Version 0.7

Confidential

V-OS Cloud License Generation

Prerequisites

V-KEY

STRONGER WITH V-OS

Table of Contents

Metadata

Property	Content
Document Title	V-OS Cloud License Generation Prerequisites
Document Version	0.7
Last Updated on	2020-12-17
Generated on	2022-05-26, 14:02

Revision History

Ver.	Date	Description/Changes
0.1	2018-08-09	Initial release
0.2	2018-09-20	Upgrading version
0.3	2018-10-24	Content restructuring and editing
0.4	2019-05-13	Update PROD environment URLs and ports
0.5	2019-10-24	Update URLs and mTLS requirements
0.6	2020-01-09	Added Roles and Accounts section
0.7	2020-12-17	Removed ports in URLs of PROD and STG environments

Confidential

This document contains detailed information relating to V-Key's various Products / Services, for which all copyright, trademark(s), patent(s) and/or trade secrets belong to V-Key Inc / Pte Ltd. TAKE NOTICE that this should not be circulated to competitors or disclosed to third parties (other than directors, officers, employees, and agents of the Customer).

> ***Note:*** *Due to low popularity of CDMA mobile devices and mobile network operators are phasing out CDMA network, V-Key does not test any V-Key software product on mobile devices that run on CDMA network. The compatibility of the V-Key software products on CDMA mobile devices is unknown.*

1 Introduction

V-Key issues license to customers who subscribe to V-OS Cloud for deployment of V-OS App Protection, V-OS PKI Token, and V-OS Messaging solutions. To obtain a V-Key license, you must meet certain prerequisites. This document provides the details of the prerequisites that you need to meet before the license can be generated. It inherits the standard V-Key License Generation Prerequisites with the addition of V-OS Cloud specific requirements.

> *Due to the complexity in the process such as multiple physical security clearances at the data center, custodians holding split passwords, the management approval process with maker/checker, etc., V-Key combines requests from various clients and do assets generation once every week. Therefore, it would take approximately 10 working days for the license files to be ready. If you need to obtain the license for your release, you need to send in the request in advance to prevent delay.*

2 Prerequisites

V-OS Cloud license requires the followings as prerequisites:

- For Android only:
  - Package Name
  - Google Service JSON
- For iOS only:
  - App Bundle ID
- V-OS APS Manageability
- Crypto Mode
- SSL Certificate
- URLs
- Assets and Tokens Manageability Information
- App Signer Certificates
- Push Notification Certificates
- mTLS Information

> ***Note:*** *The target OS versions that V-OS Cloud supports are as follows:*
>
> - *Android: 5.0 and above*
> - *iOS: 8.0 and above*

3 Information Preparations

3.1 Package Name, Google Service JSON, and App Bundle ID

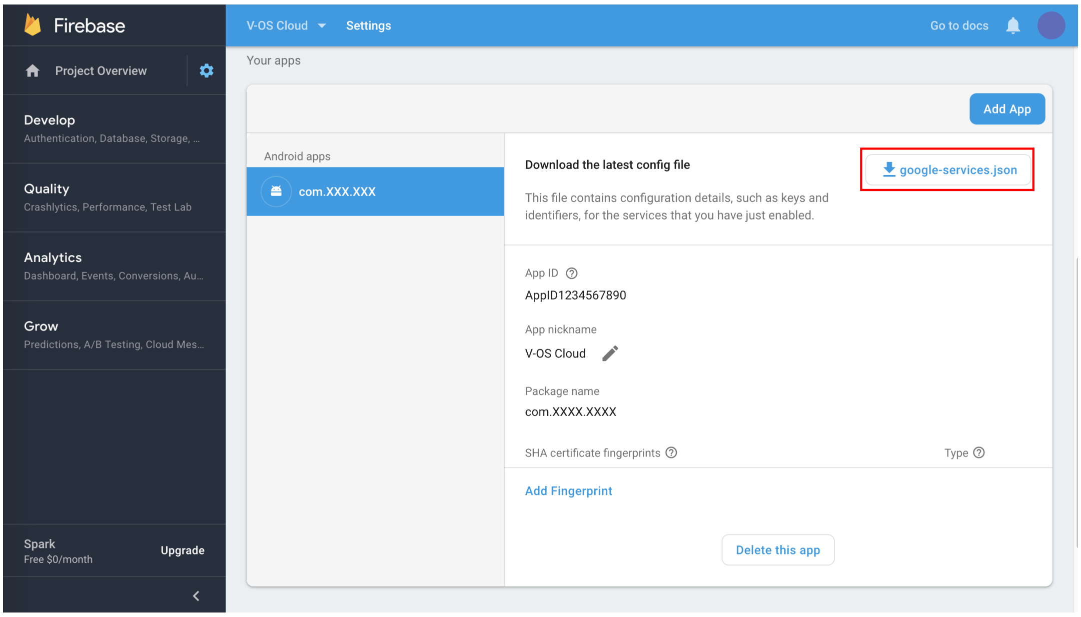
You need to provide the following information if you integrate V-OS App Protection on V-OS Cloud:

- For Android:
  - Package Name
  - Google Service JSON (`google-service.json`)
- For iOS:
  - App Bundle ID

3.1.1 Getting the Google Service JSON

If you have already using Firebase for your app project, do the following steps to get the `google-service.json`.

1. Login to Firebase Console with your Google account.
2. On your project, go to the Settings page of the target app.
3. Select the ⤓ **google-services.json** button to download the `google-services.json` file.



Download the latest config file

This file contains configuration details, such as keys and identifiers, for the services that you have just enabled.

App ID
AppID1234567890

App nickname
V-OS Cloud

Package name
com.XXXX.XXXX

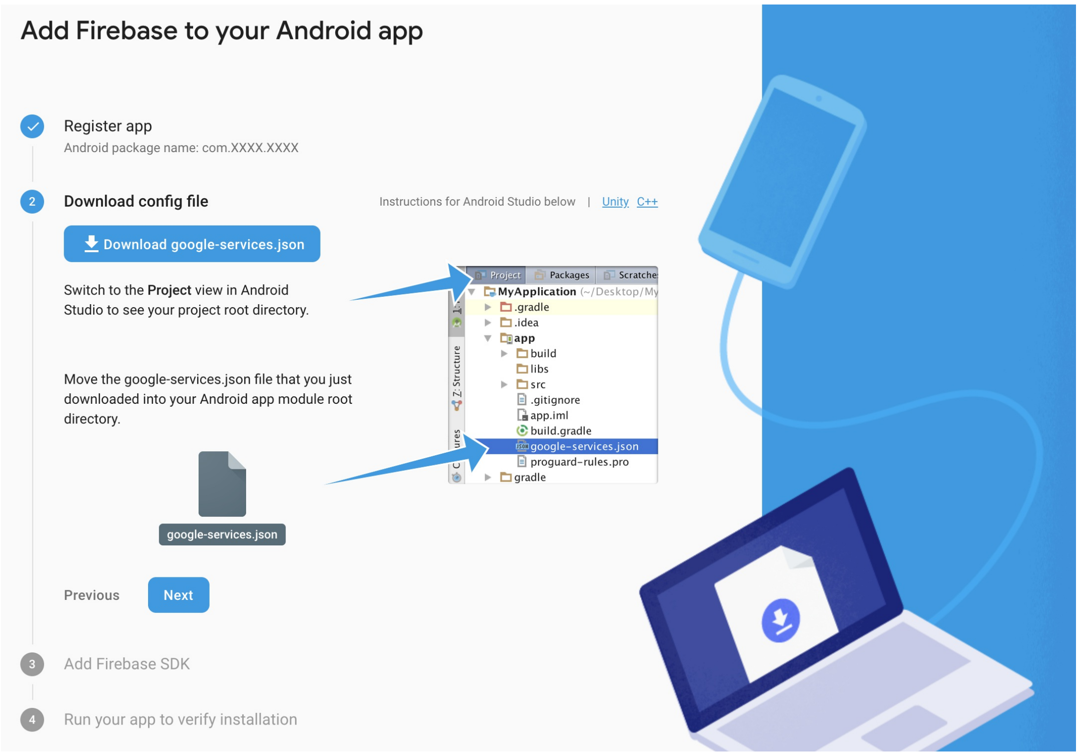
SHA certificate fingerprints
Type

Fig 1: Getting Google Service JSON File 1

If you have not used Firebase for your app project, do the following steps to get the `google-service.json`.

1. Login to Firebase Console with your Google account.
2. On your project, select **+ Add app** to create a new Android app.
3. Fill in the package name and other optional information if necessary.
4. Select **Register app** to register the App.

5. Select the ⤓ **Download google-services.json** button to download the `google-services.json` file.



Add Firebase to your Android app

✓ Register app

Android package name: com.XXXX.XXXX

2 Download config file

Instructions for Android Studio below | Unity C++

⤓ Download google-services.json

Switch to the **Project** view in Android Studio to see your project root directory.

Move the google-services.json file that you just downloaded into your Android app module root directory.

Previous Next

3 Add Firebase SDK

4 Run your app to verify installation

Fig 2: Getting Google Service JSON File 2

6. Follow the instruction to integrate the `google-services.json` to your app project and the rest of the steps to finish the setup.

3.2 V-OS APS Manageability

Is V-OS App Protection Server (APS) used to manage the app?

- Yes: Manageability = 1
- No: Manageability = 0

3.2.1 Roles & Accounts

If manageability is `1`, you can access V-OS App Protection Server UI of V-OS Cloud for V-OS App Protection reports. The roles of V-OS App Protection Server UI are Admin, Ops, Dev, and Report.

- **Admin:** Administrator with access to all features.
- **Ops:** takes care of operational matters, such as updates to profiles, signatures, and checking of logs with access to:

Tab	Page
Home	· Analytics
	· Activity Logs
App Protection	· Troubleshooting
	· Profiles
	· Signatures
	· Exception List
	· Cuckoo Filter Signatures
V-OS	· Firmwares
Smart Token	· All

- **Dev:** takes care of software development, with access to:

Tab	Page
Home	· Analytics
App Protection	· All
V-OS	· Firmwares
Smart Token	· All

- **Report:** takes care of reporting with access to:

Tab	Page
Home	· Analytics
	· Troubleshooting

Tab	Page Activations
App Protection	· Threats
	· Devices
	· Applications
Smart Token	· Reports
	· Device Lock

3.3 Crypto Mode

V-OS Cloud supports Strong Crypto mode only.

3.4 SSL Certification

Because V-OS Cloud uses HTTPS (https://cloud.v-key.com) with a registered SSL certificate, your app needs to integrate the profile (with the V-Key's SSL certificate included) that generated from V-OS Cloud. You do not need to have your own SSL certification.

3.5 URLs

You need to provide the following URLs for integrating V-OS PKI Token to/from V-OS Cloud:

- Activation Status URL
- Authentication Respond URL (Callback URL)

3.6 Assets and Tokens Manageability Information

You need to provide the following information for managing and registering assets/tokens:

- Customer Identification:
  - Customer Name: the official name of your company
  - Customer ID: the ID assigned to your company
- Management Email Address: the email address that is used to register in V-OS App Protection server
- Subject Strings for Attributes in Certificates:
  - CN: Common Name
  - O: Organization
  - OU: Organization Unit
- Tenant Validity:
  - Root CA: XX Years
  - Other CAs: XX Years

3.7 Signer Certificates

You use signer certificate to sign the Android APK and iOS IPA files. The signer certificate is necessary for V-OS protection. It is recommended to include all the developers' certificates.

3.7.1 Android Signer Certificate

To export your signer certificate for Android, do the following steps:

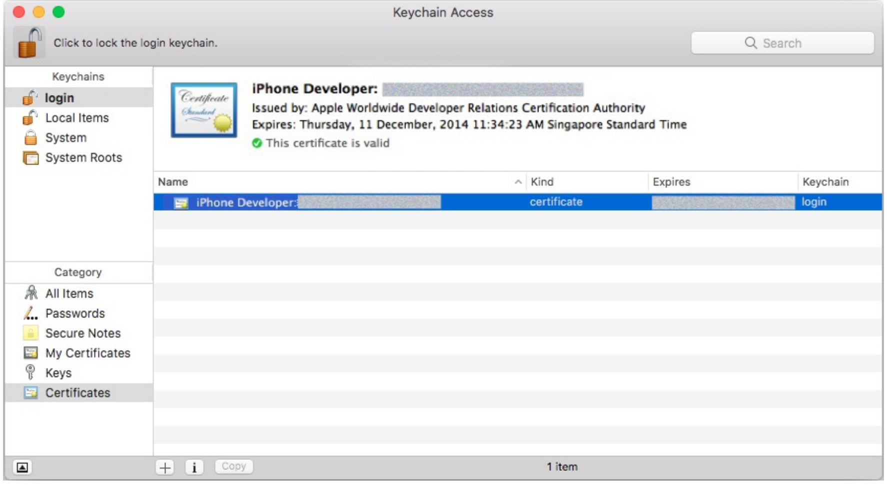
1. On your computer, look for the **Keystore** file for APK signing. The file location may vary for different setups. Below are the common locations where you can find your **Keystore** file depending on your operating system:
   - on Mac OS X and Linux environment: `~/android/` folder
   - on Windows XP: `C:\Documents and Settings\<user>\.android\` folder
   - on Windows Vista and Windows 7, 8, and 10: `C:\Users\<user>\.android\` folder
2. Go to **Terminal** and obtain the certificate information with the command line as follows:

```
    keytool -export -keystore <keystore> -alias <alias> -file
<output.cer>
```

3.7.2 iOS Signer Certificate

To export your signer certificate for iOS, do the following steps:

1. On your Mac, open the `Keychain Access.app` from the `/Applications/Utilities/` folder.
2. Look for the specific file that is used to sign the iOS IPA file. There are various certificates for different setups. In the example that follows, the **iPhone Developer: XXX** certificate is used.

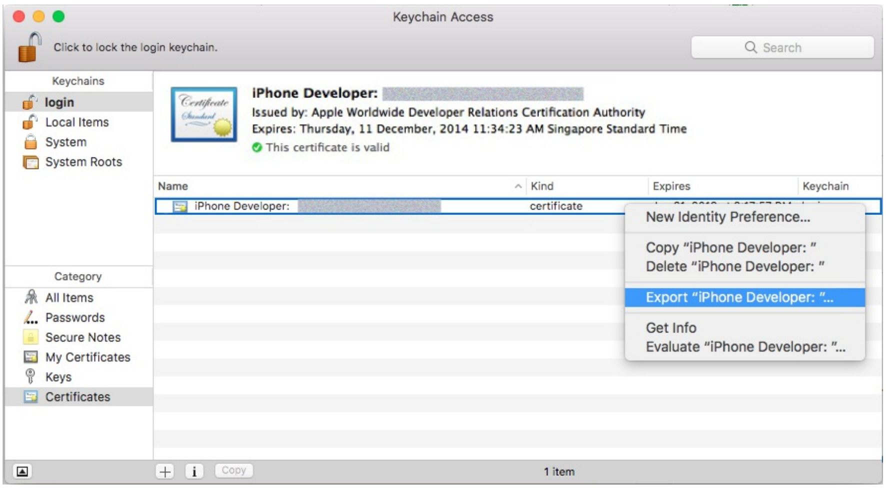


iPhone Developer: [redacted]
Issued by: Apple Worldwide Developer Relations Certification Authority
Expires: Thursday, 11 December, 2014 11:34:23 AM Singapore Standard Time
This certificate is valid

Name	Kind	Expires	Keychain
iPhone Developer: [redacted]	certificate	[redacted]	login

Fig 3: Keychain Access

3. Right-click on the certificate and select **Export "iPhone Developer: XXX"**.

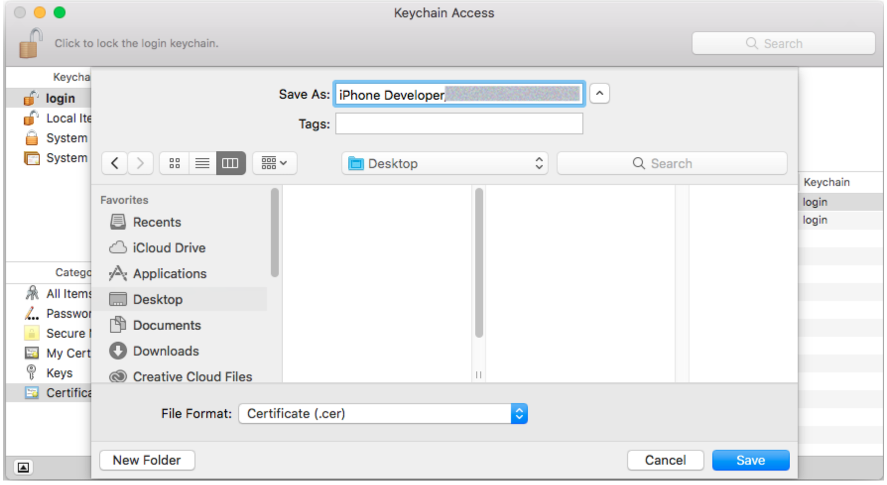


iPhone Developer: [redacted]
Issued by: Apple Worldwide Developer Relations Certification Authority
Expires: Thursday, 11 December, 2014 11:34:23 AM Singapore Standard Time
This certificate is valid

Keychain Access
Click to lock the login keychain.
Search
Keychains
login
Local Items
System
System Roots
Category
All Items
Passwords
Secure Notes
My Certificates
Keys
Certificates
Name
Kind
Expires
Keychain
iPhone Developer: [redacted]
certificate
New Identity Preference...
Copy "iPhone Developer: "
Delete "iPhone Developer: "
Export "iPhone Developer: "...
Get Info
Evaluate "iPhone Developer: "...
Copy
1 item

Fig 4: Export Certificate

4. Make sure you set the **File Format** to **Certificate (.cer)**.



Save As: iPhone Developer [redacted]
Tags:
File Format: Certificate (.cer)

Fig 5: Export Certificate

5. Choose your preferred directory and click **Save** to export your certificate.

3.8 Push Notification Certificate

For using push notification in V-OS Cloud solution, you need to provide the following certificates:

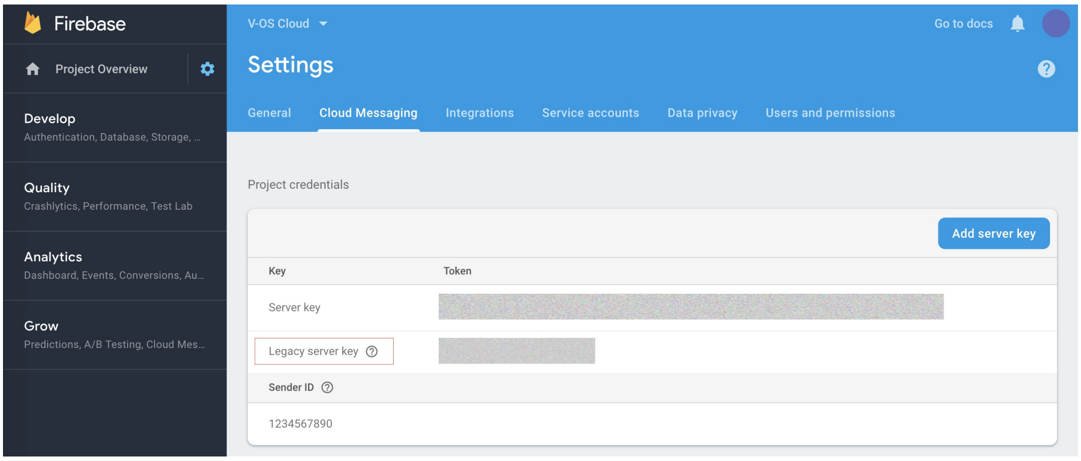
- Android: Firebase Cloud Message Server Key (FCM Certificate)
- iOS: APNS Certificate

3.8.1 Getting FCM Certificate

To get the FCM certificate (Legacy Server Key) of Cloud Message at Firebase, do the following steps:

1. Login to your Firebase account.
2. Go to your project overview.

3. Go to the **Settings** page of your target app.
4. Select the **Cloud Messaging** tab.
5. Obtain the **Legacy server key**.



Key	Token
Server key	[redacted]
Legacy server key	[redacted]

Sender ID

1234567890

Fig 6: Getting Legacy Server Key

3.8.2 Getting APNS Certificate

To get the APNS certificate, do the following steps:

1. On your Mac, open the **Keychain Access** app from the `/Applications/Utilities` folder.
2. On the top menu, select **Keychain Access** → **Certificate Assistant** → **Request a Certificate From a Certificate Authority**. from the **Keychain Access**.

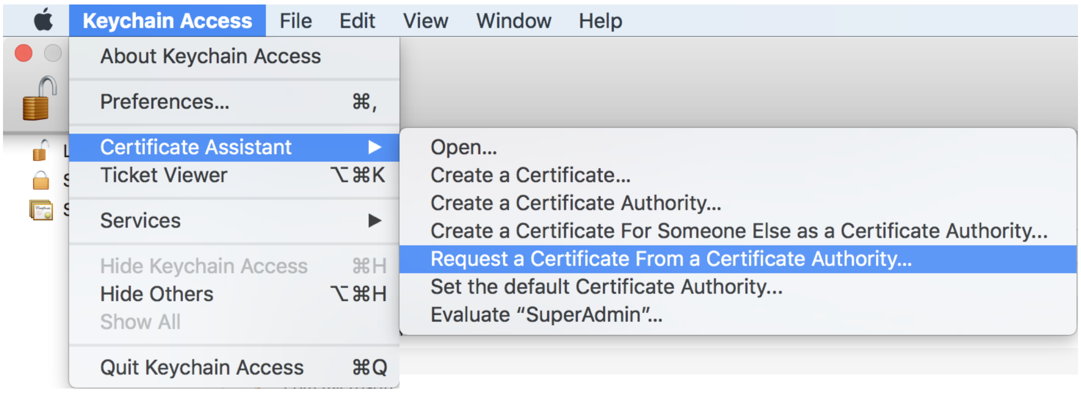
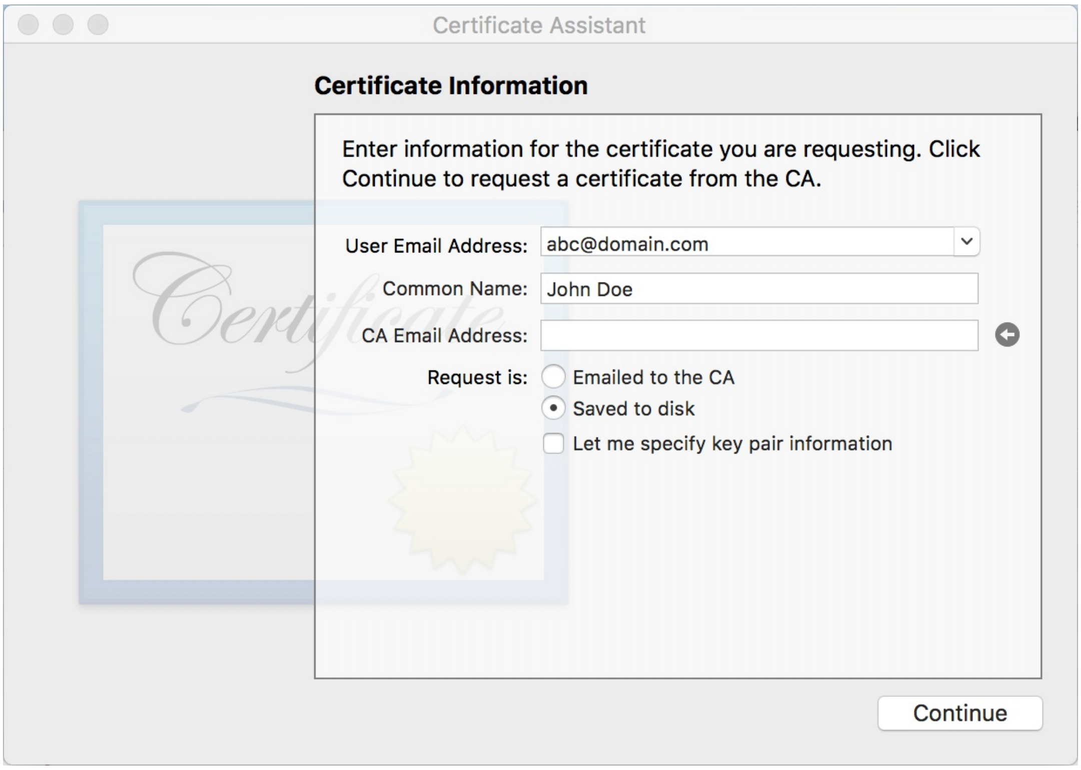


Fig 7: Certificate Assistant Links

3. On the **Certificate Assistant** dialog, fill the **User Email Address** and **Common Name** field as desired.



Certificate Information

Enter information for the certificate you are requesting. Click Continue to request a certificate from the CA.

User Email Address: abc@domain.com

Common Name: John Doe

CA Email Address:

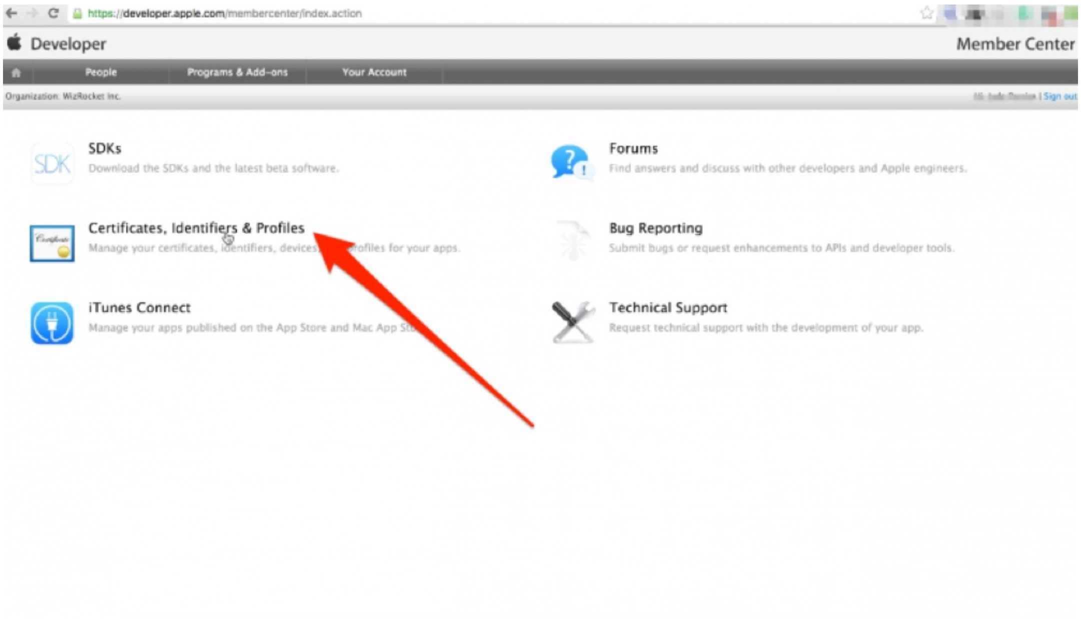
Request is: ○ Emailed to the CA

● Saved to disk

□ Let me specify key pair information

Fig 8: Certificate Assistant

4. Leave the **CA Email Address** field blank.
5. Select the **Save to disk** radio button.
6. Select the **Continue**.
7. Select the **Save** button to save the file to local. This is the CSR file that to be used in the later stage.
8. On your web browser, log in to `developer.apple.com` with you Apple ID.
9. Go to the **Member Center**.
10. Select **Certificates, Identifiers & Profiles**.



Download the SDKs and the latest beta software.

Find answers and discuss with other developers and Apple engineers.

Manage your certificates, identifiers, devices rofiles for your apps.

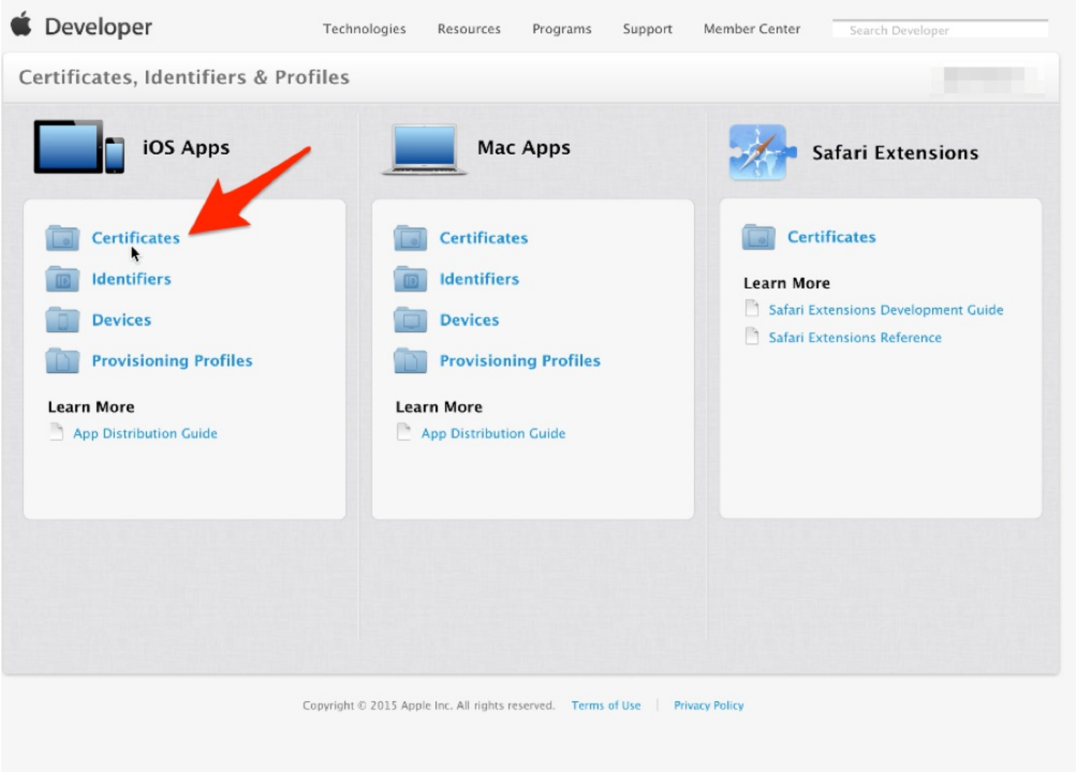
Submit bugs or request enhancements to APIs and developer tools.

Manage your apps published on the App Store and Mac App St

Request technical support with the development of your app.

Fig 9: Certificates, Identifiers & Profiles

11. Select **Certificates** under **iOS Apps**.



Copyright © 2015 Apple Inc. All rights reserved.

Fig 10: Certificates

12. Select **Development** or **Production**, under **Certificates** section on the left menu, depending on which one you want to generate.

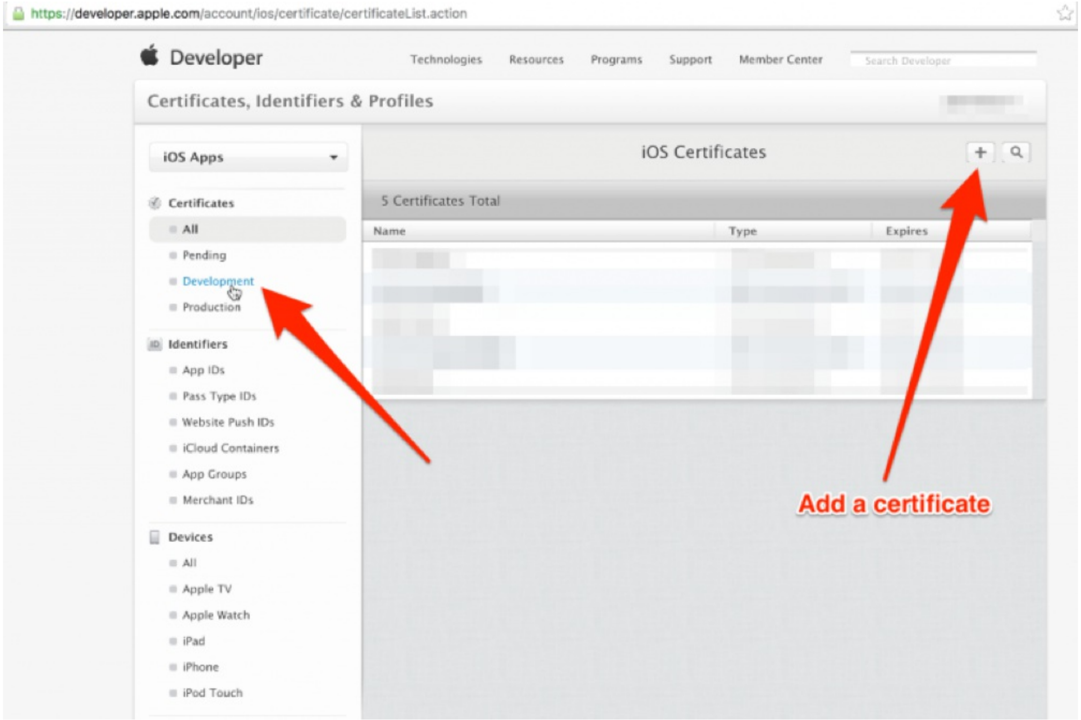
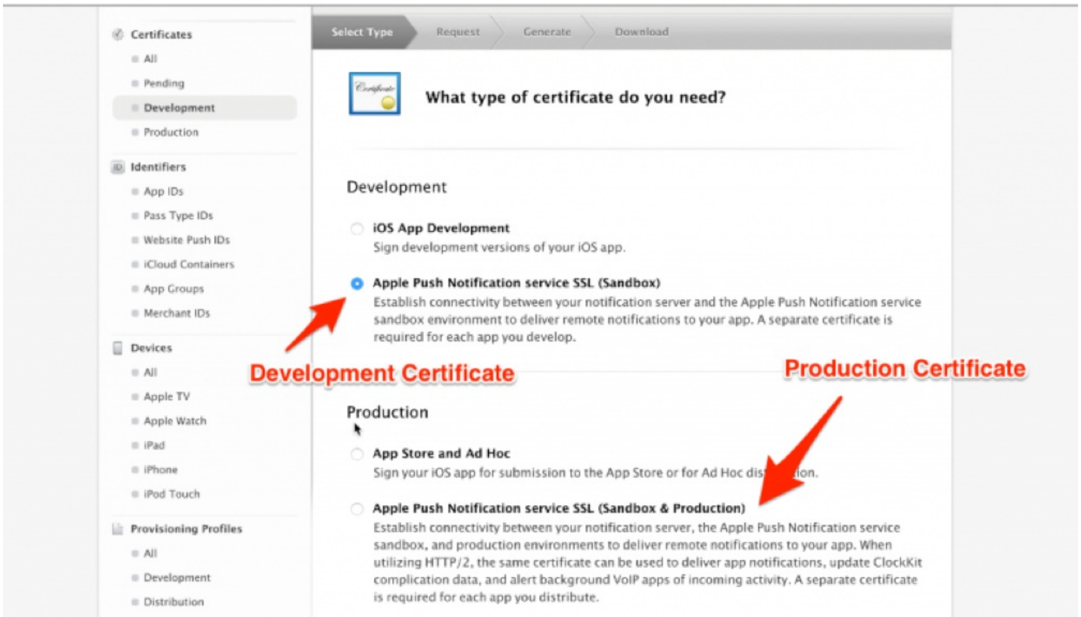


Fig 11: Add Certificate

13. Select the **+** button to add a certificate.
14. Select the **Apple Push Notification service SSL (Sandbox)** checkbox for **Development** or **Apple Push Notification service SSL (Sandbox & Production)** for **Production**.



What type of certificate do you need?

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.

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.

Fig 12: Select Certificate Type

15. Select the **Continue** button.
16. Select the App Bundle ID for which you want to create the Certificate.
17. Upload the CSR file that you have created from **Keychain Access** previously.
18. Download the certificate that has been generated. The certificate downloaded from

`developer.apple.com` will be in the `.cer` format. Continue with the steps to convert the certificate to `.p12` format.

19. Double click on the `.cer` file downloaded to open it in the **Keychain Access** app.
20. In the **Keychain Access** app, right click on the certificate and select **Export ""...**.
21. On the popup dialog, set the **File Format** to **Personal Information Exchange (.p12)**.
22. Select the **Save** button to export your certificate.

3.9 mTLS Requirements

The mTLS feature enables certificate-based server-to-client authentication. In addition to the regular client-to-server authentication, the mTLS feature enables the server to authenticate the client by verifying the client-side X.509 certificate. The certificate proves the client's identity to the server. This feature enhances the app security and adds an extra layer of protection to the users' data. The server data is only accessible to the client if the client is successfully authenticated. If you wish to use the mTLS feature, you need to provide V-Key the CSR template (see Creating CSR Template) for asset generation.

> ***Note:*** *The mTLS feature only supports the following OS version. Inform V-Key about the type of cryptography algorithm that you wish to use.*
>
> *If you use ECC (Elliptic Curve Cryptography), from Android 4.4 and iOS 9 onwards. If you use RSA, from Android 4.3 and iOS 7 onwards.*

3.9.1 Creating CSR Template

To create a CSR template, do the following steps:

1. Create a file with name `csr_template.conf`.
2. Copy the following contents and paste them into the `csr_template.conf` file that you have created.

```
[ req ]
distinguished_name  = req_distinguished_name
req_extensions = v3_req

[ req_distinguished_name ]
countryName     = Country Name (2 letter code)
countryName_default    = SG
countryName_min     = 2
countryName_max     = 2

stateOrProvinceName    = State or Province Name (full name)
stateOrProvinceName_default = SG

localityName      = Locality Name (eg, city)
```

```
localityName_default      = SG

0.organizationName    = Organization Name (eg, company)
0.organizationName_default  = V-Key Pte Ltd

organizationalUnitName    = Organizational Unit Name (eg, section)
organizationalUnitName_default = Crypto
commonName      = Common Name (e.g. server FQDN or YOUR name)
commonName_default = client.mTLS.000000000000004e.v-key.com
commonName_max      = 64

# emailAddress      = Email Address
# emailAddress_default = support@v-key.com
# emailAddress_max    = 64

[ v3_req ]
basicConstraints = CA:FALSE
keyUsage = nonRepudiation, digitalSignature, keyEncipherment
# subjectAltName = @alt_names

# [alt_names]
# DNS.1 = www.example.net
# DNS.2 = www.example.org
```

3. Modify the values in the file, and comment/uncomment certain fields, if necessary.

4 Appendix: V-OS Cloud Environments and URLs

V-OS Cloud gives you two environments, namely Staging (STG) and Production (PRO). STG is the staging environment where you can deploy the solution and do testing (IT/ST and UAT) on. PRO is the production environment where your solution will be hosted officially. The environments are hosted on Amazon Web Services (AWS) where backup and security are assured.

4.1 STG Environment

Your app must be verified to connect to STG of V-OS Cloud at the following URLs and ports. You need to have license and credential access these URLs:

- V-OS PKI Token Server URL: https://stg-cloud.v-key.com/
- V-OS Smart Token Server URL: https://stg-cloud.v-key.com/vtap
- V-OS Provisioning Server URL: https://stg-cloud.v-key.com/provision
- V-OS TMS URL: https://stg-cloud.v-key.com/tms

4.2 PRO Environment

Your app must be verified to connect to PROD of V-OS cloud at the following URLs and ports. You need to have license and credential access these URLs:

- V-OS PKI Token Server URL: https://cloud.v-key.com/
- V-OS Smart Token Server URL: https://cloud.v-key.com/vtap
- V-OS Provisioning Server URL: https://cloud.v-key.com/provision
- V-OS TMS URL: https://cloud.v-key.com/tms