Installing and Building Apps on a Mac
Contents

1. Introduction .............................................................................................................. 5

2. Installing Scripture App Builder .................................................................................. 5

3. Installing Prerequisites for Android ............................................................................ 5
   3.1. Java Development Kit (JDK) ......................................................................................... 6
   3.2. Installing Android Software Development Kit (SDK) ................................................... 7
       3.2.1. Copying the Android SDK files from someone else ............................................. 7
       3.2.2. Downloading the Android SDK packages from the internet ............................... 8

4. Installing Prerequisites for iOS ................................................................................. 14
   4.1. Install Xcode .............................................................................................................. 14
   4.2. Verify Xcode Installation ........................................................................................... 14
   4.3. Install Fastlane ........................................................................................................... 14

5. Installing Aeneas ..................................................................................................... 15

6. Testing App in iOS Simulator .................................................................................... 18
   6.1. Run the iOS Simulator ............................................................................................... 18
   6.2. Installing Additional Simulators ................................................................................ 19
   6.3. Manually Installing Apps into the Simulator ............................................................. 19

7. Creating iOS Certificates and Provisioning Profiles ................................................... 19
   7.1. Enroll in the Apple Developer Program .................................................................... 19
   7.2. Create Signing Certificate .......................................................................................... 20
   7.3. Create Provisioning Profile ........................................................................................ 21

8. Building an iOS App ................................................................................................. 23

9. Testing an iOS App ................................................................................................... 26

10. Using Xcode to Test an iOS App .............................................................................. 26

11. Using DeployGate to Test an iOS App ................................................................... 27
    11.1. Creating a DeployGate Account ............................................................................. 27
11.2. Uploading Your First App ................................................................. 27
11.3. Registering a Device ..................................................................... 29

12. Building from Terminal ................................................................. 35
1. Introduction
This document provides information on how to install Scripture App Builder and build apps on an Apple macOS system. It is possible to build an Android app using SAB on Windows, Linux or Mac, but if you want to build an iOS app for the iPhone or iPad, you will need to build it using a Mac computer.

<table>
<thead>
<tr>
<th>App Builder Platform</th>
<th>Build Android Apps</th>
<th>Build iOS Apps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Linux</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>macOS</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Creating an Android app on a Mac is essentially the same process as it is for Windows or Linux. To create a corresponding iOS app, you will need to enter a few more configuration items.

The apps generated by SAB for iOS will run on an iPhone or iPad with iOS 8.0 or higher.

2. Installing Scripture App Builder
To install the Scripture App Builder program files:

1. Download the current Mac installer file (dmg) from the SAB website: http://software.sil.org/scriptureappbuilder/download/
2. Double click on the file within Finder to open the disk image that contains the Scripture App Builder application.
3. Copy the Scripture App Builder application to your Application folder. This can be done by starting another instance of Finder and opening the Applications folder. From there you can simply drag the Scripture App Builder icon from the disk image window opened in the last step to the Applications folder in Finder.

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 Android apps. We recommend you use Amazon Corretto, which is a free distribution of OpenJDK.

Go to the Amazon Corretto 8 Downloads page:

https://docs.aws.amazon.com/corretto/latest/corretto-8-ug/downloads-list.html

There are many download files on this page. You are looking for the file that corresponds to your computer’s operating system type: macOS x64. It is easiest to download the pkg (package file), e.g. amazon-corretto-8.212.04.2-macosx-x64.pkg, rather than the tar.gz file.

When the Mac .pkg file has downloaded, double click on it to start the installation wizard. Follow the steps in the wizard.
3.2. Installing Android Software Development Kit (SDK)

There are two ways of installing the Android Software Development Kit (SDK):

1. **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 3.2.1 for more details.

2. **Download the Android SDK packages from the internet:**
   Download the basic SDK command line tools and then download and install three additional packages. This method will require an internet connection.

   See 3.2.2 for more details.

3.2.1. Copying the Android SDK files from someone else

If you know someone who has already downloaded and installed the Android SDK onto their Mac 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/your-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 (several hundred MB, or even a few GB, depending on which additional packages have been installed). To build an app with Scripture App Builder, 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:

<table>
<thead>
<tr>
<th>Android SDK Folder</th>
<th>Required for building apps?</th>
</tr>
</thead>
<tbody>
<tr>
<td>tools</td>
<td>Yes</td>
</tr>
<tr>
<td>build-tools</td>
<td>Yes (you only need the sub-folder for the latest version)</td>
</tr>
<tr>
<td>platforms</td>
<td>Yes (you only need Android-28 for now)</td>
</tr>
<tr>
<td>platform-tools</td>
<td>Yes</td>
</tr>
<tr>
<td>add-ons</td>
<td>No</td>
</tr>
<tr>
<td>docs</td>
<td>No</td>
</tr>
</tbody>
</table>
3.2.2. Downloading the Android SDK packages from the internet

To install the Android SDK from the internet, you need the basic SDK command line tools rather than the full Android Studio installation.

The installation will be done in two steps:

A. Download and install the basic SDK command line tools.

B. Download and install additional packages from within Scripture App Builder.

A. Download and install the basic SDK tools

Do the following:

   But do not download Android Studio.

2. Scroll down to the bottom of the web page until you see the sub-heading, Command line tools only.

3. Download the latest Mac package, e.g. sdk-tools-darwin-4333796.zip. The file size is around 98 MB.

4. Before downloading, you will need to click a checkbox to confirm you agree with the Terms and Conditions.
5. When the zip file has downloaded (if you are not using Safari, you may have to unzip the zip file first), open a terminal window and move the contents to a new folder:

```bash
mkdir ~/Android-SDK
mv ~/Downloads/tools ~/Android-SDK
```

When the file is unzipped, the /Users/your-name/Android-SDK folder should contain a single sub-folder named ‘tools’. You do not need to run any setup program here.

B. Download and install additional SDK packages

The Android SDK as it stands will not be sufficient to build an app. You need to download three additional packages:

(i) Android SDK Build-tools (110 MB),
(ii) Android SDK Platform-tools (23 MB), and
(iii) Android SDK Platform (101 MB).

To install these packages:

1. Launch Scripture App Builder.
2. Select Scripture App Builder ➤ Preferences from the main menu.
3. Go to the Android SDK tab, which is the second tab.
4. Enter the Android SDK Location by clicking Browse and navigating to the SDK folder which you have just created. If you followed the suggestion above, this will be /Users/your_name/Android-SDK.
If the Android SDK location is valid, you should see the version of the Tools package displayed in green.

5. Click **Install Packages...** to begin downloading and installing the three packages that are “Not Found” (Build tools, Platform tools and Platform API). You will need an internet connection.

   Type **y** (for yes) followed by **Enter** to agree with the license agreement.
Depending on the speed of your internet connection the downloads might take some time. There are over 230 MB of files to download.

As each of the three packages are downloaded, you should see a series of messages, concluding with “...complete” and “…finished”.
Info: "Install Android SDK Build-Tools 28.0.3 (revision: 28.0.3)" ready.
Info: Installing Android SDK Build-Tools 28.0.3 in /Users/richard/Android-SDK/build-tools/28.0.3
Info: "Install Android SDK Build-Tools 28.0.3 (revision: 28.0.3)" complete.
Info: "Install Android SDK Build-Tools 28.0.3 (revision: 28.0.3)" finished.

[====================================] 100% Unzipping... android-9/zipalign

Installing platform-tools...
Info: Parsing /Users/richard/Android-SDK/build-tools/28.0.3/package.xml
Info: Parsing /Users/richard/Android-SDK/tools/package.xml
Info: Preparing "Install Android SDK Platform-Tools (revision: 29.0.2)".
Info: "Install Android SDK Platform-Tools (revision: 29.0.2)" ready.
Info: Installing Android SDK Platform-Tools in /Users/richard/Android-SDK/platform-tools
Info: "Install Android SDK Platform-Tools (revision: 29.0.2)" complete.
Info: "Install Android SDK Platform-Tools (revision: 29.0.2)" finished.

[====================================] 100% Unzipping... platform-tools/system

Installing platform...
Info: Parsing /Users/richard/Android-SDK/build-tools/28.0.3/package.xml
Info: Parsing /Users/richard/Android-SDK/platform-tools/package.xml
Info: Parsing /Users/richard/Android-SDK/tools/package.xml
Info: Preparing "Install Android SDK Platform 28 (revision: 6)".
29% Downloading platform-28_r06.zip...

[====================================] 100% Unzipping... platform-tools/system

Installing platform...
Info: Parsing /Users/richard/Android-SDK/build-tools/28.0.3/package.xml
Info: Parsing /Users/richard/Android-SDK/platform-tools/package.xml
Info: Parsing /Users/richard/Android-SDK/tools/package.xml
Info: "Install Android SDK Platform 28 (revision: 6)" complete.
Info: "Install Android SDK Platform 28 (revision: 6)" finished.

[====================================] 100% Unzipping... android-9/framework.

logout
Saving session...
copying shared history...
saving history...truncating history files...
completed.
Deleting expired sessions...18 completed.

[Process completed]
If you get a message saying “Install… failed” or “Failed to read or create install properties file”, this might be because:

- the internet connection is not working, or
- you do not have enough space on your hard disk.

If the installation of a package has failed, make sure you have an internet connection and enough hard disk space. Then click **Install Packages** again.

6. When all the required packages have been installed, click **Check Installation**. A message will appear listing the packages that have been installed.

If the installation was successful, you should see the version numbers displayed in green:

The Android SDK has now been installed for use by Scripture App Builder.
4. Installing Prerequisites for iOS

If you want to build iOS apps, you need to install the following components:

1. Xcode 9 or greater.
2. Sigh resigning tool.

4.1. Install Xcode

Xcode 9 or greater is required to build iOS Apps. Xcode is also required to upload the iOS app to the App Store.

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 is 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
```

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 Fastlane

The Fastlane Tools are used to sign the iOS apps that are created so that they may be published to the App Store. It requires the Xcode installation from the previous step to install successfully.

To install Fastlane Tools:

1. Launch Scripture App Builder.
2. Select Scripture App Builder ➤ Preferences from the main menu.

3. Go to the Fastlane tab, which is the third tab.

![Fastlane Settings](image)

If Fastlane Tools installation is valid, you should see the version number displayed in green. If you see the version displayed in red, you can re-install by following the next step.

4. Click Install… to begin downloading and installing the Fastlane Tools. You will need an internet connection.

5. Installing Aeneas

Aeneas is the audio-text synchronization tool that may be run from within Scripture App Builder to create timing files for phrase-by-phrase highlighting. If the apps you are building do not include audio or if the timing files are already available, then there is no need to install it.

To install aeneas:

1. Download the aeneas tools for Mac file (dmg) from the SAB website from the section labeled Audio Synchronization Tools:

   [http://software.sil.org/scriptureappbuilder/download/](http://software.sil.org/scriptureappbuilder/download/)

2. Double click on the file within Finder to open the disk image that contains the aeneas-mac-setup install package. Control-click on the install package and select
Open. You will get a warning that this package is from an unidentified developer. Press Open.

3. The introduction screen will display. Press Continue.

4. A “Read Me” screen will display next followed by a “License” screen. Press Continue on both screens.

5. Pressing the Continue button on the License screen will bring up a screen asking if you agree to the terms of the license. Press Agree.

6. The “Destination Select” screen selects the default drive. Press Continue.

7. The “Installation Type” screen displays next for a standard install. Press Install.
8. The installer will prompt for credentials to install the software. Enter a username and password with permissions and press **Install Software**.

9. At this point the installation will start and will show progress screens until it completes. A terminal window will popup briefly to test the installation. When the application completes successfully, the original screen will show:

10. Press **Close**.

Aeneas is installed in `/usr/local/lib/python2.7/site-packages`. 

17
6. Testing App in iOS Simulator
When you want to test your app, you can either use a device or a simulator. To test with a device, you will need a signing certificate and provisioning profile (see next section). To test with a simulator, you will need to download Xcode, the integrated development environment used to build and test iOS and Mac apps. Xcode is available for free in the Mac App Store and is quite large (5.46GB). SAB requires Xcode 9 or greater (which requires macOS Sierra). To install:

2. Install from the App Store.
3. Start Xcode at least once to complete the installation.

6.1. Run the iOS Simulator
Once you have a project configured and ready to test, click on the Run iOS App in Simulator on the toolbar.

![Run iOS Simulator](image)

Select the simulator you would like to run on and click Start. It takes a little bit of time for the simulator to start. If you want to switch simulators, select a different Simulator from the Simulator Type combo box and click Start again. It will close the previous Simulator and start the new own.

Select the project you want to test (it defaults to the selected project in the Apps list) and click on Build. This will build the app for the Simulator in a separate terminal window. When the build is complete, you can click on Launch to run the app in the Simulator.

You may close the dialog and make changes to your project. When you restart the Run iOS Simulator dialog again, you will need to Build and Launch again for the changes to be included.
6.2. Installing Additional Simulators
You may install simulators for previous versions of iOS by launching Xcode and viewing the preferences dialog and selecting the Components tab.

![Simulator Components](image)

6.3. Manually Installing Apps into the Simulator
If there is a problem with launching the simulator from the Run iOS Simulator dialog, you can manually install the app by dragging the built app from the Simulator output folder (found in ~/App Builder/Scripture Apps/Sim Output) onto the Simulator.

To start the Simulator, launch Xcode and from the Xcode menu select Open Developer Tool ➢ Simulator. You will still need to rebuild the app from the Run iOS Simulator dialog.

7. Creating iOS Certificates and Provisioning Profiles

7.1. 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:

2. Press the **Start Your Enrollment** button to start.
7.2. Create Signing Certificate

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

To work with certificates you will first need the Apple Worldwide Developer Relations (WWDR) Certification Authority and ensure it is installed within Keychain Access.

To get the certification authority:

1. Download the WWDR Certification Authority from:
   https://developer.apple.com/certificationauthority/AppleWWDRCA.cer
2. Double-click this file within Finder to install it into Keychain Access.

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. Select Certificates ➤ Production on the left of the page.
4. Click the (+) button on the top right of the page to add a certificate.
5. In answer to ‘What type of certificate do you need?’ select Production ➤ App Store and Ad Hoc. Then press the Continue button.
6. On the page ‘About Creating a Certificate Signing Request’, follow the on-screen instructions to create a CSR file. Then press the Continue button.
7. On the page ‘Generate Certificate’, click Choose File and select the CSR file you created in the previous step. Then press the Continue button.
8. On the ‘Your certificate is ready’ page, click Download to the download the certificate (ios_distribution.cer) to your Mac.
9. Find the certificate file in your downloads and double-click it to add it to your login keychain.

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

Additional information about signing certificates is available at:
7.3. Create Provisioning Profile

Provisioning profiles are used to associate your app with specific iOS devices, such as your device and others who are on your app development and testing team.

Creating new mobile provisioning files

To create a new provisioning profile, you will need an App ID and at least one registered device.

To create an App ID:

1. Go to the Apple Developer website and log in to your account if you are not already.
2. Select Certificates, Identifiers & Profiles.
3. Select Identifiers ➔ App IDs on the left of the page.
4. Click the button on the top right of the page to add an App ID.
5. Enter an App ID Description of your choice.
   It can be the App Name from the App ➔ Package page of Scripture App Builder.
6. For App ID Suffix, choose Explicit App ID and enter your app package name from the App ➔ Package page of Scripture App Builder.
7. Leave all the App Services unchecked. None are needed in the app.
8. Click the Continue button.

To register a device (i.e. a specific iPhone or iPad for testing):

1. Select Devices ➔ All on the left of the page.
2. Click the button on the top right of the page to add a device.
3. In the section Register Device, enter a name of the device (such as “John Smith’s iPhone”) and its UDID (unique device identifier, a sequence of 40 letters and numbers). Press Continue.
4. On the ‘Review and register’ page, check that the device information is displayed correctly and press Register to confirm.

   Note that you can register up to 100 devices of each type (e.g. up to 100 iPhones, 100 iPads) per year of your Apple Developer Program membership. You can remove devices that you no longer need at the beginning of the next membership year.

To create a provisioning profile:

1. Select Provisioning Profiles ➔ Distribution on the left of the page.
2. Click the button on the top right of the page to add a provisioning profile.
3. On the page ‘What type of provisioning profile do you need?’, under Distribution, select Ad Hoc (for testing) or App Store (for submission to the App Store). Click Continue to move to the next page.

4. On the ‘Select App ID’ page, select the App ID from the list of App IDs you have already defined. Click Continue to move to the next page.

5. Select the certificate to include in the profile (iOS Distribution). Click Continue to move to the next page.

6. On the ‘Select devices’ page, check the device(s) that you want to be able to install and run the app. Click Continue to move to the next page.

7. On the ‘Name this profile and generate’ page, give the profile a name of your choice. Click Continue to move to the next page.

8. On the ‘Your provisioning profile is ready’ page, click Download to save your new .mobileprovision file to your computer. This is the Provisioning Profile file that you will select in Scripture App Builder.

Download existing mobile provision files
If other members of your team have already created provisioning profiles, there are several ways to download them. They can be downloaded from the App Developer website by selecting the profile to be downloaded and pressing the Download button.

Another method is to use sigh, which was installed in an earlier step, to download all of the available mobile provision files. If you are associated with multiple teams, the program will ask you to select a team and will download the files available for that team.

To use sigh to download existing provisioning profiles:

1. Create a directory where the downloaded files will be saved (e.g. ~/Documents/<some directory>

2. Open an instance of “terminal” and “cd” to the directory that has been created.

3. Run “sigh download_all”

4. The utility will prompt: “Your Apple ID Username:”. Enter your ID. (e.g. YourId@your.org)

5. The utility will prompt: “Password (for YourId@your.org)”. Enter your password.

6. If the user is associated with multiple teams, the utility will prompt: “Multiple teams found on the Development Portal, please enter the number of the team you want to use:” followed by a list of the teams. Enter the number of the team that the app you are building is associated with.
7. The utility will then download all of the mobile provision files to the current directory.

8. Building an iOS App

To build an iOS app:

1. Create a new app project following the instructions in the SAB Building Apps document.
2. Select the Signing (iOS) tab to open the iOS signing options for the app.
3. 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.
4. For the Provisioning Profile entry, enter or browse to the mobile provisioning file associated with the app that was downloaded in the earlier steps.
5. Click on the **IPA** tab.

   This allows you to set the name of the ipa file to be generated as well as the build and version information.

   The **Filename** field on the screen 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. So 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”.

24
6. Click on the **Build iOS App** button at the top of the screen. A terminal window should open. The build script for the iOS App should run within that terminal window.

![Build iOS App](image)

7. Examine the terminal window once the shell script has been completed. The message “Signed release IPA built successfully” should appear in the window if the app has been built successfully. (Note that occasionally the terminal window will appear behind the Scripture App Builder and that you have to select the terminal to review the results).

8. 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/Scripture Apps/Ipa Output/ and ~/App Builder/Scripture Apps/Sim Output/.

![IPA file and app](image)
9. Testing an iOS App

After building the iOS IPA file, you will want to install it and test it on one or more devices before you submit it for publication to the Apple App Store. This manual describes two ways of doing this:

1. Use Xcode to install the IPA file to an iPhone or iPad that is connected to your computer. 

   *This method is recommended if you have your test devices with you. It does not involve uploading and downloading the IPA to and from the internet.*

2. Use DeployGate to upload the IPA file to the internet and share it with limited number of devices to download, install and test.

   *This method is recommended if you have good internet access and/or you have a team of testers who are elsewhere.*

10. Using Xcode to Test an iOS App

To test your iOS IPA file using Xcode, do the following:

1. Launch Xcode and select Window ➤ Devices and Simulators.

2. Connect an iPhone or iPad to the Mac using a cable and unlock the device.

3. On the Mac, iTunes may launch and show a dialog asking “Do you want to allow this computer to access information...” Click on the Cancel button.

   • Note: This feature can be turned off in iTunes. Select iTunes ➤ Preferences..., select the Devices tab, and click on the Prevent iPods, iPhone, and iPad from syncing automatically checkbox.

4. On the device, it may prompt to Trust This Computer. Tap on the Trust button. This may require to enter the Passcode to trust this computer. The device will show up in the Xcode Devices window.

5. The first time the device is connected to the Mac, Xcode will take some time to Prepare debugger support. This may take some time. Please wait.

6. In the Xcode Devices window, there will be a section named INSTALLED APPS. Click on the + button.

7. Find the IPA file to add. Click Open after you have selected it.

8. Click the Install button next to the name of the app.

9. Wait until the Mac installs the app to your device.

After the install is complete, you will see the app icon on your device and you can test it.
11. Using DeployGate to Test an iOS App

DeployGate enables you to test your app and share it with a limited number of users to test. You upload the IPA file for iOS apps or the APK for Android apps and then download them to your phone or tablet device. You can also invite testers to install your app and help with the testing.

A DeployGate app is installed on the testing device. It will show all of the apps that you have uploaded to DeployGate and allows them to be installed on the device.

11.1. Creating a DeployGate Account

The first step is to create a DeployGate account. To do this:

1. Go to https://deploygate.com
2. Press the Get Started button.
3. Enter an email address, a user name and a password.
4. Press the Sign up for DeployGate button.

11.2. Uploading Your First App

Once the sign in screen has been successfully completed, you are presented with a screen that prompts you to upload your app. While there are several methods described for embedding it as part of your build process, the way we have used this to date is to simply upload the IPA file that has been created by locating the ipa file in Finder and then dragging it to the bottom area of the screen where it has a green Upload App area:
Dragging the file to the upload area causes a new dialog to be displayed with the name of the file and a text box where you can enter a short note that will be displayed on your profile window and also on the DeployGate app when the user is selecting the app. It is a good place to write a short note on the reason for the update so that it is easy for the testers to see that the app has been updated and to see what the primary reason they need to update is. Complete the screen and press the **Upload** button.
When the upload is complete, a new dialog is displayed with a QCode bar code and the option to send an email to your device. The QCode can be read in with your iPhone or iPad which will trigger an installation of the DeployGate app using your app profile.

Alternatively, you can enter an email address at this point, which you would also open on the iPhone to install the DeployGate app with the correct profile. Or you can simply go on and add users and devices later.

After this, the screen that is displayed is what you will normally see when you login. The screen has an entry for each app that you have uploaded. It has an Upload App button that can be used to upload new versions of the same app or to upload a new app.

11.3. Registering a Device
If the iOS device was not originally in the mobile provisioning profile and if the device has not been previously registered in your Apple Developer account, you need to add it to both.
There is a method for manually doing this, but DeployGate provides a way of simplifying the process so that you don’t need to go and look up UDID for the device.

First, try to install the app using the method described above. You will not be able to install it because the UDID is not registered for the app. However this will result in the device being registered in DeployGate which allows the following steps.

After attempting to install the app, re-enter DeployGate in your browser and open the entry for your app. As you can see in the screen below, it will show that a new UDID has been registered for the device. Press the **Options** button below and select the **Package Archive** option. Next click the little tag symbol inside the app box to open the UDID list.

The next screen shows a list of the devices that have been observed by DeployGate or that were included in the provisioning profile. Your new device entry will show up on the screen with a **Not Exist** entry. Make sure that the entry for the new device is checked and then press the **Export Selected UDIDs** button. This will create a file “multiple-device-upload-ios.txt” that can be used on the Apple Developer website to add these devices to the mobile provisioning file.
After logging in to Apple Developer, click on Certificates, IDs and Profiles. Press the All selection under Devices as shown in the illustration below and then select Register Multiple Devices. Then press the Choose File button.
Find the multiple-device-upload-ios.txt file that was created by DeployGate and then press Continue.
A review screen will be displayed which should list the name to be assigned to the device along with the UDID associated with the device. Review to ensure this is correct and press **Register**.

![Review and register.](image)

Confirm the device information is correct. Once this device is registered, you will not be able to edit the UDID and can only edit the name or disable it.

- **Name:** dmoore1768 - iPad 3 (WiFi)
- **Model:** iPad Wi-Fi (3rd generation)
- **UDID:** 9f0a062b59979a47a5d960b1a1369e00b545af917

You can register 96 more of this device type. The maximum number of each device type that you can register per membership year is:

- Apple TV: 100
- Apple Watch: 100
- iPad: 100
- iPhone: 100
- iPod Touch: 100

You may reset your device list at the start of your next membership year.
At this point your device has been registered to your Apple Developer account. You now need to add the device to the provisioning profile for your app. Select the provisioning profile being used to test the app. Make sure your test device is checked in the list of devices at the bottom of the screen. Press Generate.

**Note:** For the purposes of testing with DeployGate, an AdHoc type of provision profile must be created and used. If you have not selected AdHoc, the list of devices will not be available on the screen.

The following screen will display:
At this point you can either download the new provisioning profile that has been generated or return to the step in section 6.3 and run “sigh download_all” to refresh the provisioning profiles.

Next you need to rebuild the iOS app using the new profile and upload it to DeployGate again. This time when you attempt to install it, the **Install** button should be enabled and will allow you to install your app.

### 12. Building from Terminal

Scripture App Builder has a command line interface which allows you to create a new app and build it, or load an existing app and build it.

The base command calls Java to access the jar file within the Scripture App Builder application followed by a series of options described below. The base command is:

```
java -jar "/Applications/Scripture App Builder.app/Contents/Java/bin/Scripture App Builder.jar"
```
The available parameters are:

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-new</td>
<td>Create a new app project</td>
</tr>
<tr>
<td>-load &lt;project&gt;</td>
<td>Load an existing app project</td>
</tr>
<tr>
<td>-build</td>
<td>Build app project (use with either -new or -load)</td>
</tr>
<tr>
<td>-no-save</td>
<td>Do not save changes to app (use with -load)</td>
</tr>
<tr>
<td>-resign</td>
<td>Resign iOS Template App (use with either -new or -load)</td>
</tr>
<tr>
<td>-?</td>
<td>Show command line help</td>
</tr>
<tr>
<td>-n &lt;app-name&gt;</td>
<td>Set app name. Enclose the name in &quot;double quotes&quot; if it contains spaces.</td>
</tr>
<tr>
<td>-p &lt;package-name&gt;</td>
<td>Set package name, e.g. com.myorg.language.appname</td>
</tr>
<tr>
<td>-b &lt;filename&gt;</td>
<td>Add book or bundle file. This could be a USFM file or a zipped set of USFM files. It could also be a Digital Bible Library text release bundle.</td>
</tr>
<tr>
<td>-i &lt;filename&gt;</td>
<td>Include additional parameters file. Use the full path of the file and enclose it in &quot;double quotes&quot; if there is a space in the path.</td>
</tr>
<tr>
<td>-a &lt;filename&gt;</td>
<td>Set about box text, contained in text file. Use the full path of the file and enclose it in &quot;double quotes&quot; if there is a space in the path.</td>
</tr>
<tr>
<td>-f &lt;fontname&gt;</td>
<td>Set font name or filename, e.g. &quot;Charis SIL Compact&quot;, &quot;c:\fonts\myfont.ttf&quot; The font name must be one of the items in the list of fonts in the New App wizard. For other fonts, specify the full path to the font filename.</td>
</tr>
<tr>
<td>-g</td>
<td>Use Grandroid</td>
</tr>
<tr>
<td>-ic &lt;filename&gt;</td>
<td>Add launcher icon (one or more .png files). Use the full path of the files and enclose them in &quot;double quotes&quot; if there is a space in the path.</td>
</tr>
<tr>
<td>-l &lt;lang-code&gt;</td>
<td>Set language for menu items and settings, e.g. en, fr, es</td>
</tr>
<tr>
<td>Option</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>-ft &lt;feature=value&gt;</td>
<td>Set a feature, e.g. book-select=grid</td>
</tr>
<tr>
<td>-vc &lt;integer&gt;</td>
<td>Set version code, e.g. 1, 2, 3, or +1 to increment the current version code by 1.</td>
</tr>
<tr>
<td>-vn &lt;string&gt;</td>
<td>Set version name, e.g. 1.0, 2.1.4, or use +1, +0.1, +0.0.1 to increment the current value.</td>
</tr>
<tr>
<td>-ks &lt;filename&gt;</td>
<td>Set keystore filename. Use the full path of the file and enclose it in &quot;double quotes&quot; if there is a space in the path.</td>
</tr>
<tr>
<td>-ksp &lt;password&gt;</td>
<td>Set keystore password</td>
</tr>
<tr>
<td>-ka &lt;alias&gt;</td>
<td>Set key alias</td>
</tr>
<tr>
<td>-kap &lt;password&gt;</td>
<td>Set key alias password</td>
</tr>
<tr>
<td>-fp &lt;folder=path&gt;</td>
<td>Set a folder path, e.g. &quot;app.builder=c:\Scripture App Builder&quot;.</td>
</tr>
<tr>
<td>-ta &lt;target-api&gt;</td>
<td>Set Target API, e.g. 21 for Android 5.0, 22 for Android 5.1.</td>
</tr>
<tr>
<td>-si &lt;signing identity&gt;</td>
<td>Set Signing Identity to use for iOS Resigning</td>
</tr>
<tr>
<td>-pp &lt;provisioning profile&gt;</td>
<td>Set full path to provisioning profile for iOS resigning</td>
</tr>
<tr>
<td>-bn &lt;integer&gt;</td>
<td>Set build number for ipa file, e.g. 1, 2, 3, or +1 to increment by 1</td>
</tr>
<tr>
<td>-vs &lt;string&gt;</td>
<td>Set version string for ipa file, e.g. 1.0, 2.1.4 or +1, +0.1, +0.0.1</td>
</tr>
</tbody>
</table>

**Examples:**

```java
Java -jar "/Applications/Scripture App Builder.app/Contents/Java/bin/scripture-app-builder.jar" -load "Mali" -resign -bn "5" -vs "2.3.2" -si "iPhone Distribution: Summer Institute of Linguistics, Inc (SIL) (4YF5X97M4H) " -pp "/Users/builder/Documents/MobileProvision/AdHoc_org.wycliffe.app.mali.mobileprovision"
```