mobile device management
mobile application management
mobile content management

AirWatch SDK GPS Sample App - iOS 7

A Guide to Using the SDK for GPS Monitoring with iOS 7 Devices

Table of Contents

Introduction

This guide is to be used as a reference manual for the AirWatch GPS Sample Application source code that is deployed to customers/clients who have purchased the SDK. The restrictions on application backgrounding for the iOS platform makes granular GPS monitoring one of the biggest challenges for public applications. End-users that require high frequency GPS monitoring can utilize the AirWatch SDK to develop an internal application that will return GPS data to a centralized AirWatch console. Since this is an internal application, certain iOS restrictions that limit GPS for public applications do not apply here. The AirWatch SDK used in conjunction with the Sample GPS Application provided will provide increased visibility to the GPS coordinates and history for all managed devices running this application.

Requirements

- AirWatch MDM Agent from the App Store
- Apple Enterprise Developer Account
- AirWatch MDM Environment (Will not work with MAM Only deployments)
- AirWatch Software Development Kit (SDK)
- Mac with Xcode 5
- iOS 7 device with 3G/4G LTE (Devices with 3G/4G LTE Strongly recommended, see note below)

Note: This application is intended for use primarily with iOS 6 and 7 devices that have a 3G/4G LTE connection. However, wifi-only devices will collect GPS coordinates on the device, but will only upload them to the AirWatch server once the device reestablishes connection to wifi. For best results, it is strongly recommended that this application is applied to a device with a cellular data plan.

Implementation

After all of the requirements have been met and the GPS Sample Application has been obtained from your account manager, we are ready to begin implementing and deploying the application.

Step 1: Starting the Xcode Project

Inside the application zip folder you will see a file called GPS App.xcodeproj, double click on this file to launch the project into Xcode.

GPS App.xcodeproj

At this point, XCode should start up and open up the project that contains the source code for the GPS Application. Once again, please ensure that you are running Xcode 5 or else this application will not compile.

Step 2: Importing the AirWatch SDK Framework

Once we are in Xcode, we should arrive at a screen that is similar, but not quite the same to the one shown below in figure 1. In order to import the framework, we will need to navigate to the **Link Binary With Libraries** section by using these steps:

1. On the left-hand menu, select and highlight GPS App at the top of the folder structure.
2. This will then open up the project summary page. Click on Build Phases.
3. Once the Build Phases menu appears expand Link Binary With Libraries.

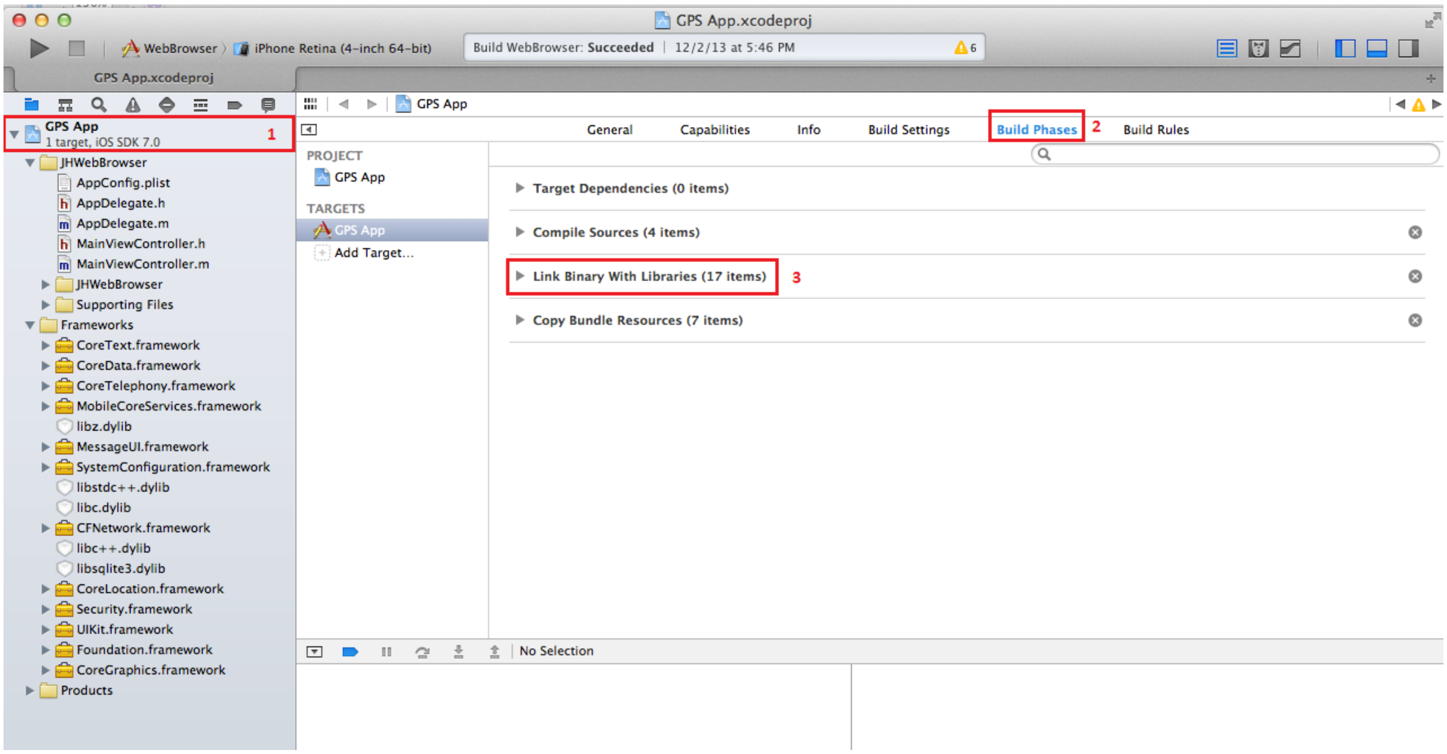


Figure 1. Xcode interface along with marked instructions on how to import a framework.

After expanding the Link Binary With Libraries section, you should see a list of currently imported frameworks. Towards the bottom of that list there is a **button with a plus symbol** on it. Click on this plus sign to bring up a modal with a list of frameworks you can add.

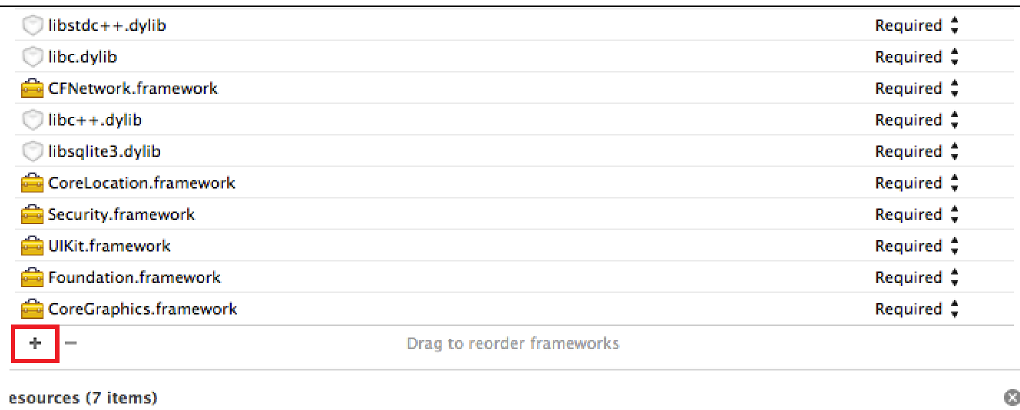


Figure 2. Link Binary With Libraries Section expanded and add framework button.

After the modal pops up, you will see a list of all the frameworks provided by the iOS SDK. Since the AirWatch SDK is a custom framework, we will need to import it. Do this by clicking on the **Add Other** button at the bottom left.

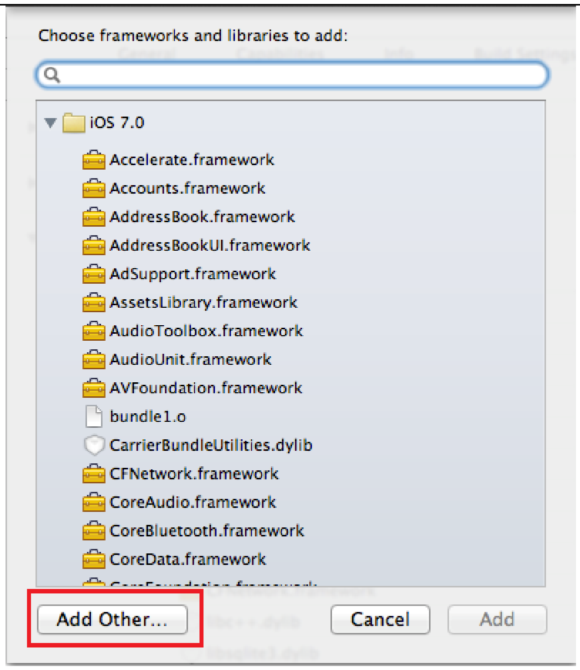


Figure 3. Add framework modal pop-up with option for adding a custom framework.

Navigate to the folder where the GPS Application is stored and within that folder is a folder named **AWSDK.framework**. Click on this folder once to highlight it and then click on open at the bottom right.

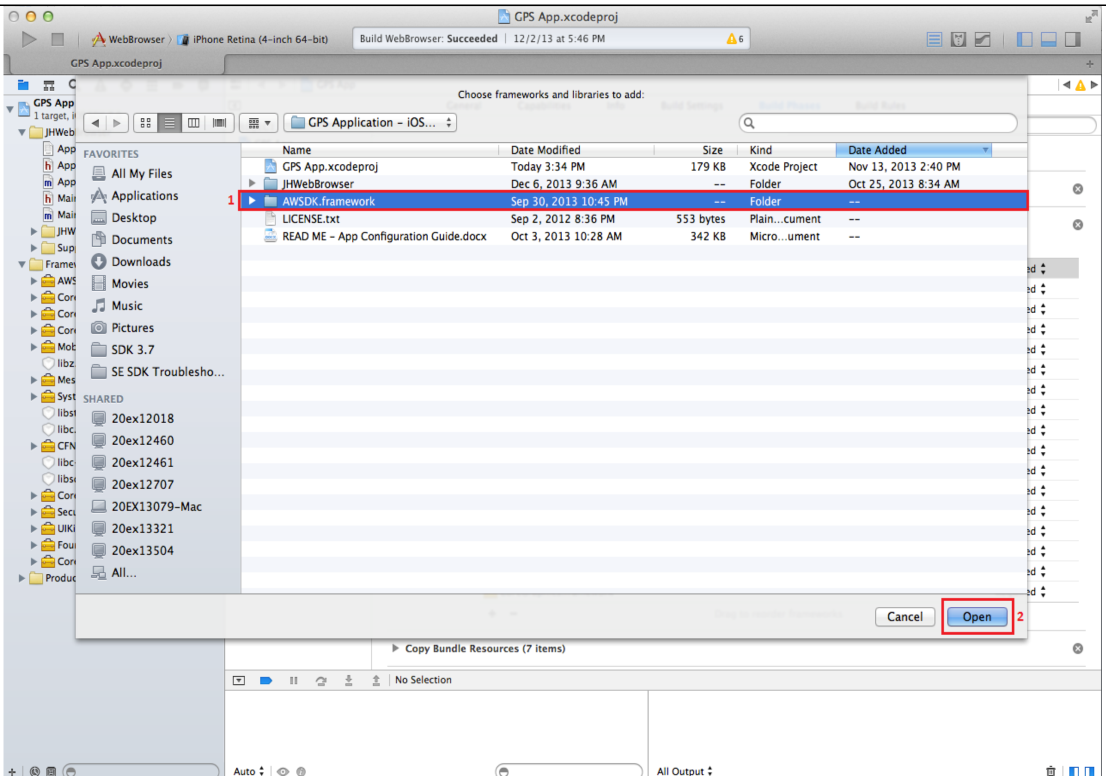


Figure 4. Figure showing the importing of the AirWatch SDK.

After adding the **AWSDK.Framework,** verify that the framework is now present on the list of frameworks under Link Binary With Libraries.

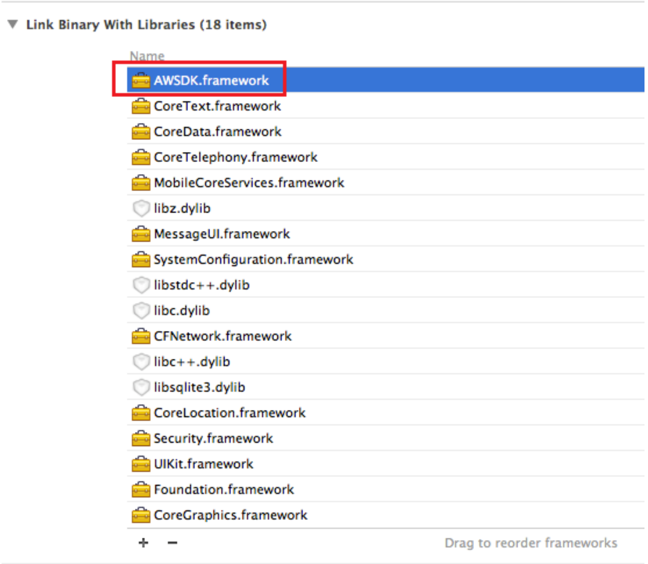


Figure 5. Link Binary With Libraries now showing the AWSDK.framework object on the framework list.

Step 3: Creating the Signing Certificate and Provisioning Profile

In order to sign and deploy any iOS application, a signing certificate and provisioning profile is required from Apple's developer portal. This section will provide instructions on how to create the certificate and profile. **If you are already familiar with this step and have the certificate/profile in hand then please move on to step 4**.

Creating a signing certificate and provisioning profile for distributing an internal app requires an Apple Enterprise Developer Account, not to be confused with the Standard Apple Developer Account. Once you have the credentials for accessing the Enterprise Developer Account, navigate to https://developer.apple.com. Click on iOS Dev Center and sign in using your credentials.

Figure 6. Login Page for the Apple Developer Portal.

Once you have successfully logged in to the portal. Locate the button titled **Certificates, Identifiers & Profiles** and click on it. You will then arrive at a page like the one shown in **Figure 7**.

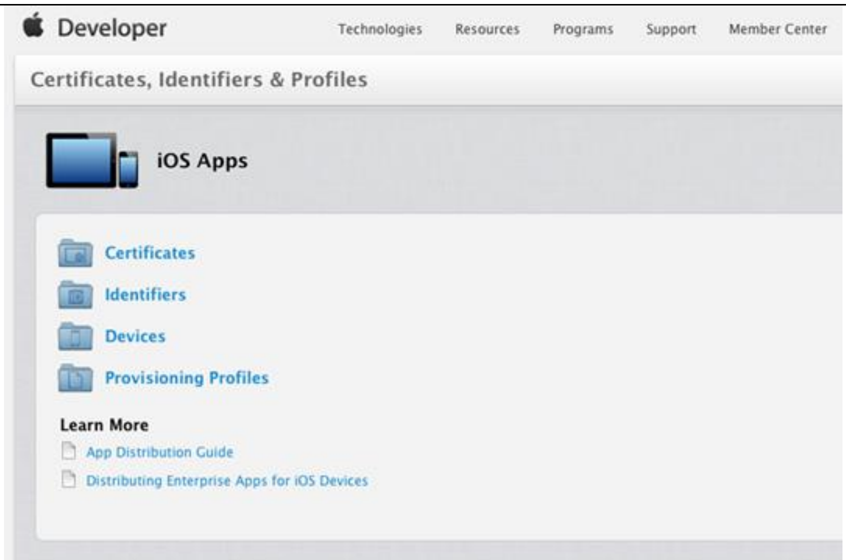


Figure 7. Certificates, Identifiers & Profiles Page on the Apple Developer Portal.

Click on **Certificates** and it will take you to a list of all your certificates in the portal. In the top right corner of the screen click on the **plus sign** to add a new certificate.

Select the **In-House and Ad Hoc** and click on **Continue** at the bottom of the page.

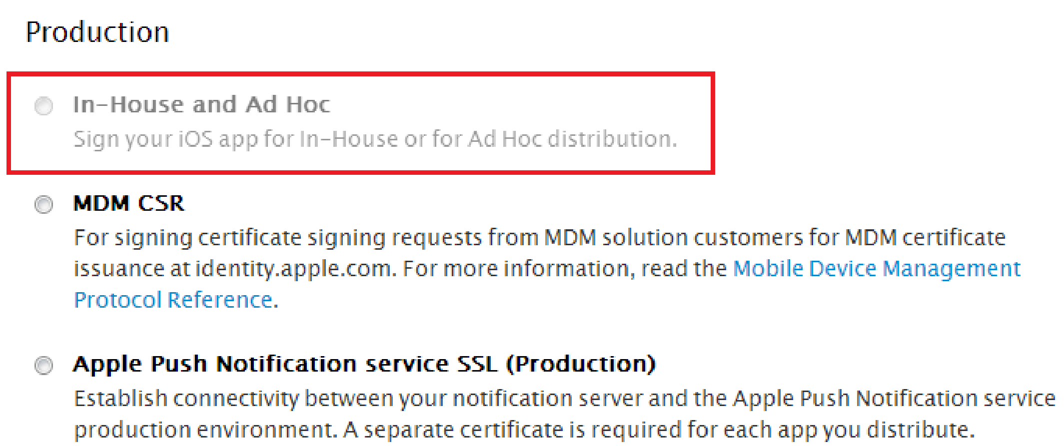


Figure 8. Production Certificate Selection List.

Note: Each Enterprise Developer Account can have a maximum of two valid In-House and Ad Hoc certificates at any given time. If there are already two existing distribution certificates then the In-House and Ad Hoc option will be grayed-out as seen in **Figure 8.**

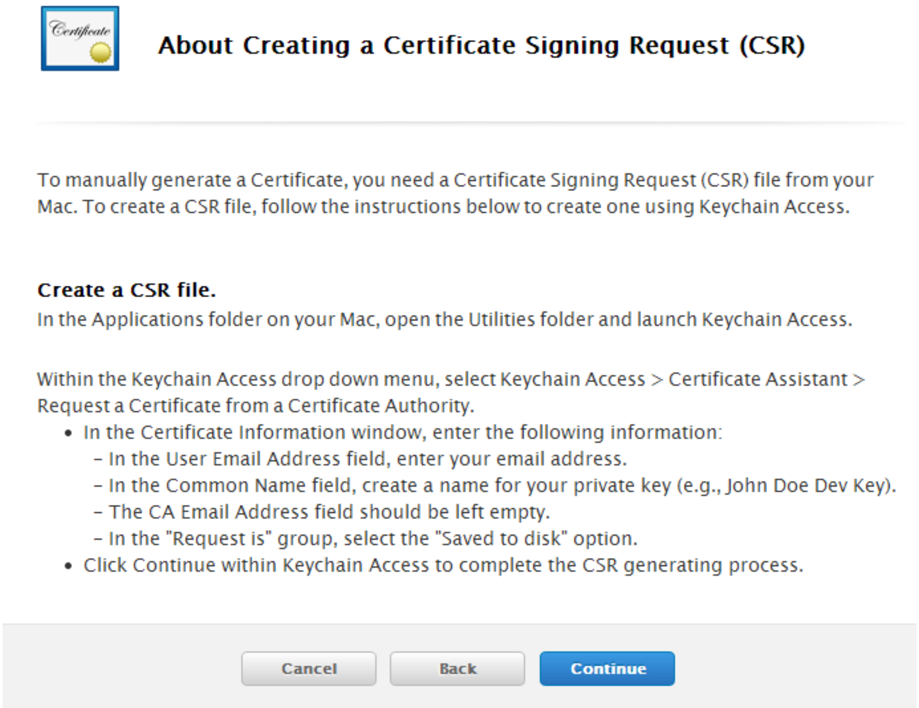


Figure 9. Certificate Signing Request (CSR) Generation Page.

Make sure you keep the browser session alive on the page shown in **Figure 9** as we'll need to come back to this page in just a little bit**.** The next step is to generate a certificate signing request for creating the certificate. This is done by using the native **Keychain Access** application on your Mac. Do this by clicking on the magnifying glass in the top right corner of the screen and searching for "keychain access".

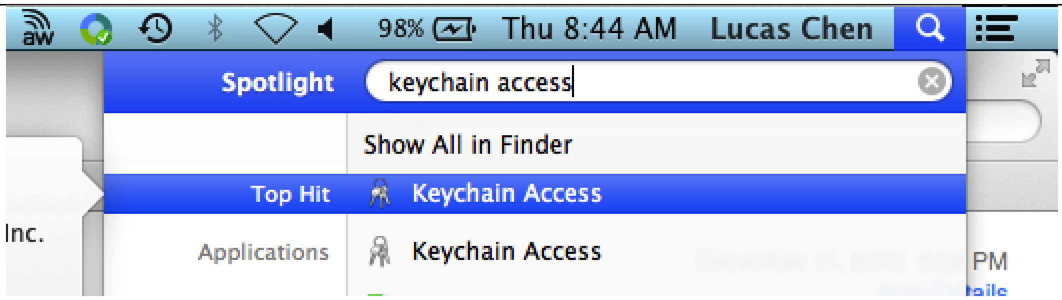


Figure 10. How to Launch Keychain Access on a Mac.

Once Keychain Access is open and is in the foreground. Click on the **Keychain Access** button on the Mac menu bar located in the top left corner of the screen. See **Figure 11** for help locating this option. If this option is not there, that means that your Keychain Access application is in the background. If this is the case then click on your Keychain Access application to bring it to the front. The Keychain Access button on the Mac menu bar should now appear.

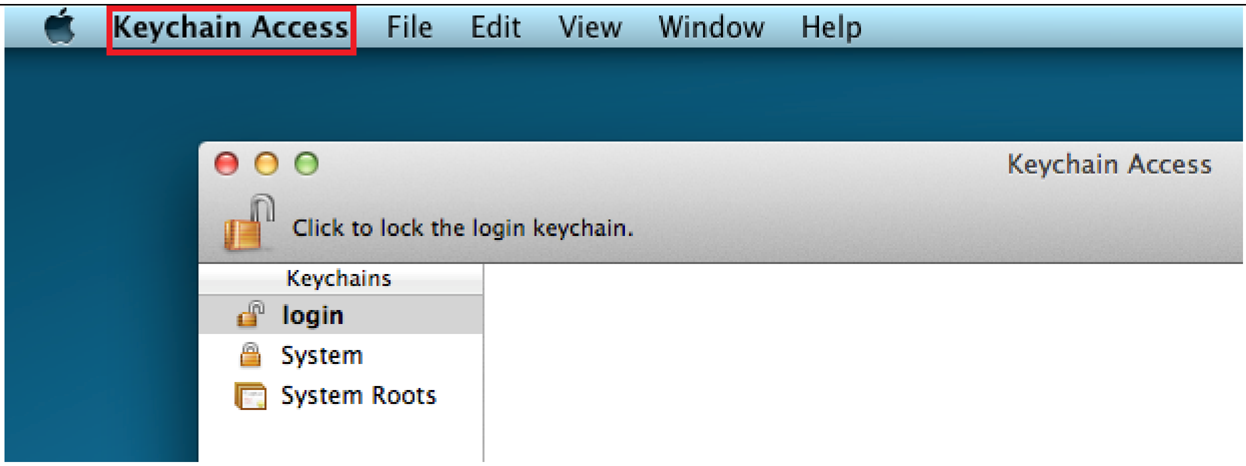


Figure 11. Mac Menu Bar with Keychain Access Button.

After clicking on Keychain Access in the Mac menu bar, select **Certificate Assistant** in the dropdown and then click on **Request a Certificate from a Certificate Authority.**

Note: Depending on your operating system, the wording for this option may vary slightly, but the general idea is to click on the option that says **Request a Certificate from …** and that should take you through the same workflow.

You will then arrive at a screen that asks you to input the **Certificate Information.** First select the radio button titled **Saved to disk.** Then enter a **User Email Address** to use as a reference for the certificate. The **Common Name** will be a general name for the certificate usually named after your company name. After these fields are filled in, your screen should look similar to **Figure 12** except with your own information.

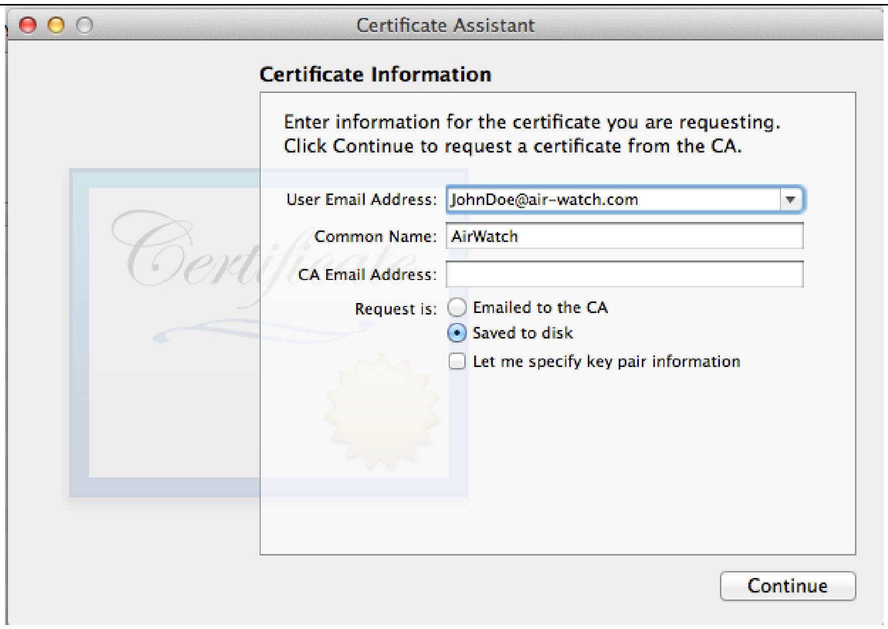


Figure 12. Certificate Assistance after Required Information has been Populated.

Next click continue and save the Certificate Signing Request (CSR) to a file location on your computer that you can remember. You will need to locate and use this file in just a moment. After you have successfully saved the CSR to your disk it is time to go back to the Apple Developer Portal Page shown in **Figure 9.** Once back on that page, click on the continue option and upload the CSR file we just created. After uploading the file, click on the **Generate** button to create the certificate. After the certificate generation has finished loading, click on **Download** button to download the generated certificate to your computer. Open up the folder where the certificate was downloaded to and either double click or drag the certificate into your **Keychain Access** application to import it. The file should have a .cer extension. Verify that the certificate with a name **iPhone Distribution: …** you created is now in your **Keychain Access** similar to in **Figure 13.**

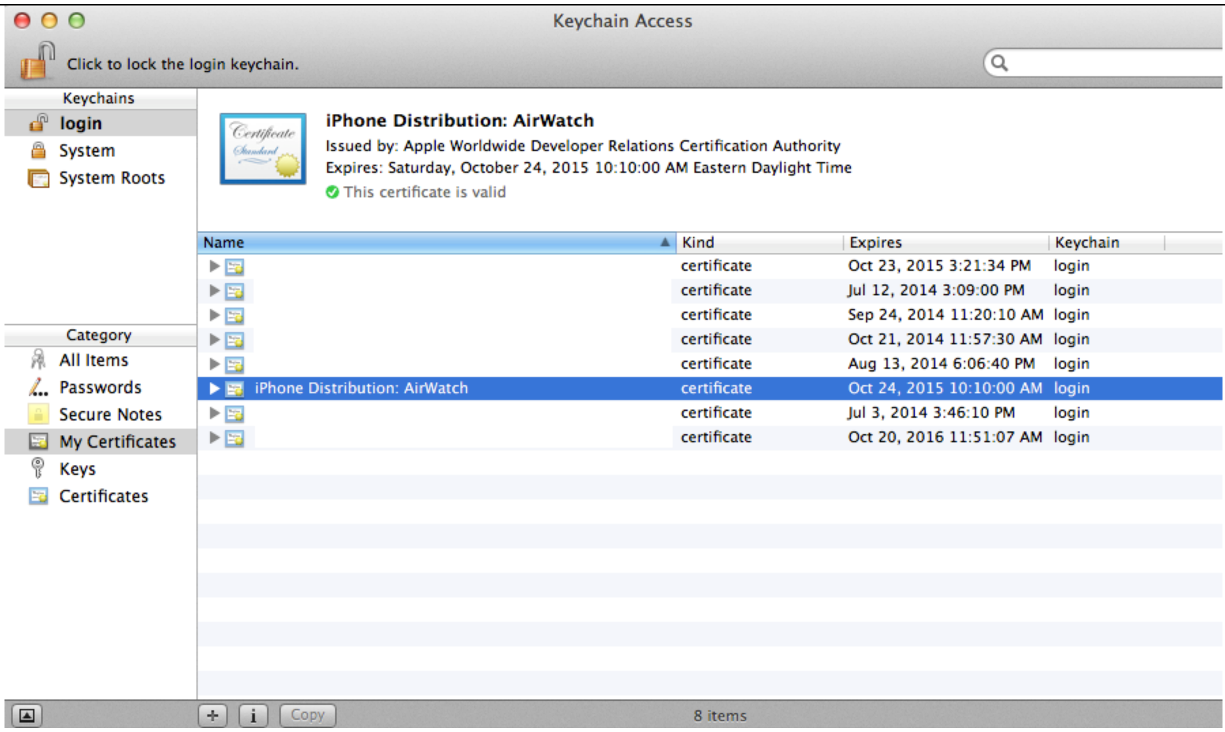


Figure 13. Keychain Access Showing the Certificate Generated by Apple.

After we've created our Apple distribution certificate, it is time to create an App ID for this application. Do this by going back to the **Certificates, Identifiers & Profiles** page back in the Apple Developer Portal and clicking on **App IDs** under the **Identifiers** section. Proceed with adding a new App ID by clicking on the plus sign at the top right portion of the page. Refer to **Figure 14 & 15** for additional guidance.

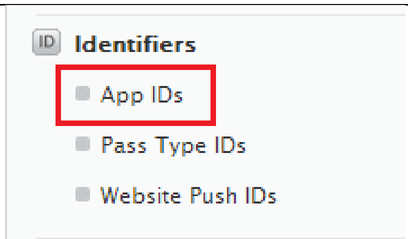


Figure 14. App ID Creation Page Button.

Figure 15. Create New App ID Button.

We'll then arrive at the **Registering an App ID** page where we'll need to create an **App ID Description, App ID Suffix** and declare the **App Services.**

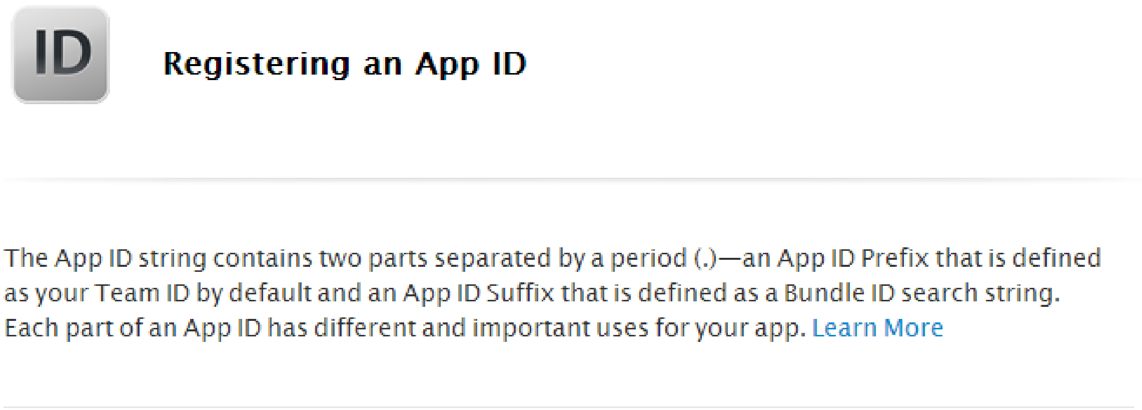


Figure 16. Application ID Creation Page.

Here is some information on what each field is for:

- **App ID Description –** This is a brief description you can write for the application ID to distinguish it from the other ones in your Apple Developer Portal. This serves no other purpose than for organization.
- **App ID Suffix –** This is going to be the bundle identifier for the application which is a unique identifier that's gets tied to each iOS application. The App ID Suffix usually takes a reverse-domain naming convention (i.e., com.air-watch.GPS). We have two options when creating an App ID Suffix, **Explicit** or **Wildcard:**
  - **Explicit –** This is for if you want to create just one specific Bundle ID with a specific name. (i.e., com.air-watch.GPS)
  - **Wildcard –** The wildcard option offers more flexibility in the bundle ID. This essentially creates a bundle identifier that allows for an infinite amount of variations. This application ID will have to contain a asterisk (*) to signify the wildcard. For example, if the Wildcard App ID **com.air-watch.*** is created then I can choose to sign my application with any bundle identifier string as long as it starts with **com.air-watch.** (i.e., com.air-watch.GPS, com.air-watch.GPS2, com.air-watch.GPS3)
- **App Services –** This is for specifying what kinds of Apple Services you want to enable for applications that utilize this bundle/application ID. If you disable iCloud then any application that is signed using this ID will not be able to leverage iCloud capabilities.

Now that we have an understanding of what each text field is for, go ahead and enter in an **App ID Description.**

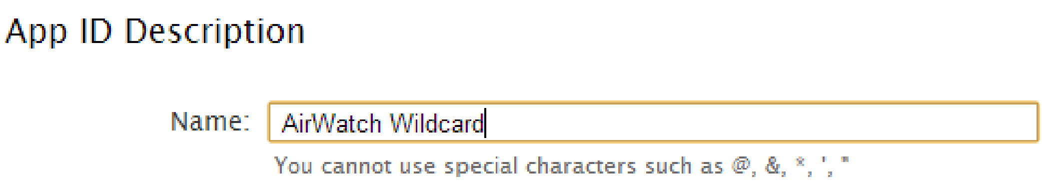


Figure 17. App ID Description Field when Registering an App ID.

Next, create either an **Explicit or Wildcard App ID Suffix.** It is recommended that you create a **Wildcard ID** to avoid having to create several different **Explicit** App IDs in case you want to deploy other applications in the future. It is common practice to name your **Wildcard App ID** in the following format: **com.companyname.*** where companyname is swapped out with the name of your own company or organization.

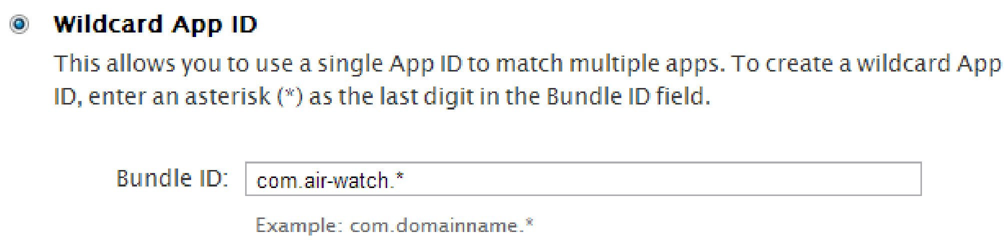


Figure 18. Wildcard App ID Suffix Field with Wildcard Identifier.

The last part of the App ID creation process is to define the **App Services.** For implementing the GPS App, you can just leave these settings as they are since none of these additional services are required for the GPS App to function.

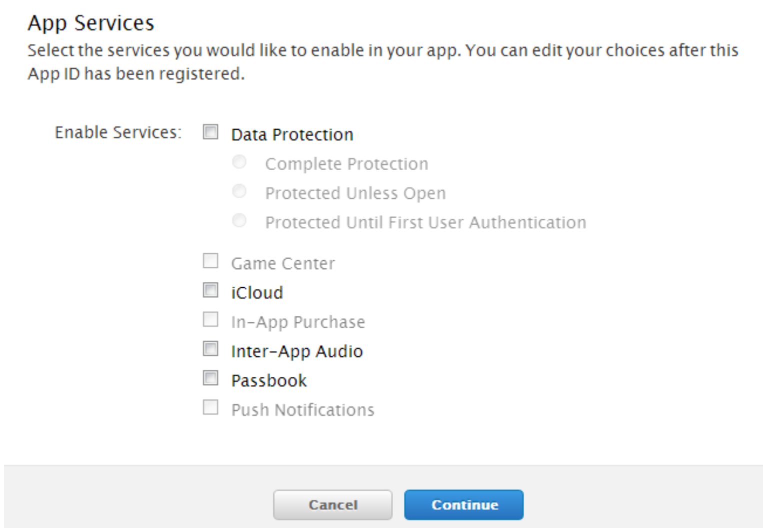


Figure 19. App Services Selection during App ID Creation.

Press **Continue** and follow any additional prompts to finish creating the application ID. Now that we have created both the certificate and the application identifier, we are ready to create the provisioning profile for distributing the application. Start by navigating back to the **Certificates, Identifiers & Profiles** page on the Apple Developer Portal and clicking on **Provisioning Profiles** on the left-hand-side menu and clicking the plus sign to add a new profile.

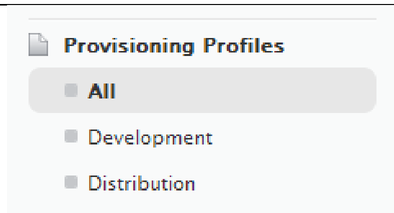


Figure 20. Provisioning Profile Page Button.

Figure 21. Create New Provisioning Profile Button.

After clicking on the add profile button, you will be taken to a screen that asks you to specify the profile type. In this case, we will want to select **In House** for distributing your application internally and then click continue.

Note: If there is no **In House** option, but rather an **App Store** option then that means you are actually using the standard Apple Developer account instead of the required Enterprise Developer Account.

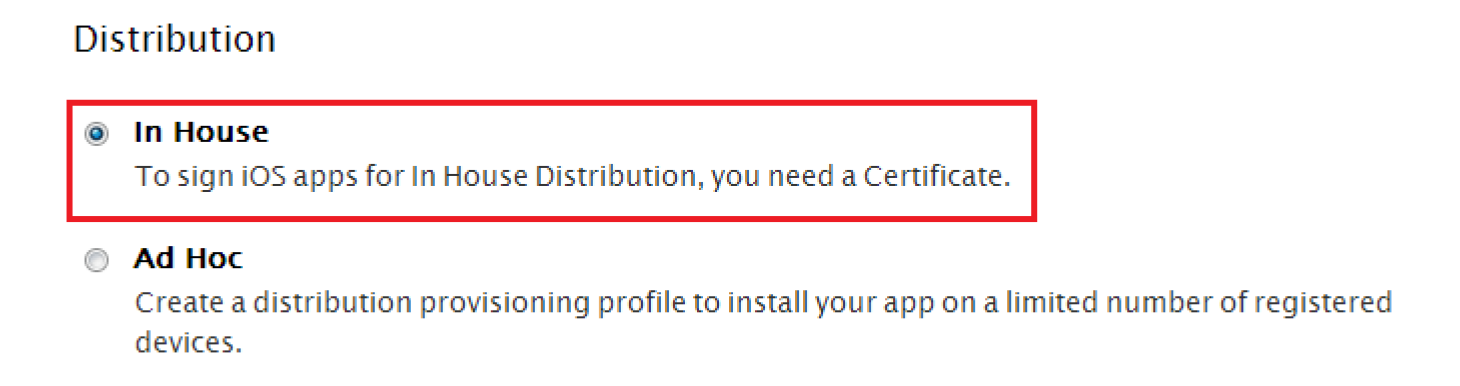


Figure 22. In House Option for Provisioning Profile Creation.

You will then be prompted to select an App ID to associate with this profile. Select from the dropdown box the **Application ID** you created earlier and click **Continue**. In the next screen, you'll need to select which certificate to use in correspondence with this profile. Select the **Distribution Certificate** you created earlier and click **Continue**.

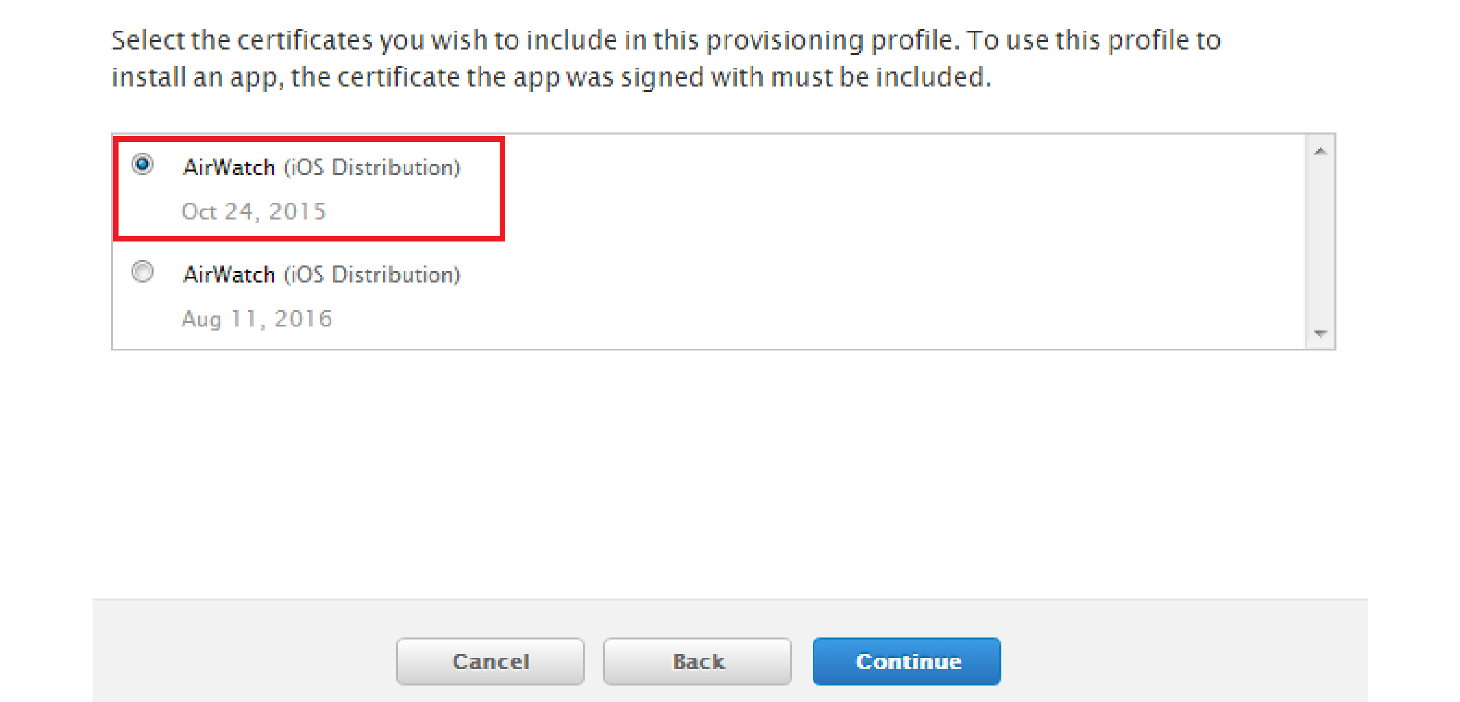


Figure 23. Certificate Selection Page When Creating a New Provisioning Profile.

Next, create a **Profile Name** to assign to this profile and click on generate. After the profile generation is complete, click on the **Download** button to download the new profile to your Mac local disk. Once the **Provisioning Profile** has finished downloading, locate the folder it is stored in and double click on the profile to add it to your library of profiles.

Step 4: Packaging and Signing the Application

After completion of the prior steps, it is time to build and package the application using the provided source code. We will be carrying out this portion within the Xcode Application. At the top left corner of your Xcode window next to where it says **WebBrowser,** ensure that iOS Device or the name of your device (if a device is connected to your Mac) is selected as opposed to one of the simulators.

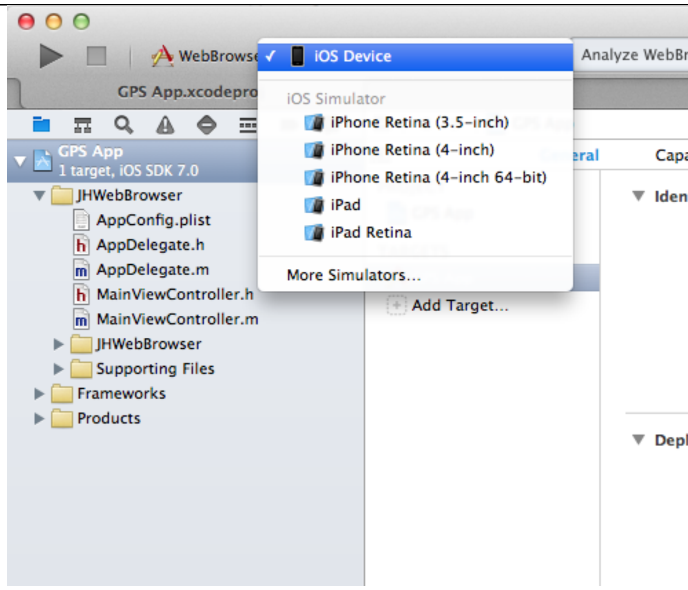


Figure 24. Menu for selecting the deployment device/simulator to deploy to.

Select the project from the leftmost vertical menu to bring up the Project Settings Pane. Click on the **General** button at the top to view the **Settings Summary**. In the **Identity** section, there is a field titled **Bundle Identifier,** create a bundle identifier that corresponds to the application ID suffix you created earlier in the app developer portal. After assigning the **Bundle Identifier**, click on the **Team** dropdown box and select **Add an Account.** (See **Figure 25** for more details)

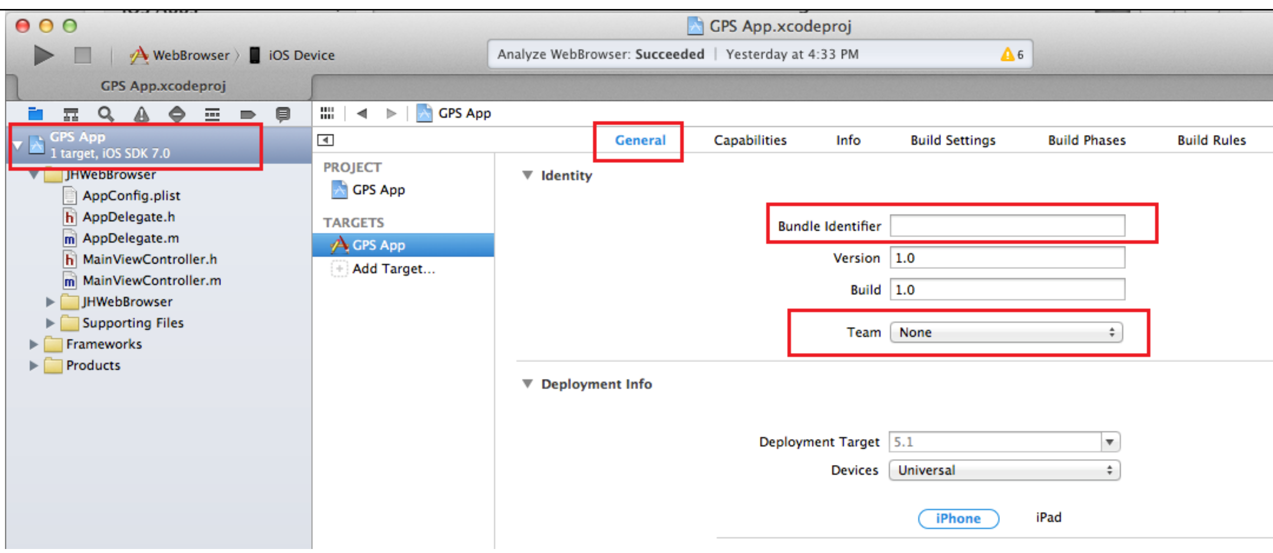


Figure 25. Project Settings Pane with Bundle Identifier and Team Dropdown Box.

Enter in the Apple ID credentials that are tied to your enterprise developer's account. After the account has been successfully added into Xcode we are ready to build the code into an .ipa file. You should now take this time to upload any app icons and change any of the names in the settings pane in order to customize the aesthetics and titles for your application. Once you're done editing the settings, it is time to package it all up. With Xcode in the foreground, click on **Product** at on the Mac Menu Bar at the top of the screen. Then Click on **Archive** in the dropdown menu to build and package the application. (See **Figure 26** for more details)

> **Note**: If the **Archive** button is grayed-out then that is most likely due to an iOS Simulator being selected as the target rather than an iOS device. You can fix this by ensuring the action illustrated in **Figure 24** has been completed.

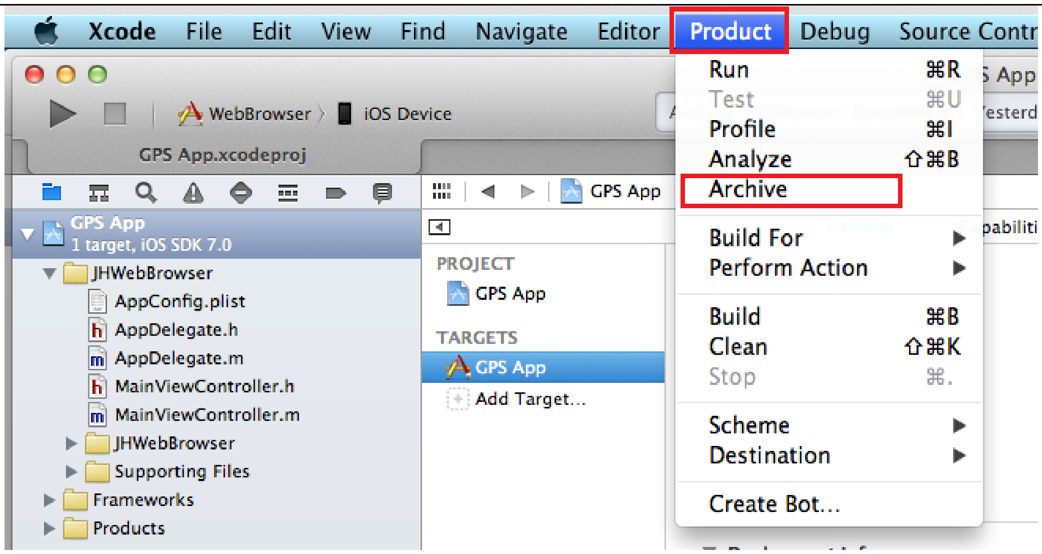


Figure 26. Product Menu in Xcode for Archiving the Source Code into an iOS Application.

After the source code has been successfully archived, a new window titled **Organizer – Archives** will appear that shows you the app you just archived. Click on the **Distribute** button to wrap up the signing process.

Figure 27. Distribute Button for Signing and Distributing the Archived Application.

After clicking on **Distribute**, a new window will pop up that asks you to **select the method of distribution** for this application. Select **Save for Enterprise or Ad Hoc Deployment** and click **Next**.

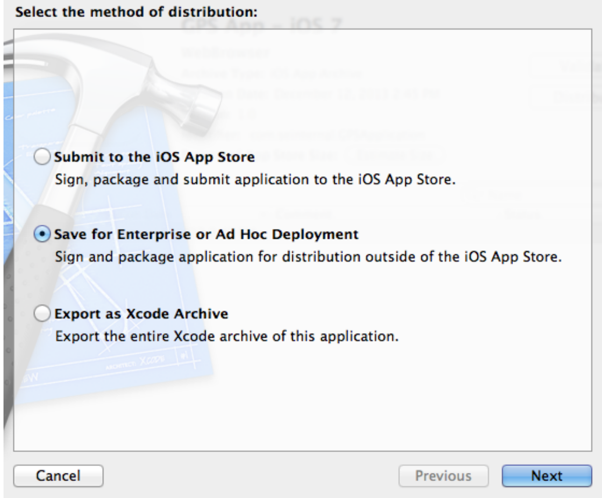


Figure 28. Distribution Method Selection Window.

After selecting the distribution method, Xcode will then fetch all the profiles you have the option to sign the application with. Select from the dropdown list the **provisioning profile** you created earlier in the Apple Developer Portal and then click **Export** which will export the .ipa file. Save the file in a location you will remember since you will need to locate it for uploading to the AirWatch Web Console. The application has now been signed and packaged for internal deployment across the organization/enterprise.

Deployment

At this point, the source code has been packaged into an iOS Application or an .ipa file and is ready to be deployed to your end users. Open up a web browser and log in to your AirWatch Web Console. In the left vertical menu, navigate to **Apps & Books** > **List View** > **Add Application.**

Note: If you are running AirWatch 6.5 or below then you will navigate to **Menu** > **Applications** > **Add Application.**

Once the **Add Application** screen appears, click upload and select the .ipa file that contains the iOS application we created from Xcode and continue through until we reach the application info page.

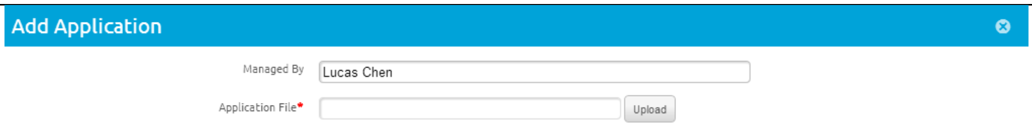


Figure 29. Add Application Window.

Verify that all the information regarding the application is correct under the **Info** tab in the horizontal menu at the top of the window.

Figure 30. Add Application Menu with Info Tab Selected.

After the application information has been verified to be correct, click on the **Assignment** tab and assign the application to a set of devices using **Smart Groups** to filter the application to only iOS 7 devices. If no groups have been created in the past, then click on **Create New Smart Group** and create a new group that encompasses the desired users/devices. After the assignment has been made, click on the **Deployment** tab in order to configure the application's settings. The first thing we'll want to do here on this window is to set **AirWatch uses AirWatch SDK** to **Yes** and check **Send Application Configuration**. Once this has been done the screen should look similar to **Figure 31**.

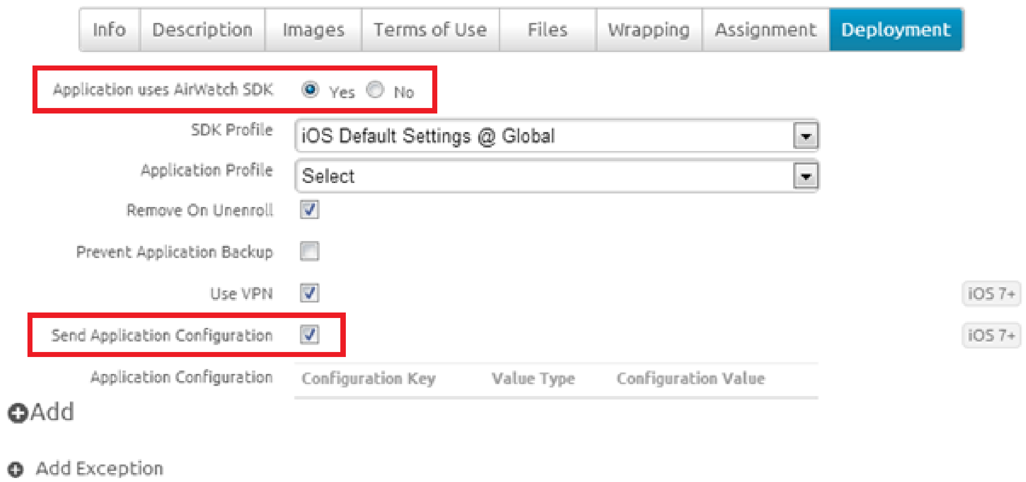


Figure 31. Application Deployment Page with AirWatch SDK and Application Configuration Enabled.

Do **NOT** publish the application just yet. Continue to read below in the **Configuration** section of this guide for details regarding configuring the Application.

Configuration

Note: The configuration feature is only available in console versions 6.5 and beyond. If you are running AirWatch 6.4 or below then you will need to edit the AppConfig.plist file within Xcode prior to build the application in order to set the configurations.

Within the GPS Application, there are 3 main settings that can be configured:

- **Location Sensitivity(meters)** – This setting defines how far the device should travel before a new location is stored and sent to AirWatch. If no value is defined by the user then this setting will be configured to 1,600 meters which is approximately 1 mile.
- **Sampling Interval(minutes) –** This setting defines how often the application will check in with a new location to the AirWatch server. The application will also run jailbreak detection at this specified interval and send the result back to the server. If no value is defined by the user then this setting will be configured to check in with the console server at an interval of once every 15 minutes.

Note: If the device has not changed location since the last sample, then the location will not be sent to the server. This is to avoid redundant coordinates from appearing on the map.

- **Landing Page URL –** The user interface on the application is a kiosk mode web browser. This single page web browser can be configured to point to a specific URL (i.e., a company website). Whenever the user opens the application, the application will attempt to navigate to this web page. If no value is defined by the user then this setting will be set to point to https://www.air-watch.com.

Step 1: Configuring the GPS Application Settings

Under the **Deployment** tab with both **Application uses AirWatch SDK** and **Send Application Configuration** are enabled, you should see a section on that page titled **Application Configuration** along with **Configuration Key, Value Type** and **Configuration Value** that looks similar to **Figure 32**.

Figure 32. Application Configuration Section.

We will want to make this section mimic what you see in **Figure 32.** You can add more configuration rows by clicking the **Add** button. Create 3 configurations with the keys **interval, sensitivity** and **url** spelled exactly as you see (case-sensitive).

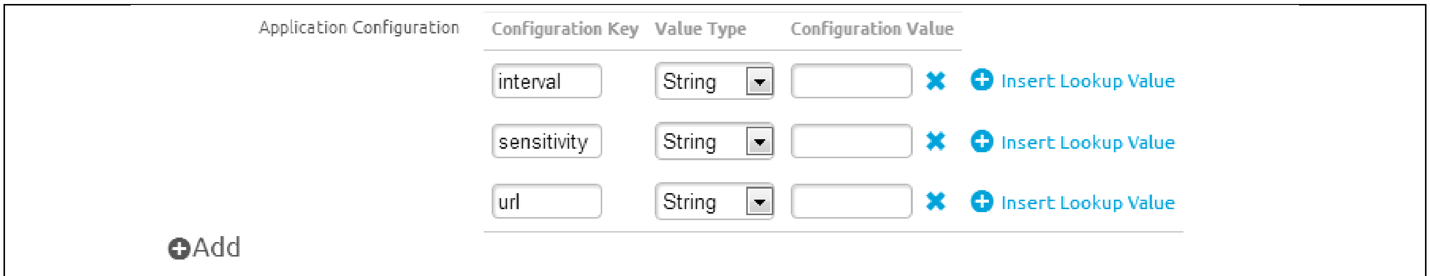


Figure 33. Three Application Configurations with Defined Keys.

Now it is time to set values for these configuration keys.

- Set the desired sampling interval by inputting a number greater than 0 in the **Configuration Value** for the **interval** key.
- Set the desired location sensitivity by inputting a number greater than 0 in the **Configuration Value** for the **Sensitivity** key.
- Set the desired landing page url by inputting the url scheme and fully qualified domain name in the **Configuration Value** for the **url** key. (i.e., https://www.air-watch.com)

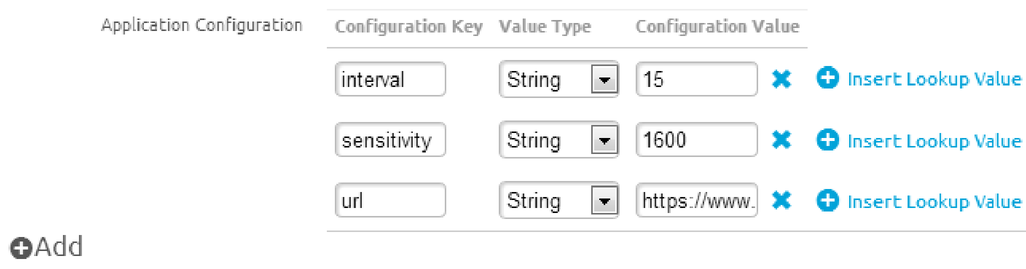


Figure 34. Application Configuration with Both Keys and Values Set.

Step 2: Save and Publish the Application

After you have set the configuration keys and values, it is time to **Save & Publish** the application. Verify that the application is being deployed to the intended devices and continue. Your users/devices will be prompted to install the application on their devices depending on whether the deployment type was set to auto or on-demand.

Updating the Configurations

After the configuration has been set and the application has been deployed, you can update the configuration setting by navigating to the application listing page and editing the GPS Application.

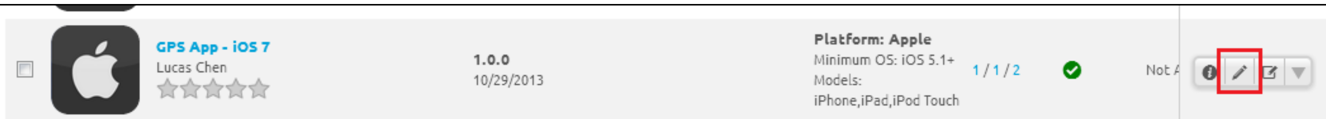


Figure 35. Edit Application Button to Change Application Configurations.

Once the **Edit Application** window is up, switch to the **Deployment** tab to edit the application configurations you set earlier. After you have updated your configuration and saved the application again, you will have to issue the command to send the new configurations down to the device. Do so by clicking on the **Triangle Shaped Button** shown toward the right in **Figure 35** and clicking **Send Application Configuration.**

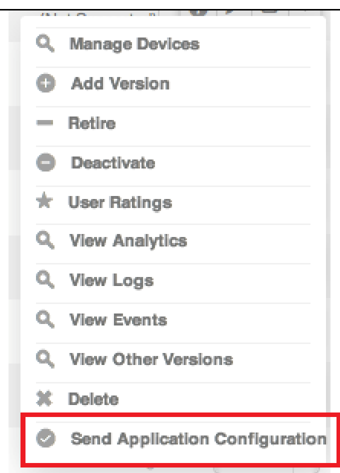


Figure 36. Send Application Configuration Command Button.

After the new application configurations are sent down to the device, you will have to terminate and restart the device for the new settings to take effect inside the application.

Note: The application will retain the old configurations if the **Send Application Configuration** is not pressed and the application is not terminated and restarted on the device.

Running the Application

Once the GPS application has been successfully pushed down and installed on a device, ensure the AirWatch Agent is present on the device and launch the application. Upon launching the application for the very first time after the install, it will flip into the AirWatch Agent to request authentication. This is a one-time initialization process that takes place after the initial installation of the app.

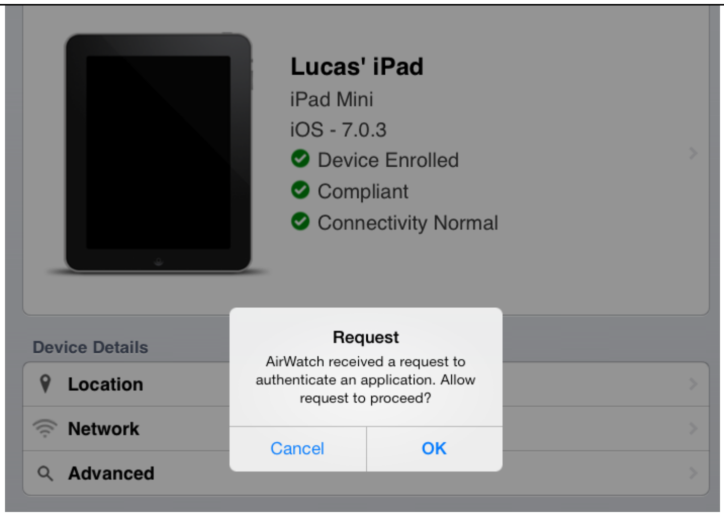
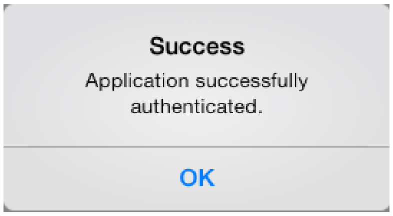
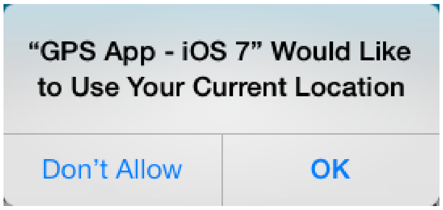


Figure 35. AirWatch Agent Requesting Authentication.

Click **OK** to proceed with the authentication and follow the additional prompts on the screen. Once the authentication is successful, the AirWatch Agent will flip back into the GPS Application.



Once the GPS Application is back in the foreground, you will be prompted as to whether or not to enable location services. Press **OK** to enable **location services** otherwise the application will not report location data back to the AirWatch console.



After this is done, the application will send GPS data and run compromised detection as long as it is running the background and not terminated. It will restart itself into the background if the device is restarted unless the user terminates the application prior to restarting.

Analysis/Reporting

Once the GPS application is alive and running on your set of devices, you can now log into your AirWatch console to view the GPS data gathered by your devices running the GPS Application. You can do this by navigating to the **Device Dashboard** and then selecting a device from the **List View** to bring up the **Device Details** page. Next, navigate into the **Location** tab to view the map and GPS history of the device.

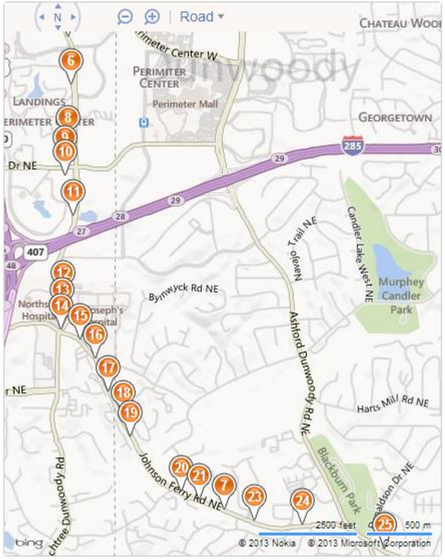


Figure 36. Sample Data Collected from a Device Running the GPS Application.

You can also run reports on the GPS Data on all devices enrolled in your AirWatch environment. Run reports by navigating to **Menu > Reports** if you are on AirWatch 6.5 and **Hub > Reports & Analytics > Reports >List View** in AirWatch 7.0. Once inside the reports page, run the **Device Location Log** report to get the GPS Data for all your specified devices.

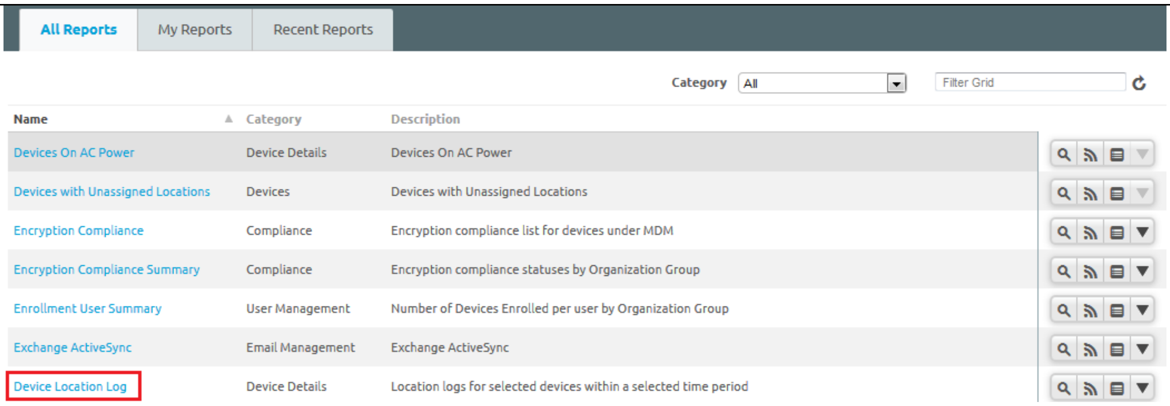


Figure 37. Device Location Log Report.

Frequently Asked Questions (FAQ)

- **My Device is checking in at the specified intervals, but I am not getting new location data.**
    - This is usually because the device has not moved location, if the device is checking in, but the location doesn't change/update then it is most likely due to the fact that the device has not yet accumulated a new location. This is to prevent redundant data points.
- **The authentication in the AirWatch Agent is failing when the application is launched for the first time.**
    - There are two possible causes for this. Ensure your device is enrolled in the correct environment and the application is managed by the AirWatch(deployed internally) server with the same bundle ID as the application on the device. Also ensure that this application is being run on an iOS 7 device and not iOS 6.
- **I restarted my device and it's no longer reporting in to the AirWatch Web Console.**
    - Upon restarting your iOS 7 device, you must unlock your device once after the restart in order for background application processes to boot up. If unlocking the device does not start the application up again then terminate and restart the GPS application.
- **Is there a way I can prevent the user from terminating the GPS application?**
    - The only way to prevent your users from terminating the GPS application is to lock the device into single-app mode. Other than that, the iOS operating system does not allow an application any means to prevent termination.
- **Will the application continue to collect GPS data if the user terminates the application by double tapping the Home button and swiping up on the app?**
    - No, the app will not be able to send data if the user kills the application. However, if the user backgrounds the application by pressing the home button once or switches to another app then the GPS application will continue to send data.
- **How will the app work on WIFI-only devices when the device is changing location without a WIFI connection?**
    - WIFI only devices will still collect GPS information and store them on the device when offline. Once the device reconnects to WIFI, it will check in at the next scheduled sampling time and report all of the coordinates that were stored offline.
- **Why do I need an enterprise developer's account as opposed to a regular developer's account?**
    - In order to deploy an application internally, the app must be signed with an enterprise developer's account. If you sign it with a regular developer's account then the application will not install successfully on the end-user devices.