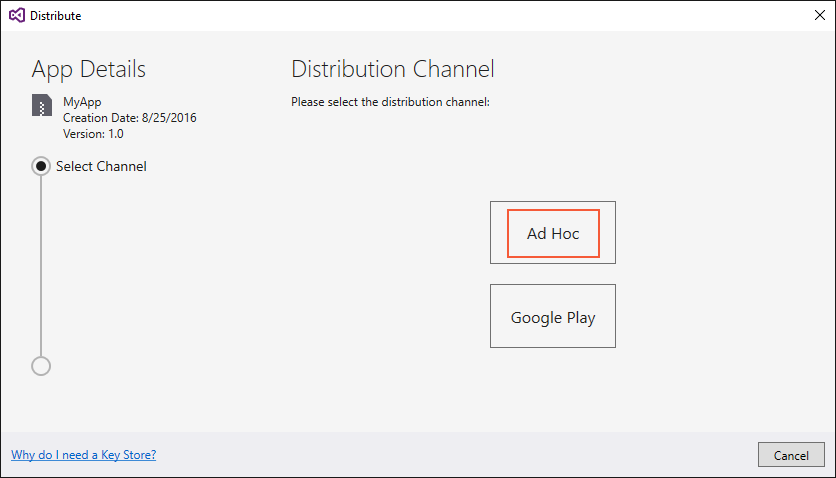
**Signing the Android Application Package**

This guide explains how to sign the Android application package using Xamarin Android 7.0 or later.

# Signing the APK

In this section, we'll use Visual Studio's integrated publishing workflow to sign the APK. In [Part 1](https://developer.xamarin.com/guides/android/deployment,_testing,_and_metrics/publishing_an_application/part_1_-_preparing_an_application_for_release) we used the **Archive Manager** to build the app and place it in an archive for signing and publishing. This section explains how to create an Android signing identity, create a new signing certificate for Android applications, and publish the archived app ad hoc to disk. The resulting APK can be side loaded into Android devices without going through an app store.

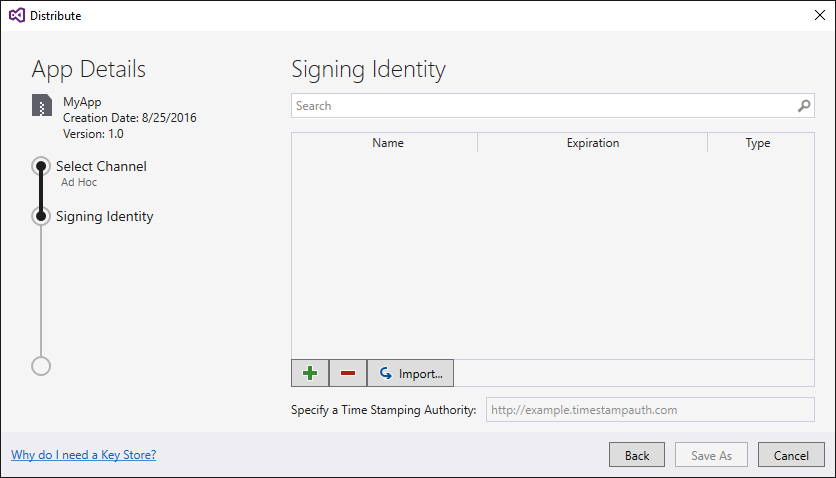
In [Archive for Publishing](https://developer.xamarin.com/guides/android/deployment,_testing,_and_metrics/publishing_an_application/part_1_-_preparing_an_application_for_release#archive), the **Distribution Channel** dialog presented two choices for distribution. Select **Ad-Hoc**



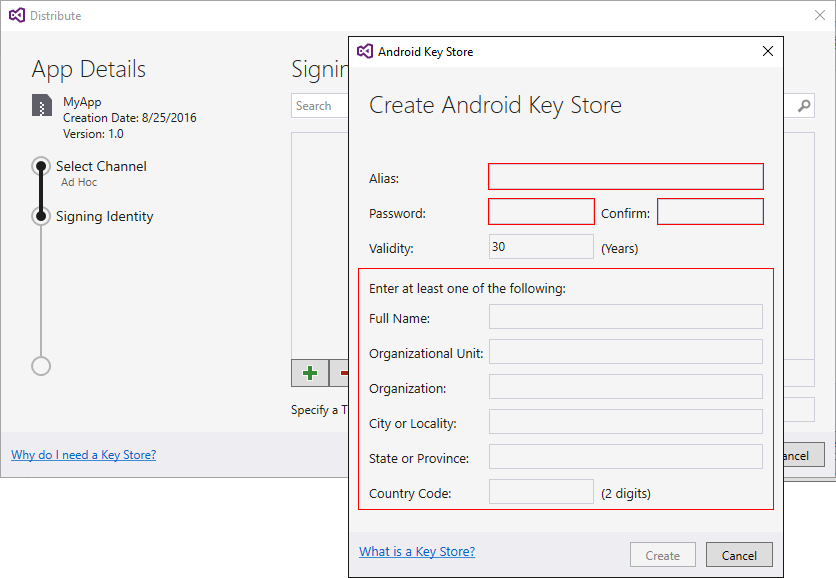
# Create a New Certificate

After **Ad-Hoc** is selected, Visual Studio opens the **Signing Identity** page of the dialog as shown in the next screenshot. To publish your .APK, you must first sign it with a signing key (also referred to as a certificate).

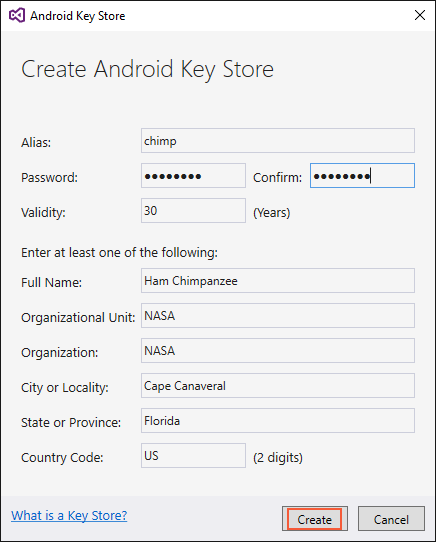
If you already have a certificate, you can click the **Import** button to import an existing certificate and then proceed to [**Sign the APK**](https://developer.xamarin.com/guides/android/deployment,_testing,_and_metrics/publishing_an_application/part_2_-_signing_the_android_application_package/#sign-the-apk-vs). Otherwise, click the click the **+** button to create a new certificate



The **Create Android Key Store** dialog is displayed; use this dialog to create a new signing certificate that you can use for signing Android applications. Enter the required information (outlined in red) as shown in this dialog.

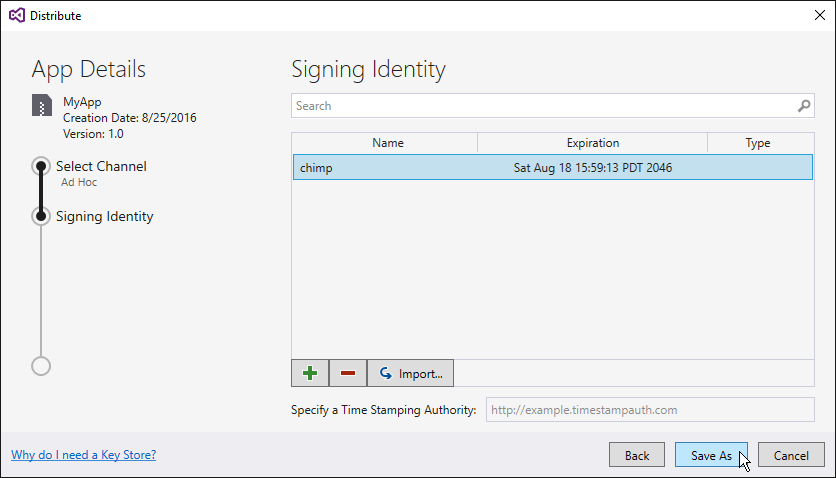


The following example illustrates the kind of information that you must enter. Click **Create** to create the new certificate

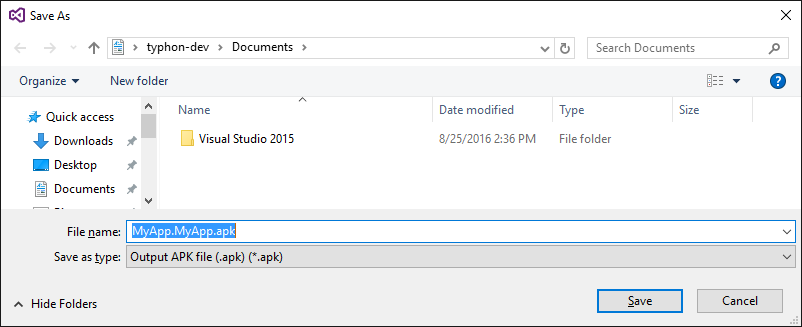


# Sign the APK

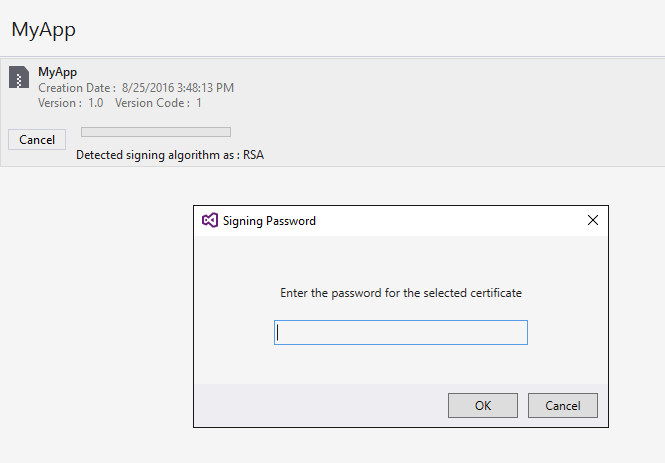
After you click **Create**, a new key store (containing a new certificate) is created and listed under **Signing Identity** as shown in the next screenshot. To publish an app on Google Play, click **Cancel** and go to [Part 3](https://developer.xamarin.com/guides/android/deployment,_testing,_and_metrics/publishing_an_application/part_3_-_publishing_an_application_on_google_play/). To publish *ad-hoc*, select the signing identity to use for signing and click **Save As** to publish the app for independent distribution. For example, the **chimp** signing identity (created earlier) is selected in this screenshot



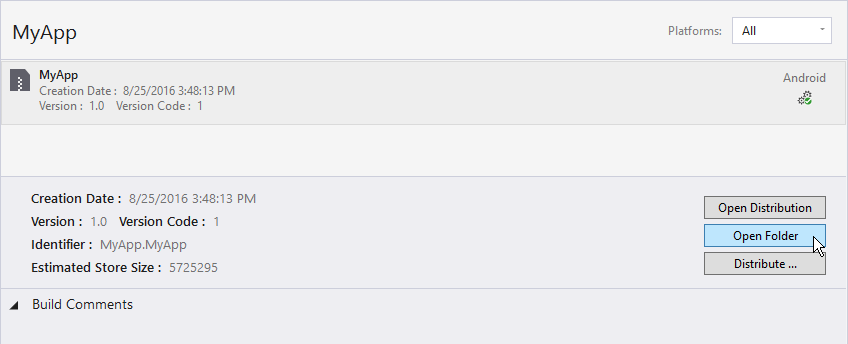
Next, the **Archive Manager** displays the publishing progress. When the publishing process completes, the **Save As** dialog opens to ask you to browse to a location where the generated .APK file is to be stored.



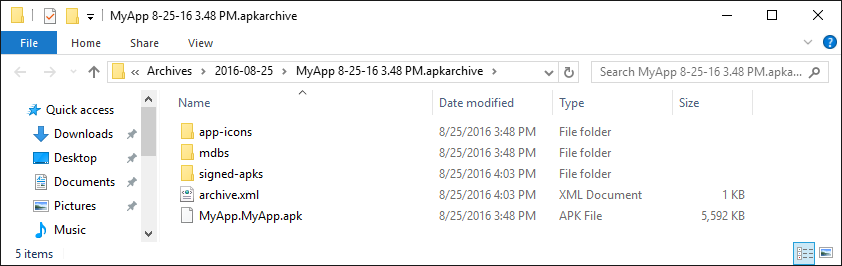
Navigate to the desired location and click **Save**. If the key password is unknown, the **Signing Password** dialog will appear to prompt you to enter the password for the selected certificate



After the signing process completes, click **Open Folder.**



This causes Windows Explorer to open the folder containing the generated APK file. At this point, Visual Studio has compiled the Xamarin.Android application into an APK that is ready for distribution. The following screenshot displays an example of the ready-to-publish app, **MyApp.MyApp.apk**



After Complete above all steps.

# Obtaining your Signing Key Fingerprint for Release

# Locate your release certificate keystore file the release certificate, you also need the certificate's alias and the passwords for the keystore and the certificate. You can list the aliases for all the keys in a keystore by entering:

Keytool -list -keystore **your\_keystore\_name**

Replace your\_keystore\_name with the fully-qualified path and name of the keystore, including the .keystore extension. You'll be prompted for the key store’s password. Then keytool displays all the aliases in the keystore.

Keystore Location:

C:\Users\[UserName] \AppData\Local\Xamarin\Mono for Android\Keystore\AliasBooKClient

Information about a keystore is obtained by running the keytool.exe command from the JDK. This tool is typically found in the following location

C:\Program Files (x86)\Java\jdkVERSION\bin\keytool.exe

Enter the following at a terminal or command prompt:

C:\Program Files (x86)\Java\jdk1.7.0\_71\bin>keytool -list -v -keystore “C:\Users\ [UserName] \AppData\Local\Xamarin\Mono for Android\Keystore\AliasBooKClient.keystore” -alias AliasDigiAgri

You should see something like the following output in your console window.



You will use the SHA-1 fingerprint (listed after **SHA1**) later in this guide.

# Creating an API project

After you have retrieved the SHA-1 fingerprint of the signing keystore, it is necessary to create a new project in the Google APIs console (or add the Google Maps Android API service to an existing project).

-

-

-

And follow **Google Map API Key.docx** document.