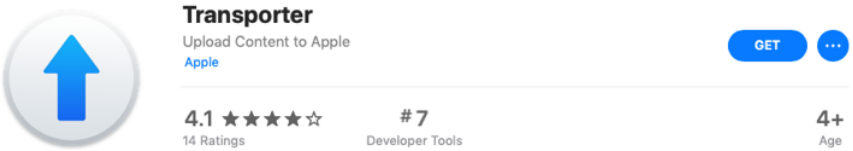
```
$ xcode-select -p
/Library/Developer/CommandLineTools
```

To correct this situation, run the following command to set the active developer directory.

```
sudo xcode-select -s /Applications/Xcode.app/Contents/Developer
```

4.3. Install Transporter

Transporter is used to upload iOS apps to App Store Connect. To install Transporter, simply search for Transporter in the Mac App Store and install it.



5. Building a Keyboard App for iOS

The process for building an Android App on a Mac follows the steps laid out in *the Keyboard App Builder 02 Building Apps* document. Building one for iOS has several additional considerations, including getting provisioning profiles for app.

5.1. Apple Development Store Setup

5.1.1. Enroll in the Apple Developer Program

To build iOS apps and distribute them through the Apple App Store, you will need to be enrolled in the Apple Developer Program. You can do this as either (i) an individual, or (ii) an organisation. The cost is USD $99 per year.

To enroll:

1. Go to https://developer.apple.com/programs/enroll/
2. Press the **Start Your Enrollment** button to start.

5.1.2. Create Signing Certificate

When you create an iOS app, it needs to be signed with a certificate.

To create a certificate:

1. Go to the Apple Developer website and log in to your account if you are not already.
2. Select **Certificates, Identifiers & Profiles**.
3. Click the  button to the right of the **Certificates** header.
4. On the **Create a New Certificates** page, select **Apple Distribution**. Then press the **Continue** button.
5. Upload a Certificate Signing Request. Press the **Learn more** link for instructions on how to use Keychain Access on your Mac computer to complete this task. Then press the **Continue** button.
6. On the **Download Your Certificate** page, press the **Download** button to download the certificate (distribution.cer) to your Mac.
7. Open the Keychain Access application. Choose the **login** item from the Keychains section on the left. Choose the **File** ➢ **Import Items…** menu item and browse to the Downloads folder and select the certificate (distribution.cer).

Now that the certificate is installed in the Keychain, you will be able to access it from within Scripture App Builder.

Note: To work with certificates, you will need the Apple Worldwide Developer Relations (WWDR) Certification Authority. It should have been installed with Xcode. When viewing your certificate in the Keychain Access application, if there is an error stating the certificate is not trusted, then install the certification authority.

To get the certification authority:

1. Download the Apple WWDR Certification Authority from:
   https://developer.apple.com/certificationauthority/AppleWWDRCA.cer

   For signing certificates created after January 28, 2021:
   https://www.apple.com/certificateauthority/AppleWWDRCAG3.cer
2. Open the Keychain Access application. Choose the **System** item from the Keychains section on the left. Choose the **File** ➢ **Import Items…** menu item and browse to the Downloads folder and select the certification authority (AppleWWDRCA.cer). You will be prompted for a username and password that has admin privileges in order to modify the **System** Keychain.
3. Choose the **File** ➢ **Lock Keychain "System"** menu item.

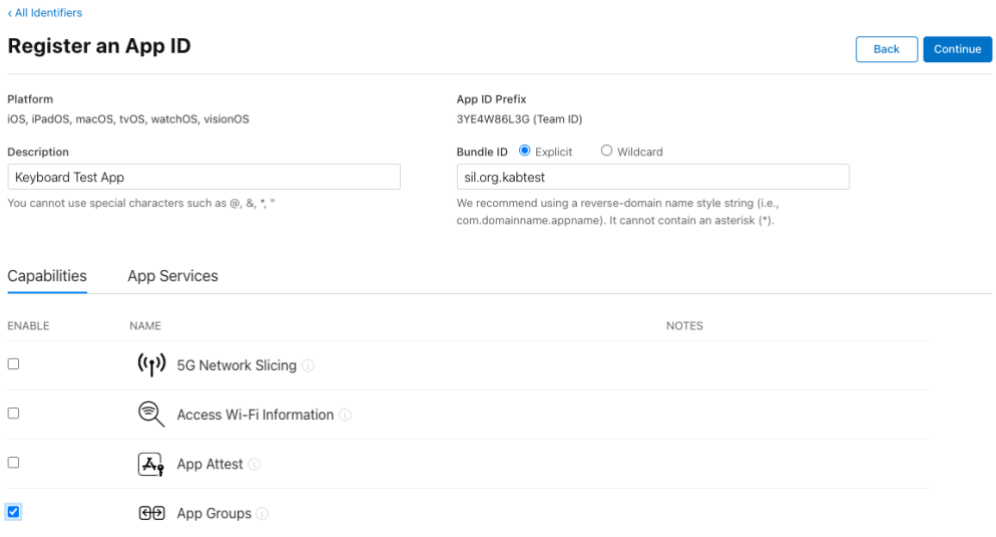
5.1.3. Create App Identifiers

First, select an app package name as described in section 4.1 of the *Building Apps* document. Log into the Apple Developer web site and select *Identifiers*. You will need to add two new identifiers to support your keyboard app, both belonging to a common unique group. For this example, a package name of *org.sil.kabtest* will be used.

For an iOS keyboard app, a second identifier is required to support the app extension that runs when a keyboard is used by another app. This identifier must be the original app package name with ".swkeyboard" appended to the end. So for this example, the second identifier will have a bundle ID of *org.sil.kabtest.swkeyboard"*.

These steps should be followed to create the app identifiers for this app:

1. Press the "+" sign at the top of the identifiers list to create a new identifier.
2. The next screen should say "Register a new identifier" and App Ids should already be selected. Press "Continue".
3. The next screen says "Register a new identifier" with options of "App" or "App Clip" with "App" selected. Press "Continue"
4. The "Register an App ID" screen is displayed next. You must fill out the *Description* field with a description of your choice. The *Bundle ID* field should be completed using the app package name (for our example: *org.sil.kabtest*). In the capabilities section, App Groups must be enabled. The following screen shot shows a completed entry:



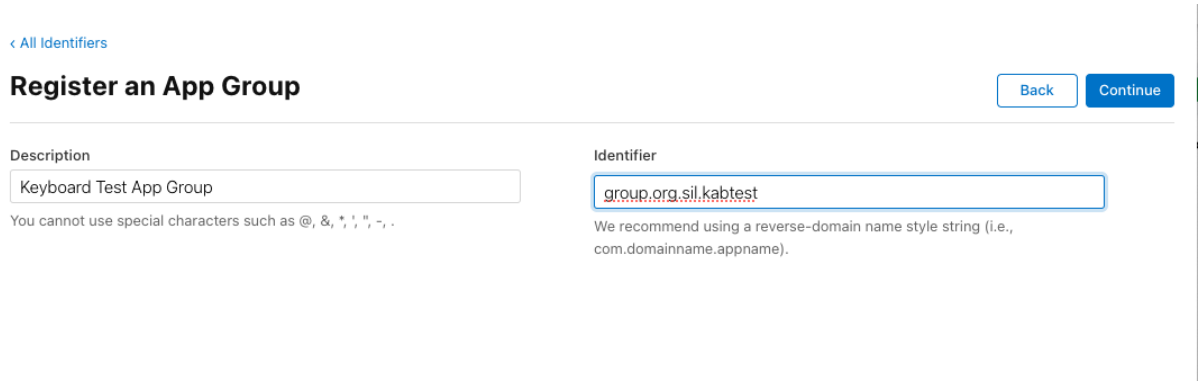
5. Press "Continue". A confirmation screen appears with the information you have entered. Verify that you have entered the Bundle Id correctly and that App Groups is checked. Then press "Register".
6. Repeat steps 1-5 using the keyboard bundle ID (example: *org.sil.kabtest.swkeyboard)* to create the second identifier for the app.

5.1.4. Create App Group

Both of the identifiers that have been created must belong to the same group. The group identifier that will be created must use your app package name with "group." appended to the front. For our example, we will create an app group with identifier *group.org.sil.kabtest*.

Start at the Identifiers list page, which is the same page that you started *Section 5.1.3* from. Follow these steps to create the app group:

1. Press the "+" sign at the top of the identifiers list to create a new identifier.
2. The next screen displays "Register a new identifier" and "App IDs" should be selected. Change the selection to "App Groups". Press "Continue"
3. The "Register an App Group" screen displays next. Fill the *Description* field with a description of your choice. Typing in the identifier will automatically start with *group.* So type in your package name to complete the group identifier (example *org.sil.kabtest* to create *group.org,sil.kabtest*). The following screen shot shows a completed entry:



4. Press "Continue". A confirmation screen appears with the information you have entered. Verify that the identifier is entered correctly and press "Register".

5.1.5. Associate App Identifiers to App Group

When you created the app identifiers, the App Groups capability was enabled, but there was no option at that point to set the group. This must now be configured for both of the two app identifiers which were created.

Start at the identifiers list page. If "App IDs" is not selected on the right, change the selection so that the list of app identifiers is displayed.

1. Find the entry for your first identifier and select it. (Example: *org.sil.kabtest*)
2. The "Edit your App ID Configuration" screen will be displayed. Beside the checked "App Groups" entry, there will now be a "Configure" button. Press the "Configure" button.
3. A list of the current group definitions will appear. Check the one that was created in the previous step for this app (*group.org.sil.kabtest*) and press "Continue".

4. This will return you to the “Edit your App ID Configuration” screen. The “Configure” button has been replaced with an “Edit” button and it should display “Enabled App Groups (1)”. Press “Save”.
5. A warning screen will display, press “Confirm”
6. The identifier list page should be redisplayed. Repeat steps 1-5 for the keyboard identifier (Example: *org.sil.kabtest.swkeyboard*) associating it with the same group (*group.org.sil.kabtest*) as the first identifier. So when this step is complete, both identifiers will be associated with the same group.

5.1.6. Create Provisioning Profiles

Provisioning profiles must be created for both of the applications that were registered. For publishing to the App Store, the app should be signed with profiles for the App Store. Within the Apple Developer website, navigate to the “Profiles” section. Use the following steps to create the required provisioning profiles:

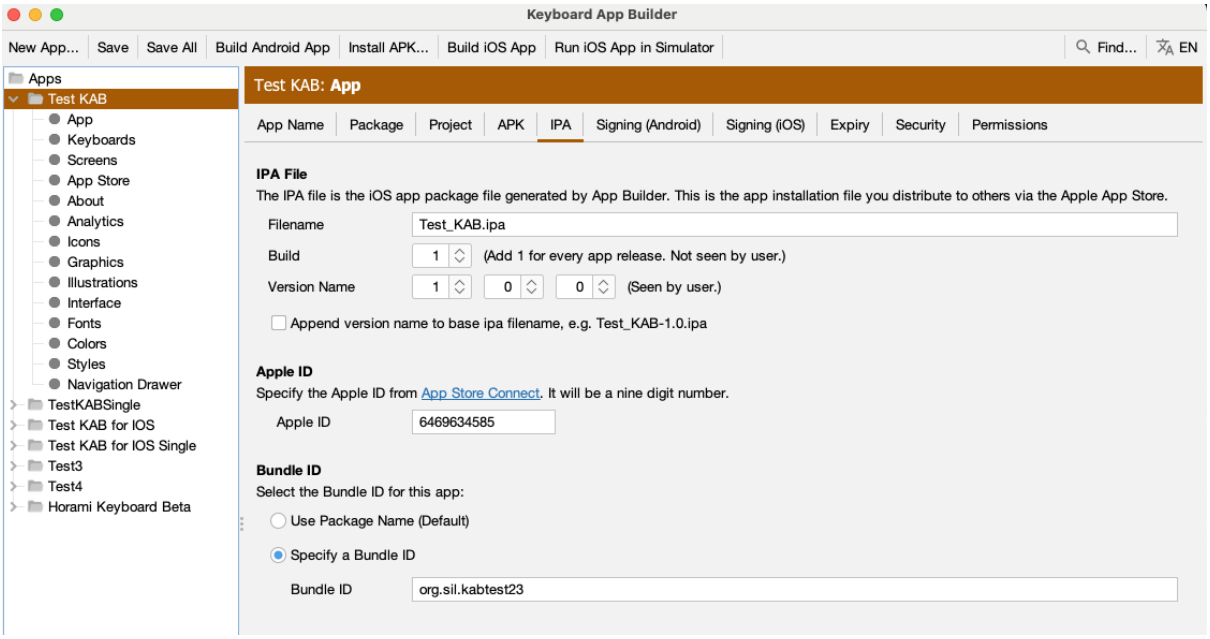
1. From the “Profiles” page, press the “+” button to create a new profile.
2. The “Register a New Provisioning Profile” screen will display. Select “App Store” as the type of profile to be created and press the “Continue” button.
3. The screen to select an app id displays. Select the app id that was created for the app package name in section 5.1.3. (Example: *org.sil.kabtest*). Press “Continue”.
4. The Certificate Selection screen will display next. Display a distribution certificate for your organization and press “Continue”.
5. The next screen asks for a name for the certificate. Add an appropriate name for the profile to be created and press “Generate”.
6. After the profile has been generated, the “Download and Install” screen displays. Press the “Download” button to copy the profile to your local Mac.
7. Repeat steps 1-6 to generate a provisioning profile for the keyboard app id (example: “org.sil.kabtest.swkeyboard”

5.2. Building iOS App using Keyboard App Builder

Most of the setup for building an app using Keyboard App Builder is described in the *the Keyboard App Builder 02 Building Apps.* This section will discuss the specific sections that are required for an iOS app.

5.2.1. IPA Tab

The ***IPA Tab*** contains the information needed by the builder to produce the IPA file, which is the final product of the builder. The IPA file is what you will send to the Apple App Store to publish.

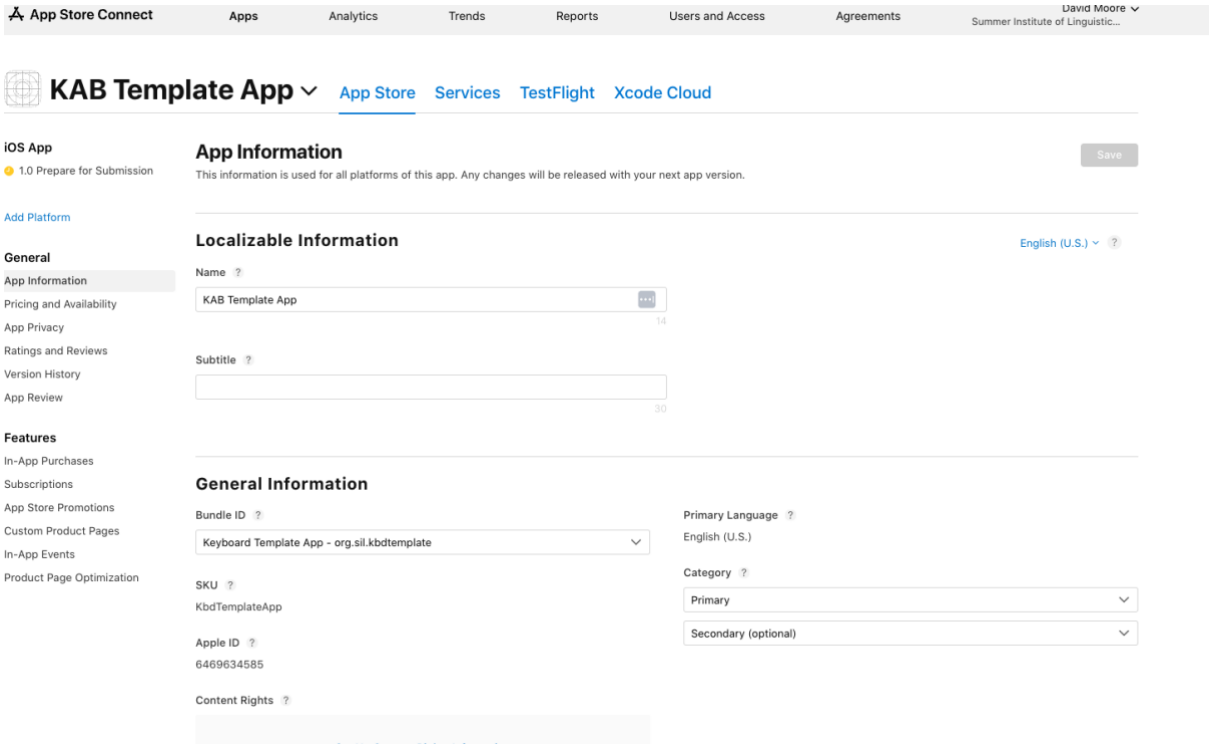


The ***Filename*** field on the screen for the ***IPA File*** section specifies the base name of the ipa file to be generated. If the checkbox at the bottom of the screen for *Append version name to ipa filename* is checked, then the version indicated by the *Version Name* fields is added to the base filename.

The ***Build*** field referenced as *Build* is also called *Bundle Version String, Bundle Version* or *CFBundleVersion* within Xcode and ITunes Connect. It represents the build number. The *Build* field expects an integer value and should be incremented with each file that is submitted to the ITunes Connect for release or testing.

The ***Version Name*** field is referenced as *Version, Bundle Short Version String, Bundle versions string, short* and *CFBundleShortVersionString* within Xcode and ITunes Connect. The field is created as a concatenation of the values of the three fields separated by a period. If the final field has a value of 0, then the version string is created from just the first two values. For values of 1, 2 and 0, the resulting string is “1.2”. For values of 1, 2 and 3, the resulting version string is “1.2.3”.

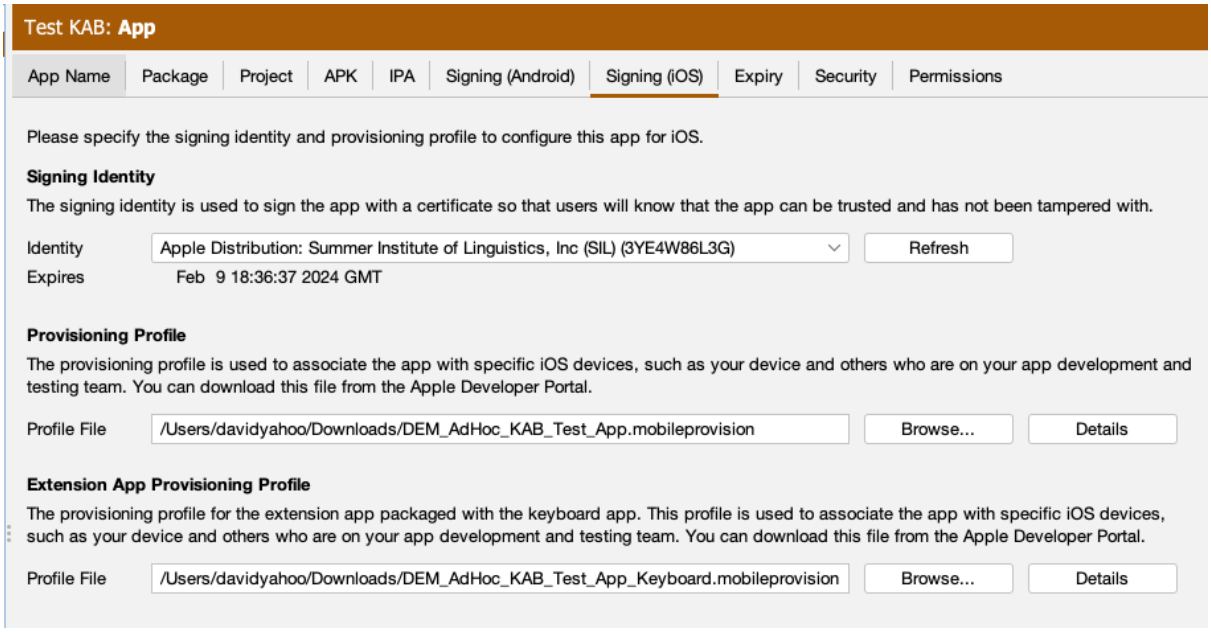
The **Apple ID** field is the id assigned by App Store Connect to the application in the app store. It can be obtained from **Apple ID** filed on the App Information tab in the App Store Connect entry for the app as shown below



The **Bundle ID** section allows the user to specify a Bundle ID for the app that is different than the package name. By default, the ***Package Name*** field from the ***Package*** tab is used as the Application Bundle ID that is used in creating the provisioning profiles and as the application identifier for the Apple App Store. If the user wants the iOS version of the app to have an identifier that is different than the package name that is used for the Android version of the app, press the ***Specify a Bundle ID*** button and then enter the ID to be used in the text box labelled ***Bundle ID.***

5.2.2. Signing (iOS) Tab

The ***Signing (iOS)*** tab is used to identify the certificate and provisioning profiles that the builder will require to sign the app in creating the IPA file.

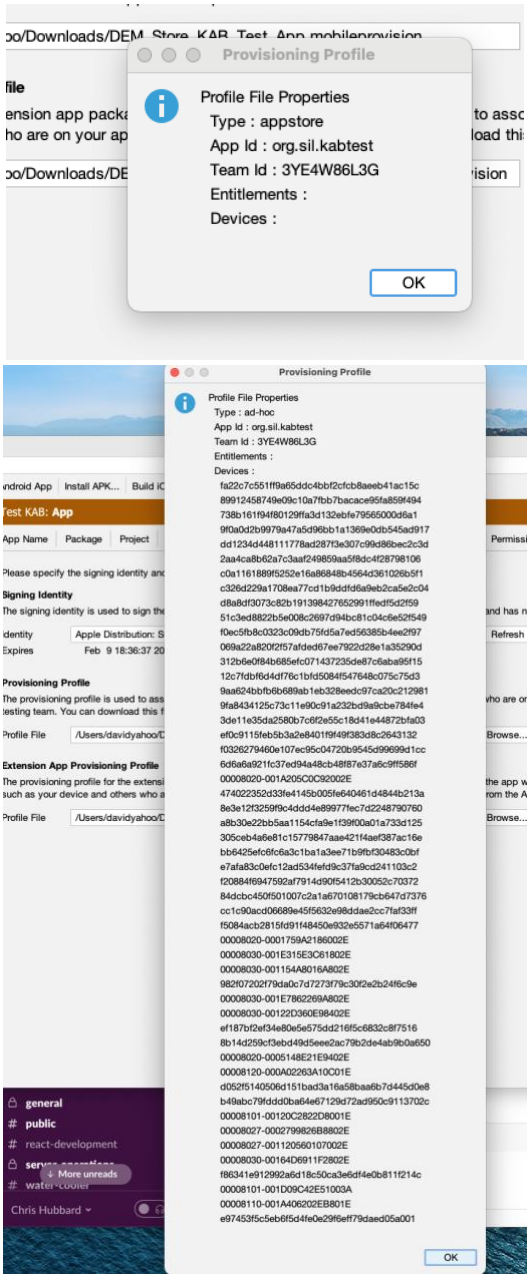


Select the **Signing Identity** from the drop down list of signing certificates which have been downloaded and installed to this system in the earlier steps. The certificate must match the signing certificate that was specified when building the provisioning profiles in the *Create Provsioning Profiles* section above.

For the **Provisioning Profile** entry, enter or browse to the mobile provisioning file that associated with the main app (example: *org.sil.kabtest*) in the *Create Provisioning Profiles* section above

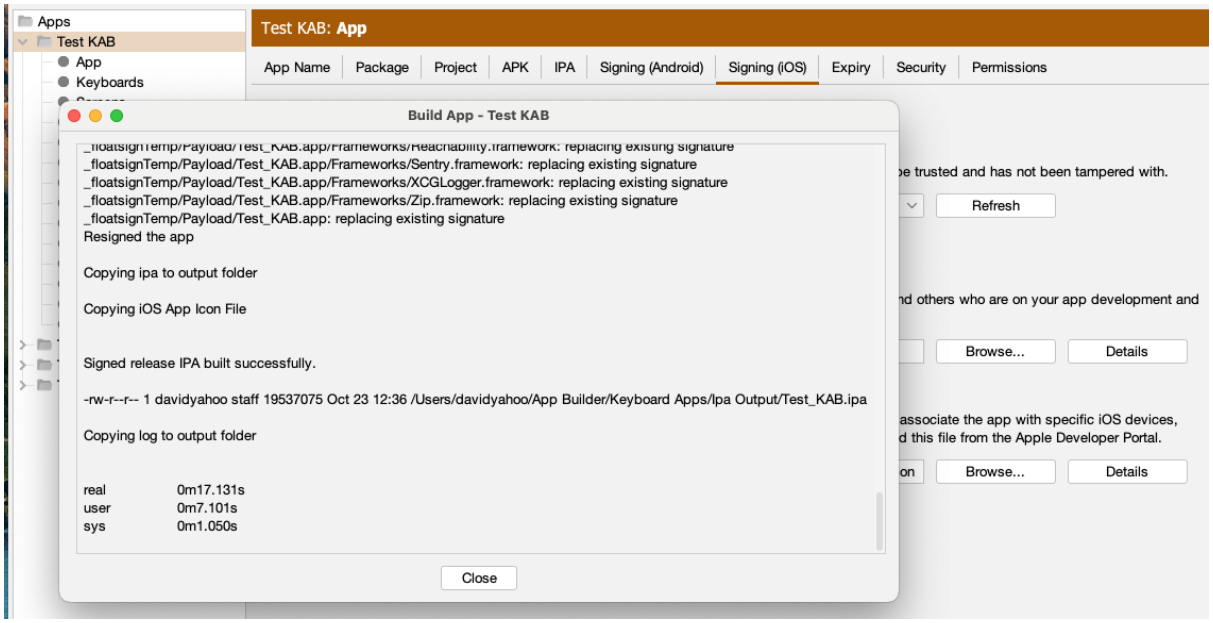
For the **Extenstion App Provisioning Profile** entry, enter or browse to the mobile provisioning file that was associated with the keyboard app (example: *org.sil.kabtest.swkeyboard*) in the *Create Provisioning Profiles* section above.

Each of the provisioning profiles has a **Details** button. Once a provisioning file has been selected, pressing this button will display the details associated with the selected file. It will identify the type of the profile (Appstore, adHoc or development), the app id and team id used to create the app, and the list of entitlements associated with the app. For an adHoc or development profile, it will provide a list of the ids of all devices that are registered as eligible devices for this profile. Below are a couple of examples of the **Details** display:


Provisioning Profile
Profile File Properties
Type : appstore
App Id : org.sil.kabtest
Team Id : 3YE4W86L3G
Entitlements :
Devices :
OK
Provisioning Profile
Profile File Properties
Type : ad-hoc
App Id : org.sil.kabtest
Team Id : 3YE4W86L3G
Entitlements :
Devices :
fa22c7c551ff9a65ddc4bbf2cfcb8aeeb41ac15c
89912458749e09c10a7fbb7bacace95fa859f494
738b161f94f80129ffa3d132ebfe79565000d6a1
9f0a0d2b9979a47a5d96bb1a1369e0db545ad917
dd1234d44811177 8ad287f3e307c99d86bec2c3d
2aa4ca8b62a7c3aaf249859aa5f8dc4f28798106
c0a1161889f5252e16a86848b4564d361026b5f1
c326d229a1708ea77cd1b9ddfd6a9eb2ca5e2c04
d8a8df3073c82b191398427652991ffedf5d2f59
51c3ed8822b5e008c2697d94bc81c04c6e52f549
f0ec5fb8c0323c09db75fd5a7ed56385b4ee2f97
069a22a820f2f57afded67ee7922d28e1a35290d
312b6e0f84b685efc071437235de87c6aba95f15
12c7fdbf6d4df76c1bfd5084f547648c075c75d3
9aa624bbfb6b689ab1eb328eedc97ca20c212981
9fa8434125c73c11e90c91a232bd9a9cbe784fe4
3de11e35da2580b7c6f2e55c18d41e44872bfa03
ef0c9115feb5b3a2e8401f9f49f383d8c2643132
f0326279460e107ec95c04720b9545d99699d1cc
6d6a6a921fc37ed94a48cb48f87e37a6c9ff586f
00008020-001A205C0C92002E
474022352d33fe4145b005fe640461d4844b213a
8e3e12f3259f9c4ddd4e89977fec7d2248790760
a8b30e22bb5aa1154cfa9e1f39f00a01a733d125
305ceb4a6e81c15779847aae421f4aef387ac16e
bb6425efc6fc6a3c1ba1a3ee71b9fbf30483c0bf
e7afa83c0efc12ad534fefd9c37fa9cd241103c2
f20884f6947592af7914d90f5412b30052c70372
84dcbc450f501007c2a1a670108179cb647d7376
cc1c90acd06689e45f5632e98ddae2cc7faf33ff
f5084acb2815fd91f48450e932e5571a64f06477
00008020-0001759A2186002E
00008030-001E315E3C61802E
00008030-001154A8016A802E
982f07202f79da0c7d7273f79c30f2e2b24f6c9e
00008030-001E7862269A802E
00008030-00122D360E98402E
ef187bf2ef34e80e5e575dd216f5c6832c8f7516
8b14d259cf3ebd49d5eee2ac79b2de4ab9b0a650
00008020-0005148E21E9402E
00008120-000A02263A10C01E
d052f5140506d151bad3a16a58baa6b7d445d0e8
b49abc79fddd0ba64e67129d72ad950c9113702c
00008101-00120C2822D8001E
00008027-0002799826B8802E
00008027-001120560107002E
00008030-00164D6911F2802E
f86341e912992a6d18c50ca3e6df4e0b811f214c
00008101-001D09C42E51003A
00008110-001A406202EB801E
e97453f5c5eb6f5d4fe0e29f6eff79daed05a001
OK

5.2.3. Build iOS App

Click on the **Build iOS App** button at the top of the screen. A window will open which will display a log for the build script that is run to produce the iOS App.

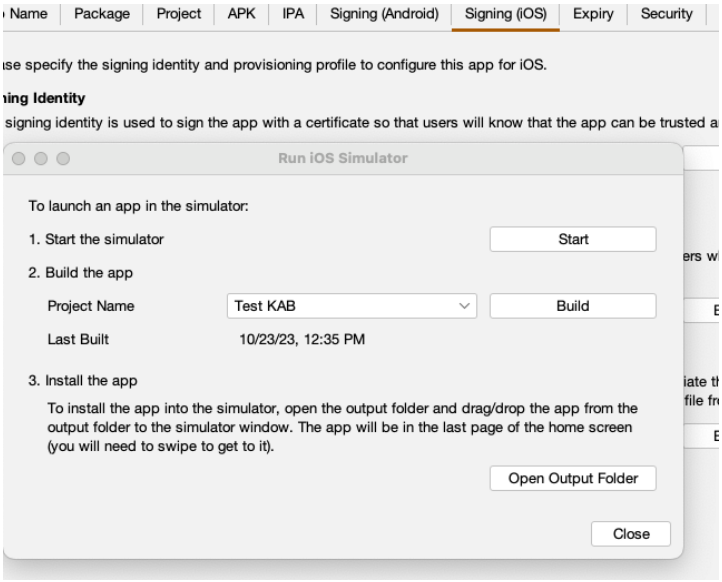


Examine the log window once the script has been completed. The message “Signed release IPA build successfully.” should appear in the window if the app has been build successfully.

The results of the build are an IPA file and an app that can be run in the simulator. They can be found in ~/App Builder/Keyboard Apps/Ipa Output/ and ~/App Builder/Keyboard Apps/Sim Output/.

5.2.4. Run iOS App in Simulator

This option is similar to the **Build iOS App** button but only produces the simulator app. It does not generate the ipa file for the app store. Click on the **Run iOS App in Simulator** button at the top of the screen. A dialog will pop up.



Pressing the **Start** button will start an instance of the iPhone simulator installed with XCode.

The **Project Name** drop down will initially be set to the name of the current project selected within KAB. If you want to build a different project, select the project from the drop down list of available projects.

The **Build** button will trigger a build of the app for the simulator and will display a window similar to the one for the **Build iOS App**. Any error that are encountered will be displayed in the log window. If everything is successful, a message: "Copying sim to output folder" should appear near the end of the log.

Open Output Folder will open a Finder window for the ~/App Builder/Keyboard Apps/Sim Output/ folder where the generated simulator apps are located. The app may be installed to the simulator by dragging it from Finder to the Simulator window. ***Due to coding limitations, only one keyboard app may be installed to a simulator instance at one time for testing. More than one KAB app will result in unpredictable results.***