

---

# **App Signing for MACS**

---

<b>App Signing for MACS</b> .....	3
Overview .....	3
Prerequisites .....	3
Steps to Complete .....	3
Registering an Apple Developer Account .....	3
Installing Apple Certificates .....	4
Generating a Signing Certificate .....	4
Creating an Application ID .....	7
Generating a Provisioning Profile .....	9
App Signing for Android .....	14
Registering an Android Developer Account .....	14
Generating a Signing Certificate .....	14
Additional Resources .....	15

---

# App Signing for MACS

## Overview

Sugar's Mobile Application Configuration Service (MACS) enables partners and eligible customers (i.e., Enterprise and Sell Premier) to create and distribute custom-branded versions of the SugarCRM mobile app. Before downloading your app from MACS for distribution, you must provide MACS with a certificate and other information in order to digitally sign the app. App signing allows the client (i.e. phone or tablet running your app) to identify who signed the app and to verify that it has not been modified since you signed it. The certificate serves as a virtual fingerprint that uniquely associates the app to you. This also helps iOS and Android ensure that any future updates to your app are authentic and come from the original author.

For more information on using MACS, please refer to the [Mobile Application Configuration Service User Guide](#).

## Prerequisites

To use MACS, you must be in one of the following authorized user groups:

- Instances running a [supported version](#) of Sugar Enterprise, Enterprise+, Ultimate, Serve, Sell Advanced, or Sell Premier
- SugarCRM's OEM partners (for use and distribution with their authorized OEM solution)

**Note:** SugarCRM partners who are creating a re-branded mobile app for a customer must log in and do so only under that customer's account. Partners are not authorized to create and store these apps under their own Sugar accounts.

## Steps to Complete

### App Signing for iOS

There are multiple steps you need to take to successfully app sign for iOS before you will be able to distribute the application via the Apple App Store, an internal enterprise store, or an MDM account. The following sections describe each of these steps in detail. For more details on any of the steps, please refer to the [Apple Developer](#) documentation on Apple's developer site.

---

## Registering an Apple Developer Account

Apple offers two levels of developer accounts: Individual or Enterprise. An Enterprise account allows you to distribute applications internally or via an MDM vendor in addition to distributing via the public store. An Individual account only allows you to distribute via the public store. Decide which option is right for you, navigate to [Apple's developer site](#), click "Create Apple ID", and follow the instructions to register for your Apple developer account.

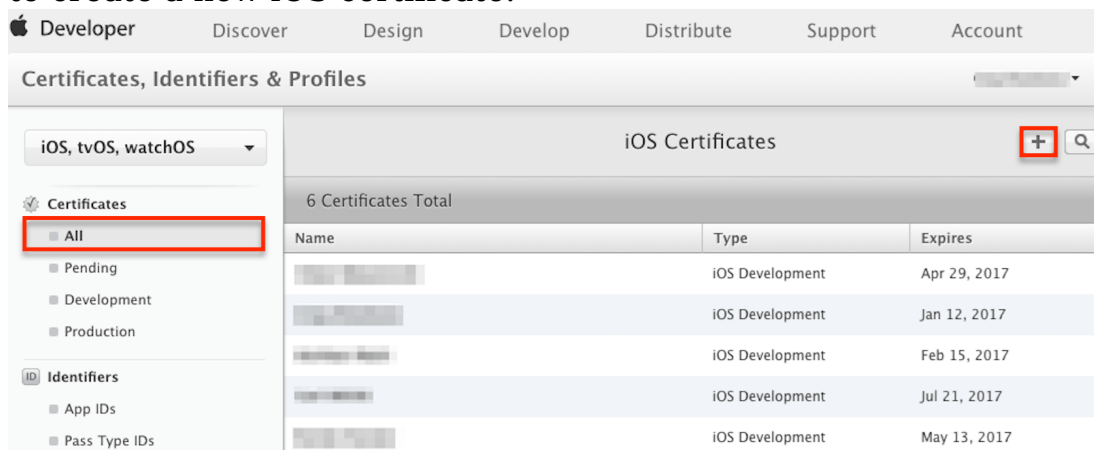
## Installing Apple Certificates

Your keychain must contain Apple root and intermediate certificates in order to successfully generate a signing certificate. Apple certificates are automatically installed when you install Xcode. If you do not have Xcode installed or if you do not wish to have it installed on your computer, then you must download the required certificates from the [Apple PKI](#) page on Apple's website.

## Generating a Signing Certificate

Once the Apple certificate is installed on your computer, you can begin generating a signing certificate. To do this:

1. Log into your Apple developer account.
2. Navigate to the Certificates, Identifiers, & Profiles section.
3. Click "All" under the Certificates panel on the left then click the Plus icon to create a new iOS certificate.



4. From this page, select a certificate type:
  - Development certificates do not allow publishing to stores but are useful if you wish to install and test the app on your device. Select "iOS App Development" to create a testing certificate.

---

## Development

- iOS App Development**  
Sign development versions of your iOS app.
- Apple Push Notification service SSL (Sandbox)**  
Establish connectivity between your notification server and the Apple Push Notification service sandbox environment to deliver remote notifications to your app. A separate certificate is required for each app you develop.

- A production certificate (a.k.a. distribution certificate) is required if you want to publish your app to stores. Depending on your developer account type, there are two different options for production certificates:
  - **App Store and Ad Hoc** : This option is visible for users with an Individual developer account type. Select "App Store and Ad Hoc" to create a distribution certificate.

### Production

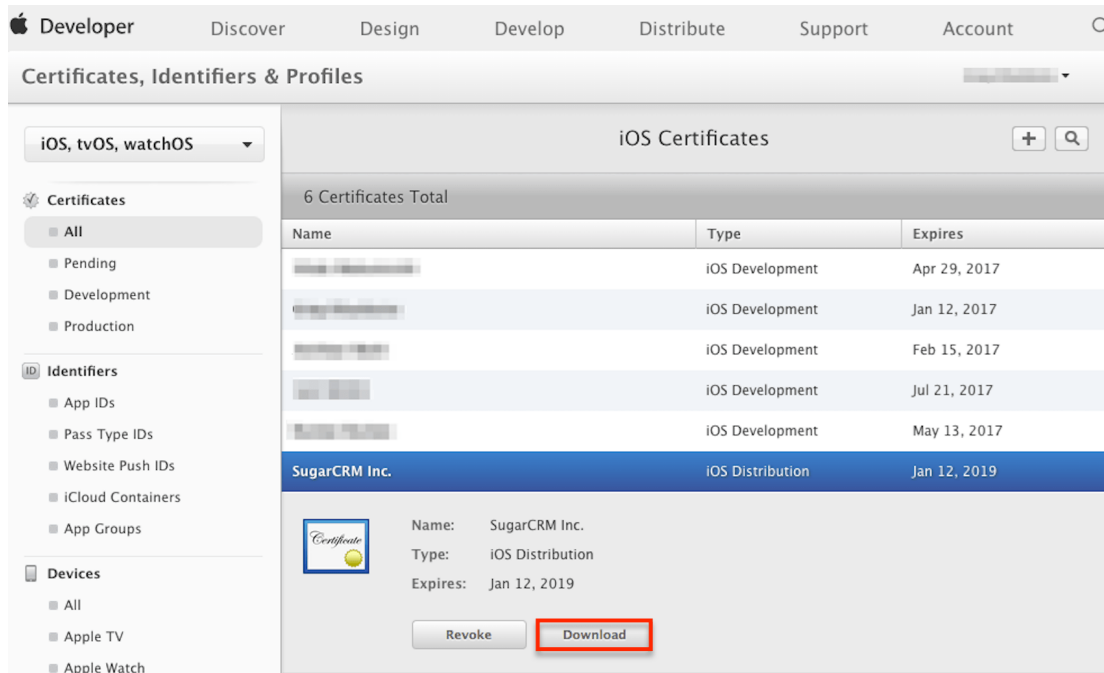
- App Store and Ad Hoc**  
Sign your iOS app for submission to the App Store or for Ad Hoc distribution.
- Apple Push Notification service SSL (Sandbox & Production)**  
Establish connectivity between your notification server, the Apple Push Notification service sandbox, and production environments to deliver remote notifications to your app. When utilizing HTTP/2, the same certificate can be used to deliver app notifications, update ClockKit complication data, and alert background VoIP apps of incoming activity. A separate certificate is required for each app you distribute.

- **In-House and Ad Hoc** : This option is visible for users with an Enterprise developer account type. Select "In-House and Ad Hoc" to create a distribution certificate.

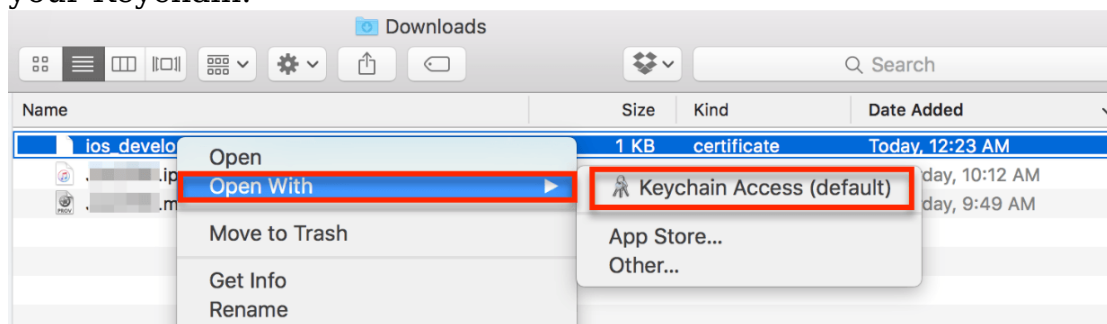
### Production

- In-House and Ad Hoc**  
Sign your iOS app for In-House or for Ad Hoc distribution.
- Apple Push Notification service SSL (Sandbox & Production)**  
Establish connectivity between your notification server, the Apple Push Notification service sandbox, and production environments to deliver remote notifications to your app. When utilizing HTTP/2, the same certificate can be used to deliver app notifications, update ClockKit complication data, and alert background VoIP apps of incoming activity. A separate certificate is required for each app you distribute.

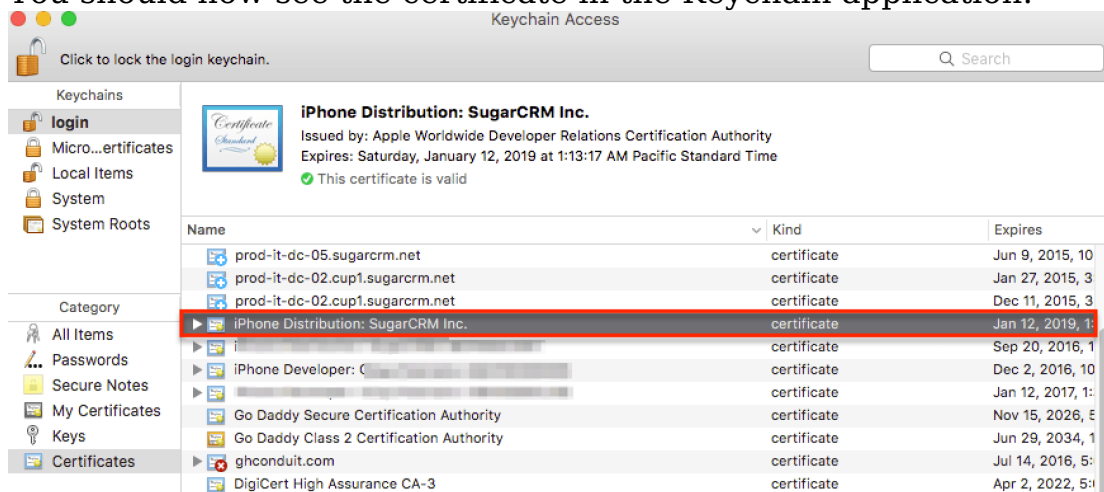
5. After selecting the certificate type that is best for your situation. Click "Next", and follow Apple's instructions for creating, uploading, and approving a certificate request. You will then see it available for download in the Certificates section.
6. Download the new certificate and install it onto your keychain. To do this, click "Download" to download the certificate onto your Mac computer then double-click on the file.



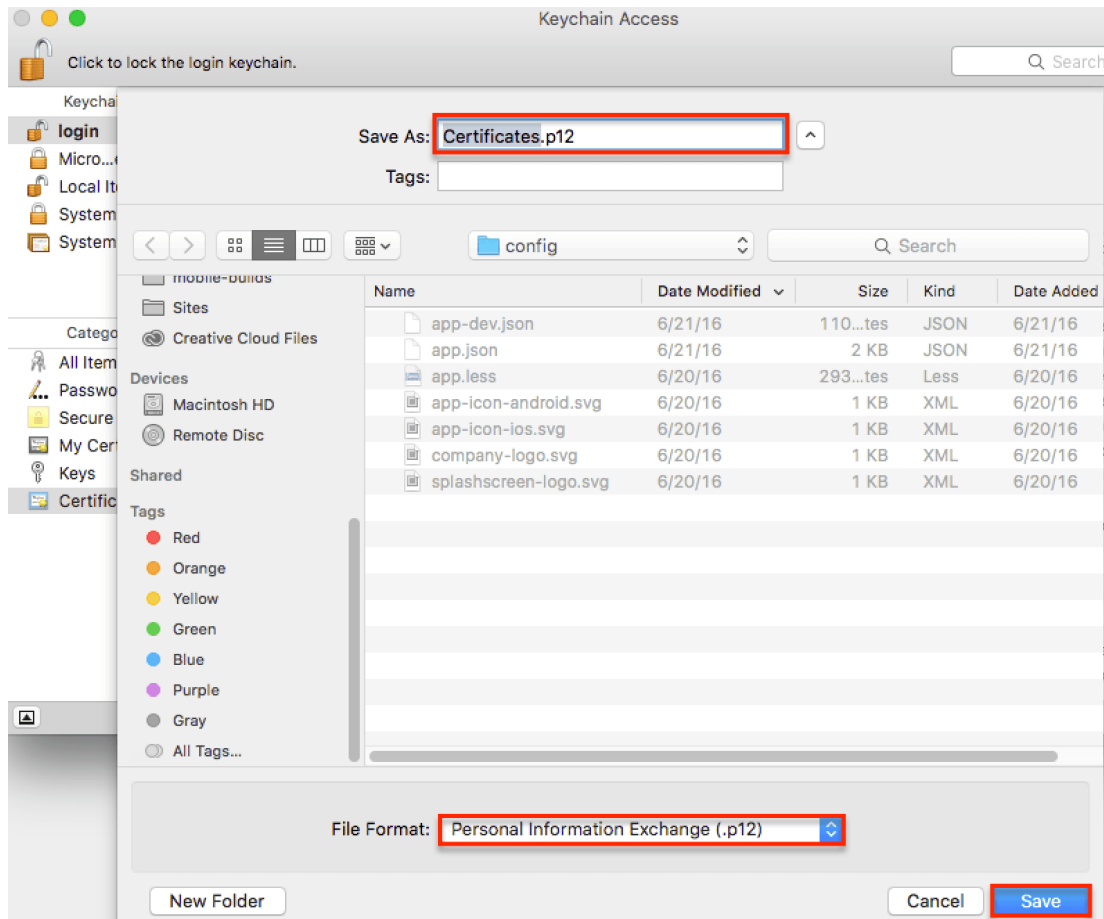
- Open your Downloads folder then right-click on the file and select "Keychain Access" from the Open With menu to install the certificate into your Keychain.



You should now see the certificate in the Keychain application:



- Export the certificate in .p12 format by right-clicking the certificate's name via the Keychain list and selecting "Export..." from the menu.
- From the Export File dialog, enter a name for the certificate and then select "Personal Information Exchange (.p12)" as the file format.

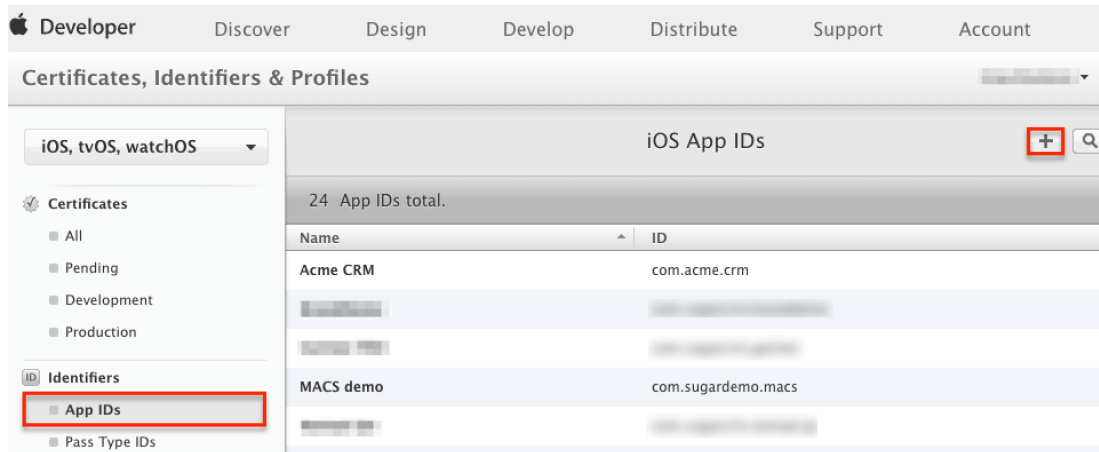


10. Click "Save" to save the file to your local file system.

## Creating an Application ID

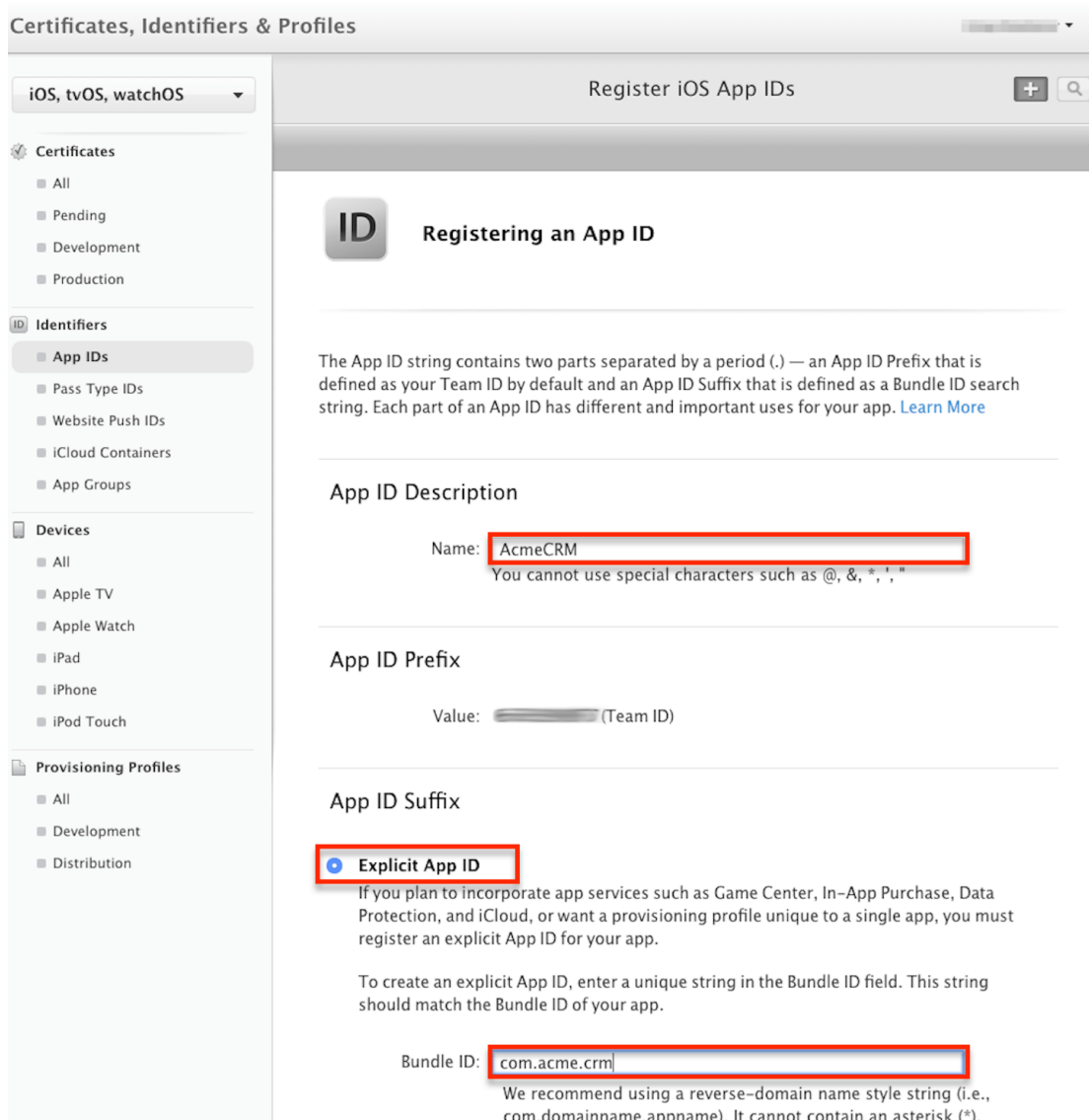
With the certificate successfully saved on your computer, you can begin creating an application ID. The app ID is a string that contains a team ID and a bundle ID which is used to identify one or more apps from a development team. The following steps explain how to create a unique app ID.

1. Log into your Apple developer account.
2. Navigate to the App IDs section and click the Plus icon to create a new ID.

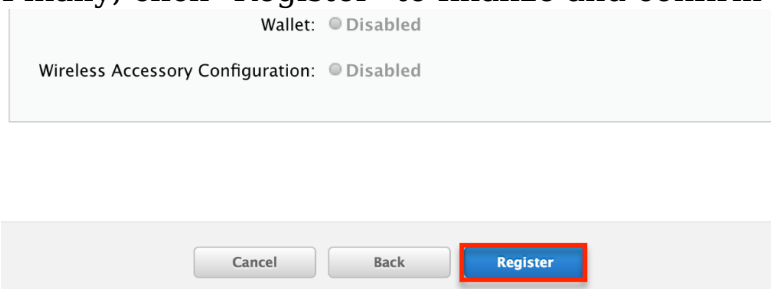


3. Enter your app's name in the Name field and specify an application ID in the Bundle ID field according to Apple's guidelines. Please note that if you are planning to use in-house distribution, then you must select the radio button for "Explicit App ID". You can also enter a wildcard app ID if you have multiple apps and wish to use one provisioning profile for multiple app IDs. For more information on wildcard IDs, please refer to the [iOS Developer](#) documentation.





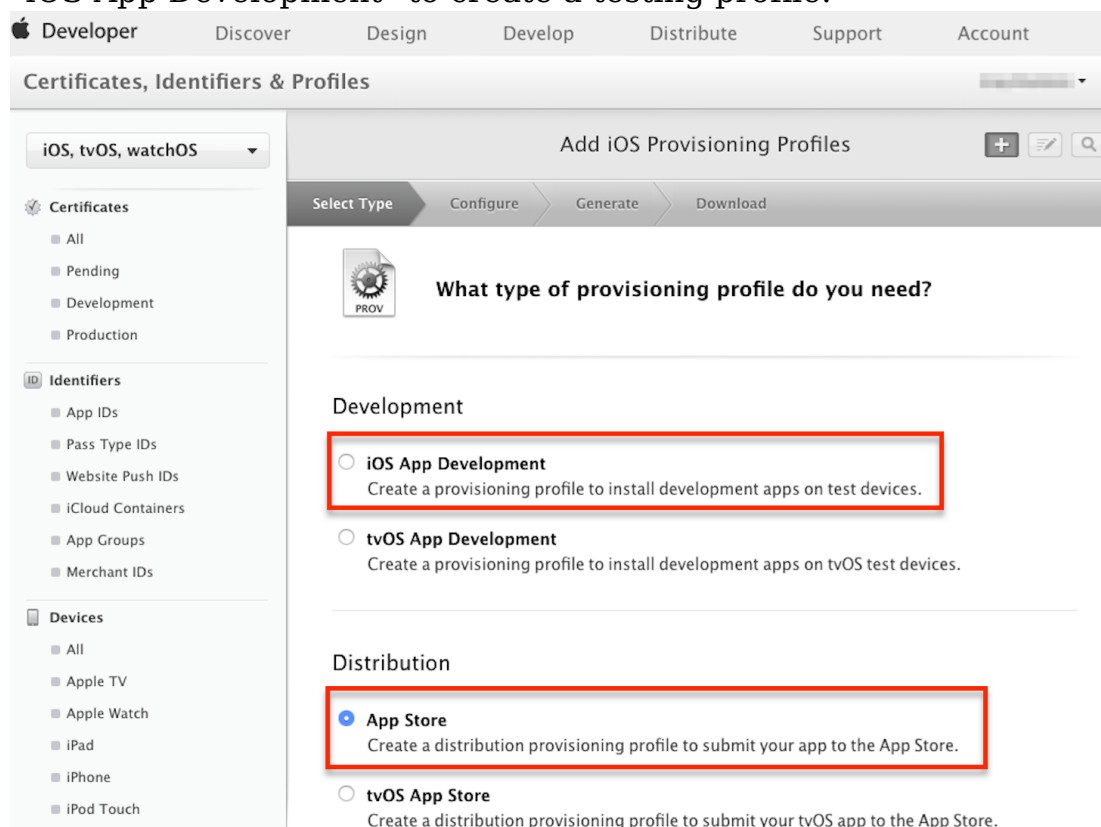
4. At the bottom of the page, click "Continue" to view the confirmation page.
5. Finally, click "Register" to finalize and confirm your app ID.



## Generating a Provisioning Profile

Once you have created the app ID, it is time to create a provisioning profile. This is necessary for installing development applications on iOS devices. To generate a development or distribution provisioning profile:

1. Log into your Apple developer account.
2. Navigate to the Provisioning Profiles section and click the Plus icon to create a new provisioning profile.
3. From this page, select a provisioning profile type:
  - Development profiles do not allow you to publish to stores but are useful if you wish to install and test the app on your device. Select "iOS App Development" to create a testing profile.



- A distribution profile is required if you want to publish your app to stores. Depending on your developer account type, there are two different options for distribution profiles:

- **App Store** : This option is visible for users with an Individual developer account type. Select "App Store" to create a distribution profile.

#### Distribution

- App Store**  
Create a distribution provisioning profile to submit your app to the App Store.

- tvOS App Store**  
Create a distribution provisioning profile to submit your tvOS app to the App Store.

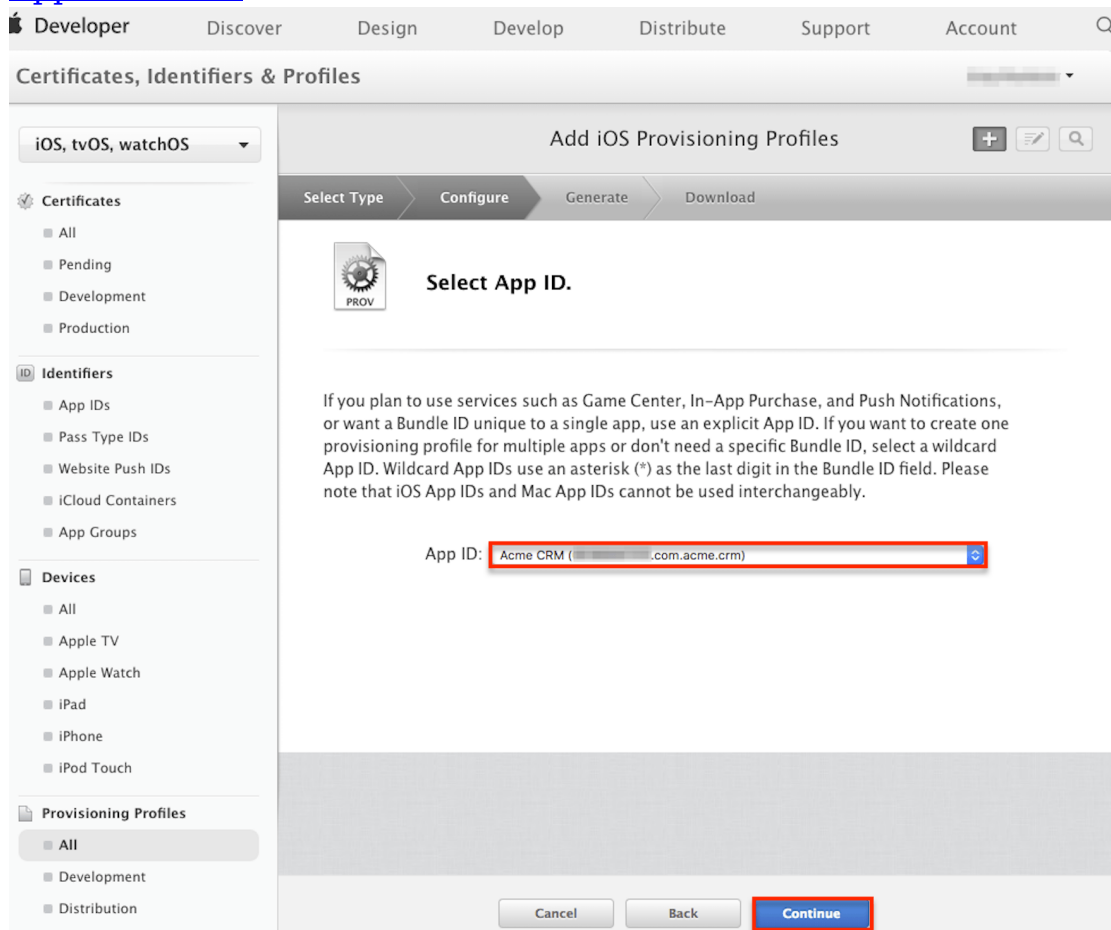
- **In-House** : This option is visible for users with an Enterprise developer account type. Select "In House" to create a distribution profile.

## Distribution

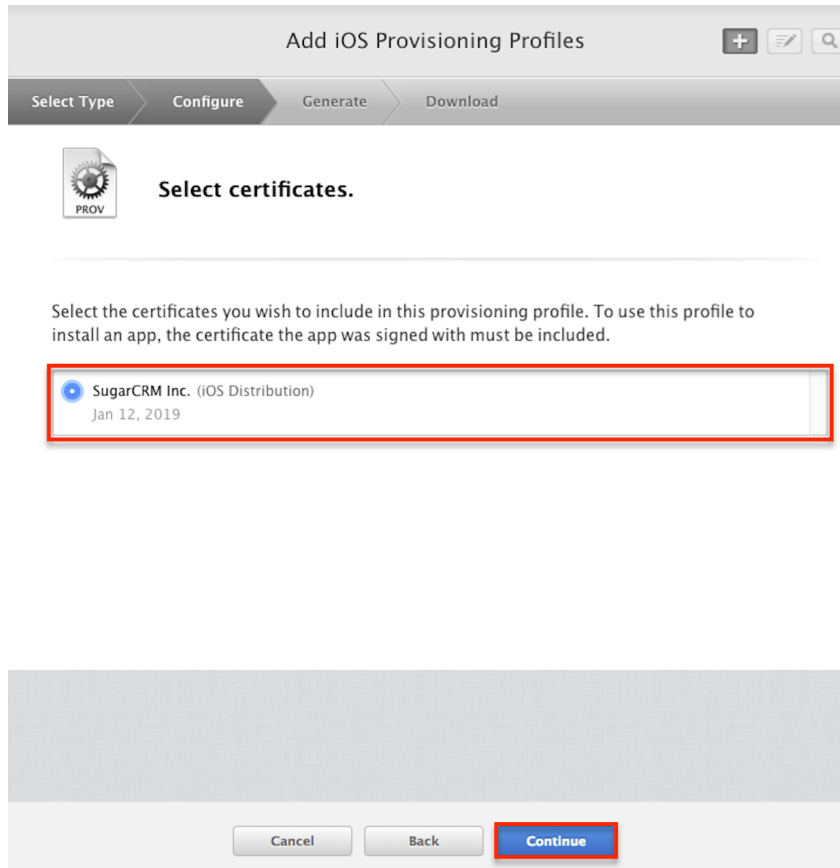
- In House**  
To sign iOS apps for In House Distribution, you need a Certificate.
- Ad Hoc**  
Create a distribution provisioning profile to install your app on a limited number of registered

- **Ad-Hoc** : This option is also visible for users with an Enterprise developer account type. If you wish to only install your app on specific devices, select "Ad-Hoc". For more information on this, please refer to the [About Ad Hoc Provisioning Profiles](#) page in Apple's App Distribution Guide.

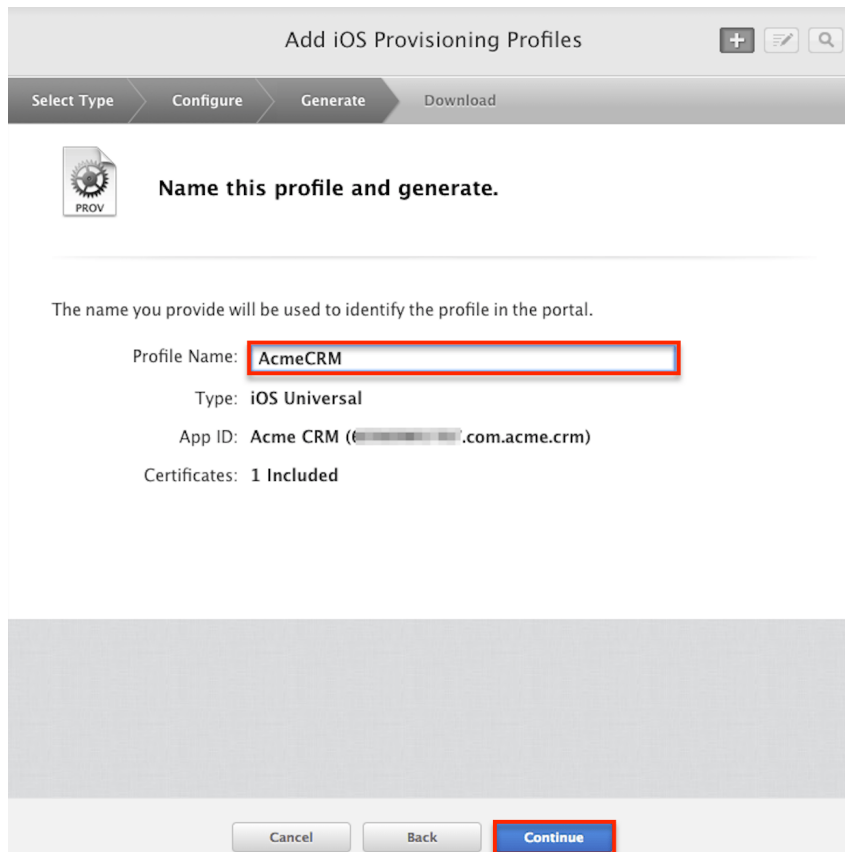
4. After selecting the appropriate profile type, click "Continue".
5. On the Configure page, select the app ID you created in the [Creating an Application ID](#) section then click "Continue".



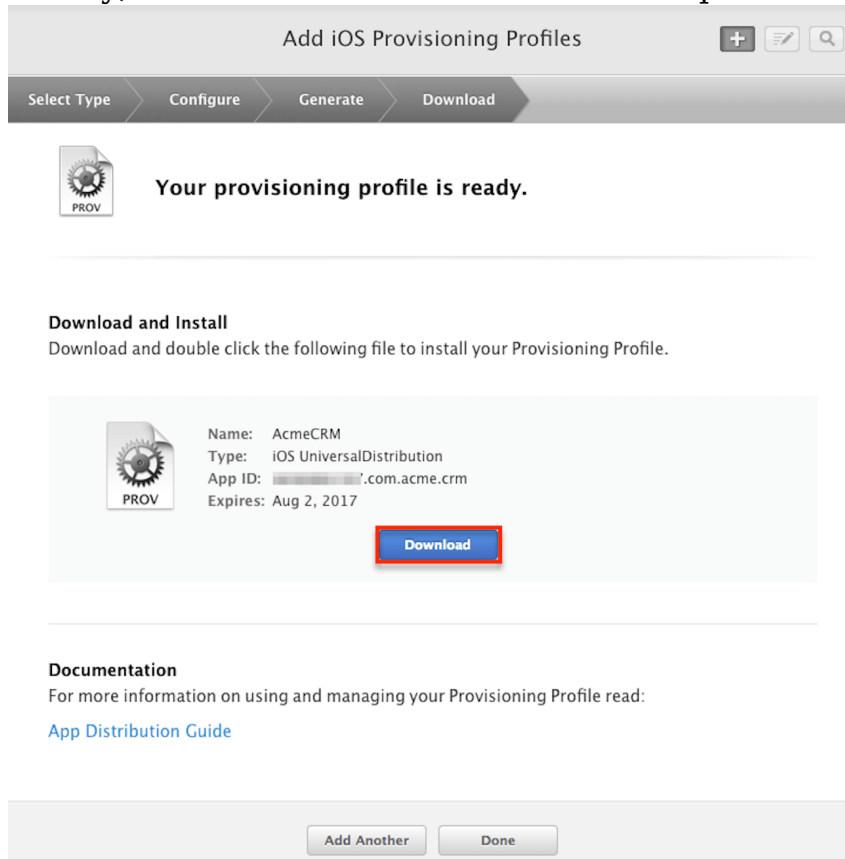
6. On the next page, select the certificate you created in the [Generating a Signing Certificate](#) section then click "Continue".



7. If you selected a development profile or an ad hoc distribution profile in step two, there will be an additional step where you must select the allowed iOS devices. This ensures the signed app can only be installed on the given phones or tablets. Click "Continue" to move to the next page.  
**Note:** You must have already registered these devices in the Devices section prior to generating the profile.
8. Enter the desired profile name in the Profile Name field and click "Continue".



9. Finally, click "Download" to download the profile to your computer.



You now have your certificate and profile files ready to upload to MACS in order to

---

sign the iOS app. For detailed information about signing identities, please refer to the [Maintaining Your Signing Identities and Certificates](#) article in Apple's Distribution Guide.

## App Signing for Android

Before you can distribute the application to the Google Play Store, an internal enterprise store, or an MDM account, you must first app sign for Android. This section will provide information on how to register for an Android Developer account and how to generate an Android signing certificate.

### Registering an Android Developer Account

If you wish to distribute your app via the Google Play store, you must first register an Android Developer Account. To do this, navigate to the [Android Developers Portal](#), and follow the instructions. For more information, please refer to the [Get Started with Publishing](#) documentation on the Android developer site.

### Generating a Signing Certificate

Prior to distributing an Android app to mobile devices via any method, you must generate a signing certificate using the Java keytool command interface utility. Keytool is part of Java development kit (JDK) and is available as part of JDK installation and Android SDK. For more information on acquiring the utility, please refer to Oracle's [Java SE Downloads](#) article or the [Android Developer Guide](#).

Once the keytool utility is available on your computer, you can then easily generate your certificate which will then be stored in your keystore file. To do this, simply run the following command from the command-line:

```
keytool -genkey -v -keystore <keystore-name> -alias <alias> -keyalg <key-algorithm> -keysize <key-size> -validity <days-valid>
```

The following arguments are used by the keytool utility:

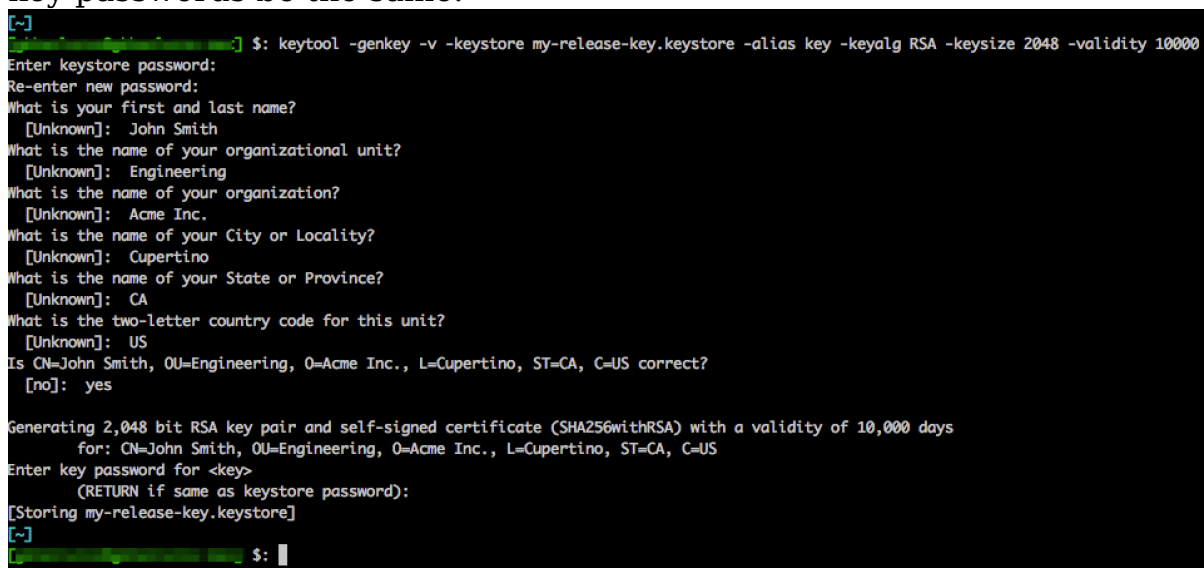
- **<keystore-name>**: File name of the resulting keystore
- **<alias>**: Alias name of the entry to process  
**Note:** The alias name for your keystore must be set to "key".
- **<key-algorithm>**: The key algorithm name  
**Note:** The key algorithm must be set to "RSA".
- **<keysize>** : Key bit size  
**Note:** The key size must be set to 2048.

- 
- **<days-valid>**: Number of days your app is valid

In the following example, the command-line keytool utility will generate the keystore as a file called my-release-key.keystore with a single key that will be valid for 10,000 days:

```
keytool -genkey -v -keystore my-release-key.keystore -alias key -keyalg RSA -keysize 2048 -validity 10000
```

Once you run this command, the keytool will prompt you to specify the keystore password and key password, along with a request for additional information (e.g. First and Last Name). Please note that MACS requires that the keystore and the key passwords be the same.



```
[~]
[~] $: keytool -genkey -v -keystore my-release-key.keystore -alias key -keyalg RSA -keysize 2048 -validity 10000
Enter keystore password:
Re-enter new password:
What is your first and last name?
[Unknown]: John Smith
What is the name of your organizational unit?
[Unknown]: Engineering
What is the name of your organization?
[Unknown]: Acme Inc.
What is the name of your City or Locality?
[Unknown]: Cupertino
What is the name of your State or Province?
[Unknown]: CA
What is the two-letter country code for this unit?
[Unknown]: US
Is CN=John Smith, OU=Engineering, O=Acme Inc., L=Cupertino, ST=CA, C=US correct?
[no]: yes

Generating 2,048 bit RSA key pair and self-signed certificate (SHA256withRSA) with a validity of 10,000 days
for: CN=John Smith, OU=Engineering, O=Acme Inc., L=Cupertino, ST=CA, C=US
Enter key password for <key>
(RETURN if same as keystore password):
[Storing my-release-key.keystore]
[~]
[~] $: |
```

For more instructions on how to sign your app for Android, please refer to the [Sign Your App](#) page on the Android developer site.

**Note:** After submitting a signed app to Google Play, you must use that same signing certificate for all future updates. If you change the certificate, Google Play will not recognize the app as a new version of the same application, and users will not be able to update. If you lose the signing certificate, you will have to register a new app in Google Play.

## Additional Resources

More information about app-signing procedures and requirements is available directly from the iOS and Android developer sites:

- [iOS Developer Center](#)

- 
- [iOS App Distribution Guide](#)
  - [Android Developers Portal](#)
  - [Android App Publishing Guide](#)
  - [Android Keystore System](#)

**Last Modified:** 2023-05-02 14:21:53