

# CREATING NATIVE MOBILE APPS USING POWERAPPS



Michael Richard B

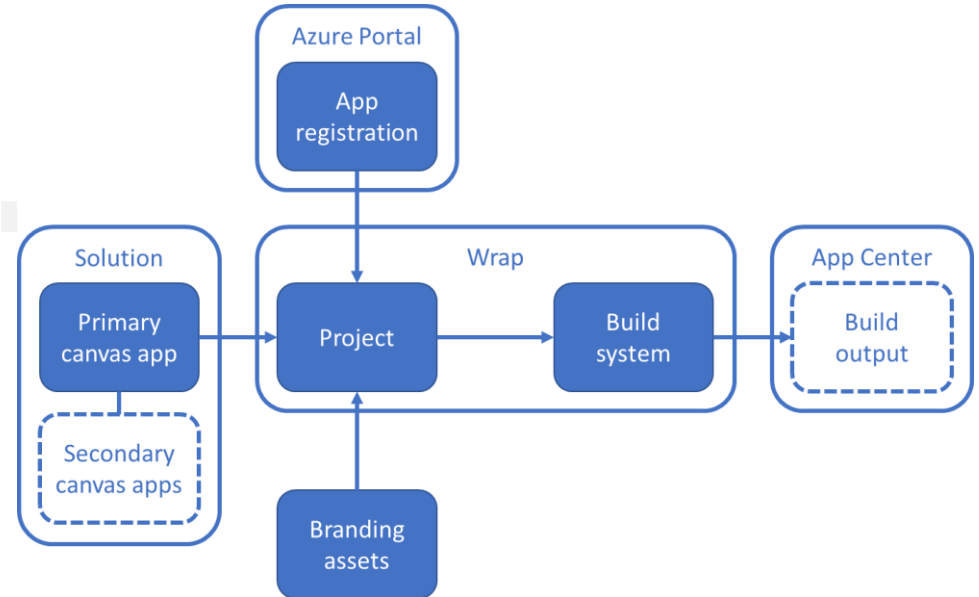
Architect

#FTRSA

*mgrb.in*

# What is a wrap?

Wrap your canvas apps as custom-branded Android and iOS apps for native distribution to mobile users through Microsoft Intune, Microsoft App Center, Google Play Store and Apple Business Manager.



# What we are going to do?

- Create a simple canvas app which will be wrapped as an Android APK file using the below
  - Android Studio
    - To generate the certificates / keys
    - To generate the signature hash
    - Signing the APK package
  - Azure Key Vault (Optional)
    - To store the certificates
  - App registration
    - Not Google Play signing
  - Install the app in a Android Device

# Pre-requisites



PowerApps Access

Environment – Dynamics 365 Apps installation access (Sys Admin)  
Canvas App (Owner)



Azure Key Vault Access (Create / Modify) (Optional – Only for Autosigning)



Azure App Registration Access (Create / Modify)



Android Studio



Open SSL =

[GitHub - openssl/openssl: TLS/SSL and crypto library](#)  
[How To Install OpenSSL on Windows – TecAdmin](#)



Hexadecimal to Base64 Converter (Optional)

[Hex to Base64 converter to convert Hexadecimal Encoded data to Base64 String. \(codebeautify.org\)](#)



Visual Studio App Center



Image creator or resizer tools

# High Level Steps





# DETAILED STEPS



# Below steps are considered known and completed..



Install Wrap Solution from AppSource



Create a canvas app



Create a PowerApp Solution



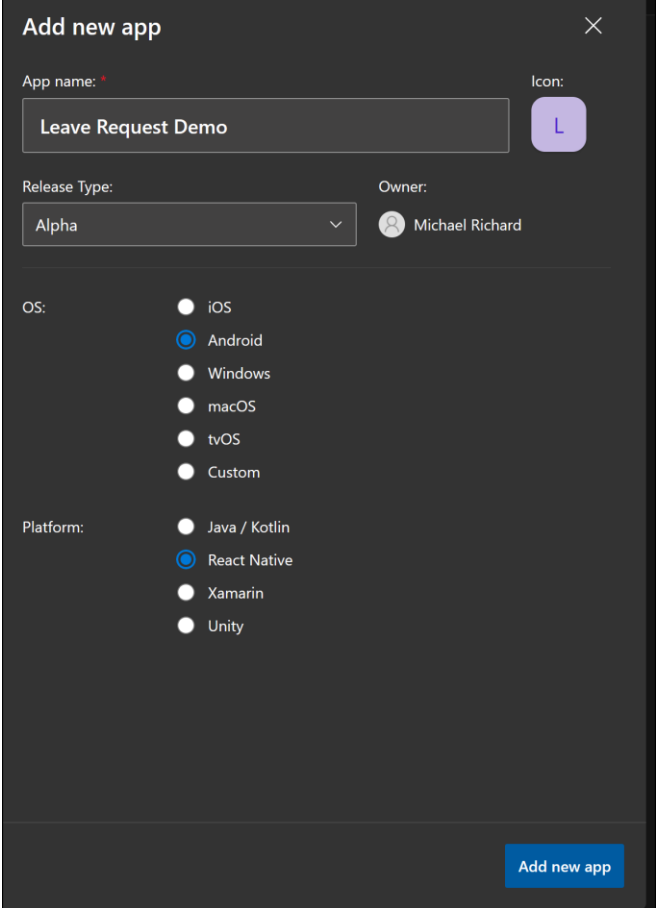
Add the canvas app to the PowerApp solution



Install Android Studio

# 4. VS App center registration

- Login to app center
- Click Add New Organization
- Once Org is added, Click Add New App to Org
- Enter the App Name, Release Type
- Enter OS as Android Platform as Reactive Native



The screenshot shows the 'Add new app' dialog in Visual Studio App Center. The dialog is dark-themed and contains the following fields and options:

- App name:** Leave Request Demo
- Icon:** A purple square with the letter 'L'.
- Release Type:** Alpha (dropdown menu)
- Owner:** Michael Richard (with a user profile icon)
- OS:** Radio buttons for iOS, Android (selected), Windows, macOS, tvOS, and Custom.
- Platform:** Radio buttons for Java / Kotlin, React Native (selected), Xamarin, and Unity.
- Button:** Add new app (blue button at the bottom right).



# 5.1 Generating Certificate

- Install Android Studio
- Open CMD as Admin (Note: JRE folder doesn't have KeyTool)
  - Go to C:\Program Files\Android\Android Studio\jbr\bin
  - Type the below command to generate the PFX certificate

```
C:\Program Files\Android\Android Studio\jbr\bin>keytool -genkey -alias AndroidApps -keyalg RSA -keystore MikeAndroidApps.pfx -keysize 2048 -validity 10000
Enter keystore password:
Re-enter new password:
What is your first and last name?
[Unknown]:
What is the name of your organizational unit?
[Unknown]: AndroidApps
What is the name of your organization?
[Unknown]: AndroidApps
What is the name of your City or Locality?
[Unknown]: Singapore
What is the name of your State or Province?
[Unknown]: Singapore
What is the two-letter country code for this unit?
[Unknown]: SG
Is CN=, OU= AndroidApps, O= AndroidApps, L=Singapore, ST=Singapore, C=SG correct?
[no]: Yes
```

# 5.2 Generating Signature Hash

- Run the export cert command to generate the SHA1 Key for android

```
C:\Program Files\Android\Android Studio\jbr\bin>keytool -exportcert -alias AndroidApps -keystore AndroidApps.pfx  
| "C:\Program Files\OpenSSL-Win64\bin\openssl.exe" sha1 -binary | "C:\Program Files\OpenSSL-Win64\bin\openssl.exe" base64  
Enter keystore password:  
dlWp      :UmYux1vV0kuZrKA0=
```

- Copy the generated Signature Hash to use in the 'New App Registration' section of the Wrap Wizard

# 6. Adding Certificate to Key Vault and Adding Tags

The screenshot shows the 'Certificates' page in the Azure Key Vault interface for the vault named 'AndroidAppDemo'. The left-hand navigation pane includes options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Access policies, and Events. The main content area features a search bar and several action buttons: Generate/Import, Refresh, Restore Backup, Manage deleted certificates, and Certificate Contacts. Below these is a table with columns for Name, Thumbprint, Status, and Expiration date. The table lists one certificate with the name 'sgppugonboardingapp', a thumbprint starting with 'DB63975546ABA3349AC7E2F...', a status of 'Enabled', and an expiration date of '8/9/2050'. There are also sections for 'Completed', 'In progress, failed or cancelled', and a message stating 'There are no certificates available.'

Name	Thumbprint	Status	Expiration date
<b>Completed</b>			
sgppugonboardingapp	DB63975546ABA3349AC7E2F...	✓ Enabled	8/9/2050
<b>In progress, failed or cancelled</b>			
There are no certificates available.			

The screenshot shows the 'Tags' page in the Azure Key Vault interface for the vault named 'AndroidAppDemo'. The left-hand navigation pane is similar to the previous screenshot but includes 'Keys' at the bottom. The main content area has a search bar and a 'Delete all' button. Below this is a text block explaining that tags are name/value pairs used for categorizing resources and consolidated billing, and that tag values are case sensitive. A warning states: 'Do not enter names or values that could make your resources less secure or that contain personal/sensitive information because tag data will be replicated globally.' Below the text is a table with columns for Name and Value. Two tags are listed: 'app.sgppug.onboardingapp' with value 'sgppugonboardingapp' and 'com.mikeapps.leaverequestdemo' with value 'MikeAndroidApps'. Each tag entry has delete and copy icons. At the bottom, there are two empty input fields for Name and Value, separated by a colon.

Name	Value
app.sgppug.onboardingapp	sgppugonboardingapp
com.mikeapps.leaverequestdemo	MikeAndroidApps

# Wrap Wizard..

## 1. Select App

**Select Apps Step**  
Leave Request\_0314

**Select the app(s) to wrap**

Primary app \*  
Leave Request\_0314

Secondary app(s)  
Select an option

## 2. Choose Platform

**Target Platforms Step**  
Android (APK), Google Play Store (AAB)

**Choose mobile platforms to target**  
Select all platforms that apply based on the device type(s) of your end users. Enable signing to reduce manual steps later.

Bundle ID \*  
com.mikeapps.leaverequestdemo

Target platform(s) \*  
 Android (APK)  
 Google Play Store (AAB)  
 iOS (IPA)

Sign my app (preview)  
 On

Azure Key Vault URI  
https://androidappdemo.vault.azure.net/

**\* My setup didn't work with Autosigning (Sign my app) enabled**

# Wrap Wizard..

## 3. Configure Branding

Select Apps Step  
Leave Request\_0314

Target Platforms Step  
Android (APK), Google Play Store (AAB)

**Configure Branding Step**

Register app

Manage output

Wrap up

Android app icons

432px

324px

216px

162px

108px

Splash screen image

undefined

Upload image

Welcome screen image

undefined

Upload image

Background fill color

#82c7fc

Button fill color

#82c7fc

Status bar text theme

Back

Next

## 4. Azure App Registration

+ New app registration

### Register your app

We need to register your app in the Azure cloud. This enables users to sign into your app. If you would like to use an existing registration that you have created, pick it below. Otherwise create a new app registration.

#### Owned registrations

com.█.apps.leaverequestdemo

#### Azure API Connections

- ✓ Azure API Connections
- ✓ Dynamics CRM
- ✓ Microsoft Graph
- ✓ PowerApps Service
- ✓ Mobile Device Management
- ✓ Power BI

#### Authentication

To configure your apps authentication flow we need to use your provided Bundle ID.

- ✓ Bundle ID com.█.apps.leaverequestdemo
- ✓ Android Redirect URI msauth://com.█.apps.leaverequestdemo/dIWpJaROF█

# Wrap Wizard..

## 5. App Center Linkage

+ New location ▾

Step  
\_0314

Forms Step  
Google Play Store

Branding Step

Output

### Manage output

Automate app setup using your App Center account. We'll need

#### App Center location

[Get App Center token](#)

Authentication token \*

8a1e [REDACTED]

**Thank you Michael Richard. Your token is valid!**

App Center org \*

[REDACTED]-Org ▾

Android App Center location \*

[REDACTED]-Org/Leave-Request-Demo ▾

## 6. Build

Apps Step  
Request\_0314

Platforms Step  
d (APK), Google Play Store

Configure Branding Step

Configure app


Generate output


Setup

### Wrap up

Review your app's details below. If it all looks good, select Build.

#### Overview

 **Application bundle ID**  
com.[REDACTED].ps.leaverequestdemo

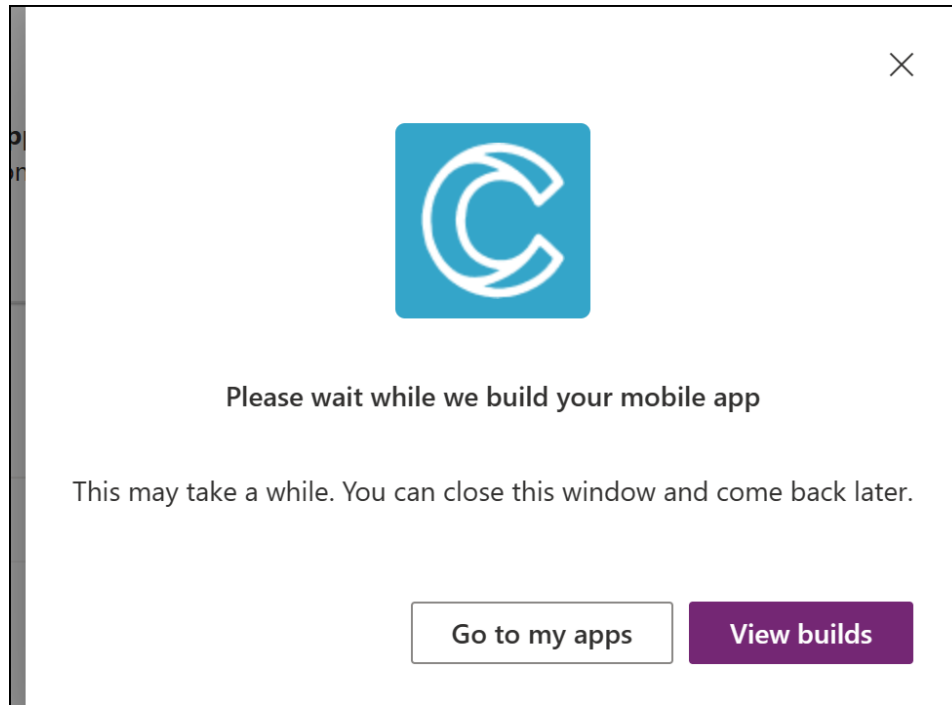
 **Sign my app**  
Enabled

**Contoso Mobile**

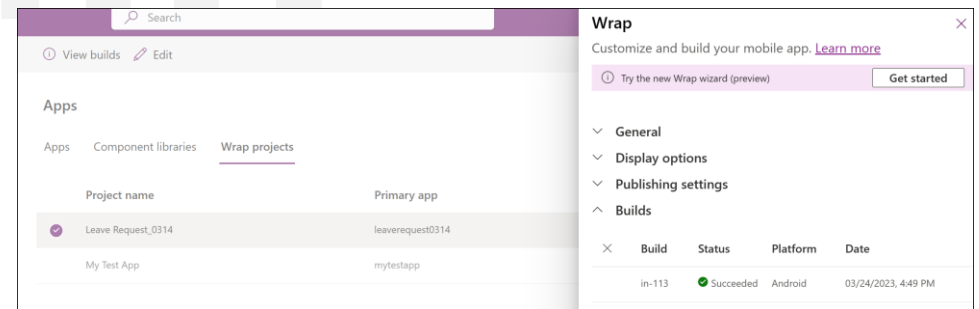
Platform	App Center location
Android	com.[REDACTED].ps.leaverequestdemo

# Wrap Wizard.

## 7. Build starts



## 8. Build Completed Successfully



# 9. App Center – APK Releases




The screenshot displays the App Center interface for an Android application. The breadcrumb navigation at the top reads: Demo Org / Leave Request Demo / Distribute / Releases / Release 2. The left sidebar contains navigation options: Overview, Build, Test, Distribute, Releases (selected), Groups, Stores, CodePush, Diagnostics, Analytics, and Settings. The main content area is divided into two sections. On the left, a table lists releases:

Release ID	Version	Count	Date
2	1.30320.30 (290000)		Mar 24
1	1.30320.30 (290000)		Mar 24

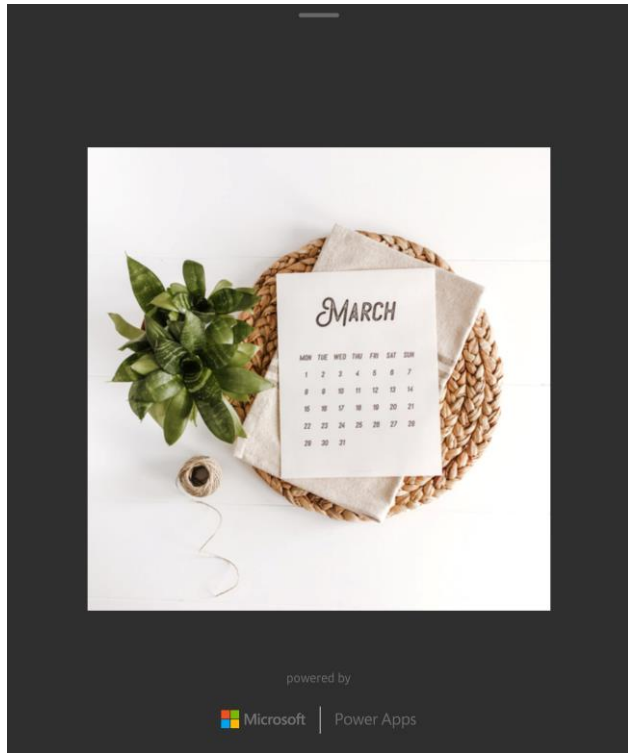
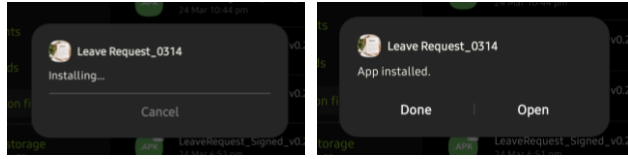
On the right, the details for Release 2 are shown. It includes a box icon, the version number **Version 1.30320.30 (290000)**, the timestamp **Mar 24, 2023, 5:01 PM**, and the download status **0 unique / 0 total**. The release was released by **in-113**. Technical specifications include: Minimum OS **Android 5.1**, Size **73.69 MB**, File extension **apk**, and MD5 fingerprint **30a5d**. Destinations are listed as **Groups: Collaborators**. Action buttons for **Distribute** and **Download** are visible at the top right of the release details.



# 10. Signing the APK Package (Optional)

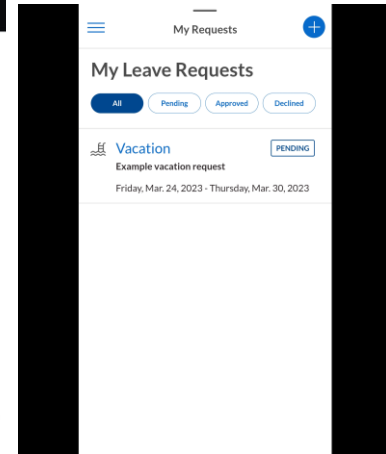
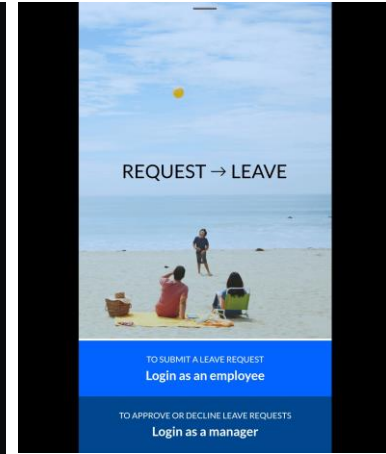
- Open command prompt (in Admin Mode)
- Navigate to Android SDK location (You can get this from Android Studio  Tools  SDK Manager  Android SDK Location)
  - Usually it will be -  
C:\Users\
- Enter the below command to sign the file
  - `apksigner.bat sign --ks PATH_TO_KEYSTORE --ks-key-alias KEY_ALIAS PATH_TO_APK`

# 11. Install APK and Enjoy!



Sign in with your work or school email account

Sign in



# Useful Links

- [Sign your app | Android Developers](#)
  - <https://learn.microsoft.com/en-us/power-apps/maker/common/wrap/code-sign-android#sign-the-apk-package>
  - [Code sign for Android - Power Apps | Microsoft Learn](#)
  - [Sign your app | Android Developers](#)
- [Open SSL:](#)
  - [GitHub - openssl/openssl: TLS/SSL and crypto library](#)
  - [How To Install OpenSSL on Windows – TecAdmin](#)
- [Frequently asked questions for wrap - Power Apps | Microsoft Learn](#)
- [Troubleshoot wrap issues - Power Apps | Microsoft Learn](#)
- [Announcing general availability of wrap for Power Apps | Microsoft Power Apps](#)

+



o



.



# THANK YOU

[mgrb.in](http://mgrb.in)

# Code Signing for Android APK

To generate a key, open a command prompt and run the following command:

```
keytool -genkey -alias SIGNATURE_ALIAS -keyalg RSA -keystore PATH_TO_KEYSTORE -keysize 2048 -validity 10000
```

Parameters:

- **genkey** - command to generate a key.
- **alias** - indicates the alias to be used in the future to refer to the keystore entry containing the keys that will be generated.
- **keyalg** - key algorithm name.
- **keystore** - the name of the keystore you're using.
- **keysize** - the size of each key to be generated.
- **validity** - validity of the key in number of days.

Example:

- If preparing Keyvault, `PATH_TO_KEYSTORE` should have `.pfx` extension.

```
keytool -genkey -alias powerappswrap -keyalg RSA -keystore powerappswrap.pfx -keysize 2048 -validity 10000
```

- If preparing for manual signing, `PATH_TO_KEYSTORE` should have `.jks` extension.

```
keytool -genkey -alias powerappswrap -keyalg RSA -keystore powerappswrap.jks -keysize 2048 -validity 10000
```

## Generate signature hash

### Note

Skip to **sign the APK package** if you've already generated keys and signature hash while creating the app registration.

After generating the key, we'll use the `exportcert` command in `keytool` to export the keystore certificate.

```
keytool -exportcert -alias SIGNATURE_ALIAS -keystore PATH_TO_KEYSTORE | openssl sha1 -binary | openssl base64
```

Parameters:

- **exportcert** - reads from the keystore the certificate associated with alias and stores it in the `cert_file` file. When no file is specified, the certificate is output to stdout.
- **alias** - the alias used while generating keys [earlier](#).
- **keystore** - the name of the keystore you're using.
- **openssl** - generates SHA1 key for Android.

Add the generated signature hash in the **Redirect URI** while [registering the app](#).

# App registration

- Create new app registration using 'Accounts in any organizational directory and Personal Microsoft Accounts'
- Add the below redirect URI for android platform
  - Package name:  
com.contoso.myapp
  - Redirect URI:  
msauth://com.contoso.myapp/  
<generated signature hash>

Home > App registrations > Register an application

**Name**  
The user-facing display name for this application (this can be changed later).

Supported account types  
Who can use this application or access this API?

- Accounts in this organizational directory only (Test\_Test\_MSProjectSienaV1 only - Single tenant)
- Accounts in any organizational directory (Any Azure AD directory - Multitenant)
- Accounts in any organizational directory (Any Azure AD directory - Multitenant) and personal Microsoft accounts (e.g. Skype, Xbox)
- Personal Microsoft accounts only

[Help me choose...](#)

Android

Redirect URIs

The URIs we will accept as destinations when returning authentication responses (tokens) after successfully authenticating users. The redirect URI you send in the request to the login server should match one listed here. Also referred to as reply URLs. [Learn more about Redirect URIs and their restrictions](#)

Package name	Signature hash	Redirect URI
com.█apps.leaverequestdemo	dIWpJaROFnC█V0kuZrK...	msauth://com.█apps.leaverequestdemo/dIWp█ux... <a href="#">View</a>

[Add URI](#)

# App registration...

- If the app registration throws the error as given in the screenshot, then run the below command in CMD to get the SHA1 Certificate fingerprints and then manually convert from Hexadecimal to Base64 and use that in Signature Hash

< All platforms Quickstart Docs

Configuring your Android app enables your users to get device-wide SSO through the Microsoft Authenticator and seamlessly access your application.

You will be able to change this later.

\* Package name  
Your app's Package Name can be found in the Android Manifest.  
com. [redacted] apps.leaverequestdemo ✓

\* Signature hash  
The Signature Hash can be generated via command line.  
msauth://com. [redacted] apps.leaverequestdemo/[redacted]FnCUmYuxlvV0kuZrKA0=  
✗ The signature hash must be base64-encoded SHA1.

Generating a development Signature Hash. This will change for each development environment.

Generating a production Signature Hash.

```
C:\Program Files\Android\Android Studio\jbr\bin>keytool -list -v -alias androidApps -keystore droidApps.pfx
Enter keystore password:
Alias name: androidApps
Creation date: 24 Mar 2023
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner:
Issuer:
Serial number: 14ab9
Valid from: Fri Mar 24 11:07:30 SGT 2023 until: Tue Aug 09 11:07:30 SGT 2050
Certificate fingerprints:
  SHA1: 76:55:A9:25:A4:
  SHA256: 9A:97:61:38:6F:2D:A6:10:6B:13:D: :9D:2B:1B:0C
Signature algorithm name: SHA256withRSA
Subject Public Key Algorithm: 2048-bit RSA key
Version: 3

Extensions:
#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 05 4  AE 94 64
0010: 54 76
```

# App Permissions on PowerApps

Install-Module -Name  
Microsoft.PowerApps.Administration.PowerShell Install-  
Module -Name Microsoft.PowerApps.PowerShell -  
AllowClobber

```
PS C:\WINDOWS\system32> Add-AdminAllowedThirdPartyApps -ApplicationId f55752c6-7d36-40d0-af50-6b834744b243
StatusCode      : 200
StatusDescription : OK
Content         : {}
RawContent      : HTTP/1.1 200 OK
                  Vary: Authorization,Accept-Language,Accept-Encoding
                  Strict-Transport-Security: max-age=15724800; includeSubDomains
                  x-ms-islandgateway: ga0000003
                  x-ms-request-id: asia.southeastasia...
Headers         : {[Vary, Authorization,Accept-Language,Accept-Encoding], [Strict-Transport-Security,
                  max-age=15724800; includeSubDomains], [x-ms-islandgateway, ga0000003], [x-ms-request-id,
                  asia.southeastasia:00000000-0000-0000-0000-000000000000]...}
RawContentLength : 0
```

- [PowerShell support - Power Platform | Microsoft Learn](https://learn.microsoft.com/en-us/power-apps/maker/common/wrap/how-to#allow-registered-apps-in-your-environment)
- <https://learn.microsoft.com/en-us/power-apps/maker/common/wrap/how-to#allow-registered-apps-in-your-environment>



# Issues encountered

- msdyn\_appiconxxxhdpi is required for selected platform type.  
RequestId 1abadfcc-4039-4c41-a344-42bb0c46b290
  - Solution: Images are mandatory
- Build failures
  - Check whether the app registration is correct with all required API permissions
  - Check whether the Redirect URI is correct with required Bundle ID / Package code and Signature Hash
  - Check whether the Automated Signing is enabled (This didn't seem to work for me)
  - Check whether the App Icons are of correct size as mentioned in the wizard
- Usual PowerShell Execution Policy issue
  - Solution: [Set-ExecutionPolicy \(Microsoft.PowerShell.Security\) - PowerShell | Microsoft Learn](#)

# Issues encountered...

- Please set the JAVA\_HOME variable in your environment to match the location of your Java installation
  - Solution: Run command - set JAVA\_HOME="C:\Program Files\Android\Android Studio\jbr"
- Unable to sign in – Something went wrong [2400]
  - Check and re-setup the API Permissions - <https://learn.microsoft.com/en-us/power-apps/maker/common/wrap/how-to#configure-api-permissions>

# If any error occurs during redirect uri setup

## Convert SHA1 hex to Base64-encoded signature hash manually

You might see the following error if your signature hash is not correctly encoded or unacceptable in the Azure portal:

"The signature hash must be base64-encoded SHA1."

When this error appears, try to generate the signature hash using the following steps instead:

1. Run `keytool -list -v -alias SIGNATURE_ALIAS -keystore PATH_TO_KEYSTORE` to list the certificate information in verbose mode.
2. Copy the **SHA1** value under the **Certificate fingerprints** section from the output. Ensure that you only copy the hexadecimal value.  
For example: `EF:11:45:3D:F1:72:D9:8C:43:32:CD:0A:49:C2:E4:75:2D:B3:2D:9F`
3. Use any available "Hexadecimal to Base64" converter to convert the copied certificate fingerprint hexadecimal value into Base64 encoded value.  
Example of the Base64 encoded value: `8CPPeLaz9etdqYaQubcqsy2Tw=`
4. Copy the generated Base64 encoded value as the **Signature hash** in the Azure portal while [registering the app](#).

# Note:

## Can I create B2C mobile apps with Power Apps?

No. Power Apps is a platform for creating business applications and uses Azure Active Directory authentication. The wrap feature wraps existing canvas apps for the same set of end users.