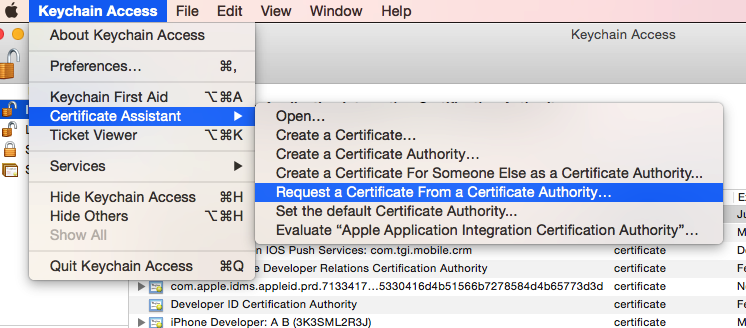
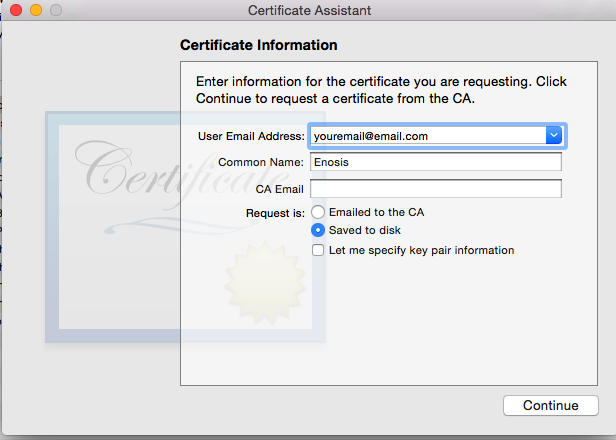
# How to create certificates and provisioning profiles for a push notification enabled app

## Create Certificate Request

Create a certificate request from **keychain access** and save it to your local disk.





### Create Certificate and Provisioning Profile

Login to <http://developer.apple.com>.

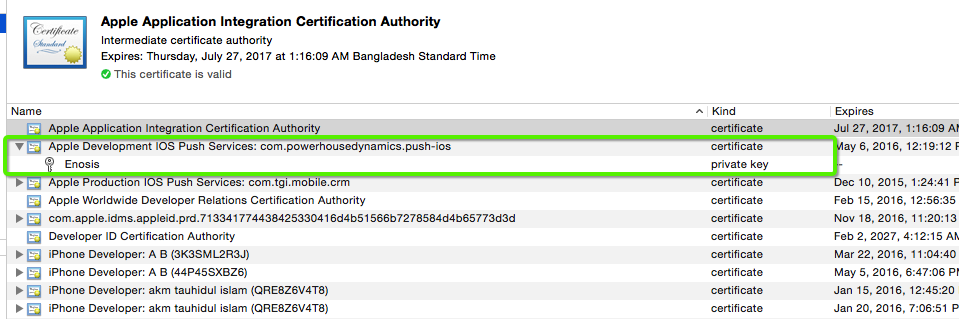
Go to **member center-> Certificates, Identifiers and Profiles->Certificates** and Click “**+**” icon to create a new certificate. When creating e new certificate select **Development**->**Apple Push Notification service SSL (Sandbox)** as the certificate typefor the app **Push.** When creating this you have to show the path of **.cerSigningRequest** created earlier. Download and save it to the disk. It will be named by “**aps\_development.cer**”

Or, use [Create Certificate](https://developer.apple.com/library/ios/documentation/IDEs/Conceptual/AppDistributionGuide/ConfiguringPushNotifications/ConfiguringPushNotifications.html) link to create SSL certificate. There is already one created by us.

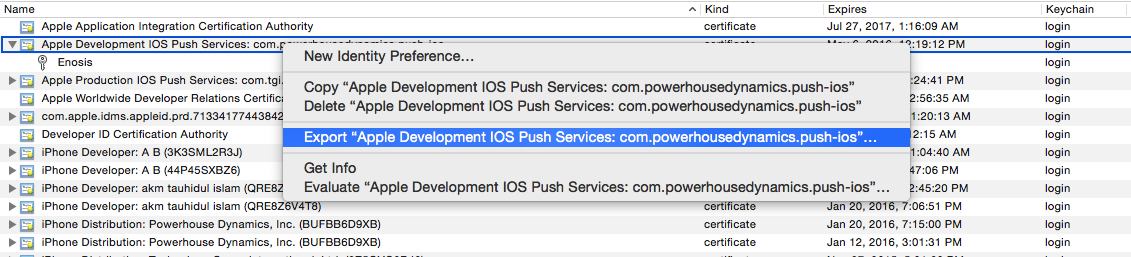
Create a new development provisioning profile selecting at least the newly created certificate.

### Exporting certificate and private key

After downloading these certificate and profiles, double click on the certificate “aps\_development.cer”. Now you will have one entry for this certificate in the keychain like the image below.



Right click on both certificate and private key one by one and export these in **.p12** format.



### Generating PEM file

So now you have three files:

* The CSR(.**cerSigningRequest)** file
* The private key as a p12 file (PushIOSKey.p12)
* The SSL certificate, aps\_development.cer

Store these three files in a safe place. You could throw away the CSR but it is easier to keep it. When your certificate expires, you can use the same CSR to generate a new one. If you were to generate a new CSR, you would also get a new private key. By re-using the CSR you can keep using your existing private key and only the .cer file will change.

You have to convert the certificate and private key into a format that is more usable. Because the push part of our server will be written in PHP, you will combine the certificate and the private key into a single file that uses the PEM format.

The specifics of what PEM is doesn’t really matter (in fact, I have no idea) but it makes it easier for PHP to use the certificate. We’re going to use the command-line OpenSSL tools for this.

Open a Terminal and execute the following steps.

Go to the folder where you downloaded the files, in my case the Desktop:

**$ cd ~/Desktop/**

Convert the .cer file into a .pem file:

**$ openssl x509 -in aps\_development.cer -inform der -out PushIOSCert.pem**

Convert the private key’s .p12 file into a .pem file:

**$ openssl pkcs12 -nocerts -out PushIOSKey.pem -in PushIOSKey.p12  
Enter Import Password:   
MAC verified OK  
Enter PEM pass phrase:   
Verifying - Enter PEM pass phrase:**You first need to enter the passphrase for the .p12 file so that openssl can read it. Then you need to enter a new passphrase that will be used to encrypt the PEM file. Again for this tutorial I used “**pushios**” as the PEM passphrase.

Note: if you don’t enter a PEM passphrase, openssl will not give an error message but the generated .pem file will not have the private key in it.

Finally, combine the certificate and key into a single .pem file:

**$ cat PushIOSCert.pem PushIOSKey.pem > PushIOS.pem**

Now, replace the webservice/PushIOS.pem with newly created one.

And use the created Provisioning Profiles for building the IOS app in XCode.