

Notificaciones Remotas

Introducción

- Notificaciones Push.
- Generadas por sistemas externos, con información de algún hecho remoto.
- Indican que la app tiene información para el usuario.
- El usuario decide si interactúa o no.
- Se perciben como las notificaciones locales.

Ejemplos

- El usuario debe ser informado de algo no programable:
 - Mensaje nuevo.
 - El estado de un pedido ha cambiado.
 - Noticias.

Problema

- El servidor de nuestra app tiene nuevos datos y quiere informar al usuario



Opción 1: Polling

- La app *pregunta* periódicamente.
 - Poco eficiente, gran consumo de batería.
 - Solo funciona en primer plano.



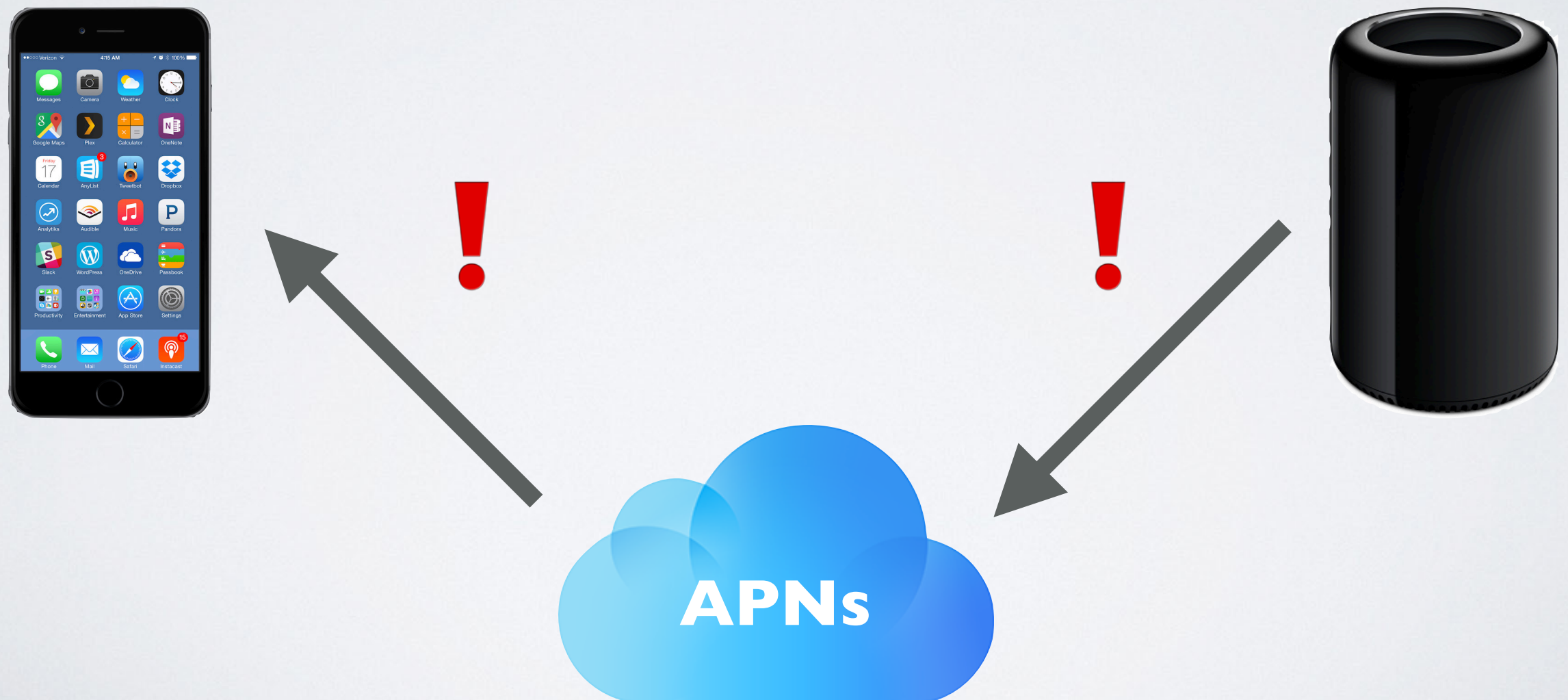
Opción 2: Notificación

- El server notifica puntualmente a la app.



Opción 2: Notificación

- Se realiza a través de Apple Push Notification service (APNs)



Apple Push Notification service

- Seguridad: cifrado, registro de dispositivos, gestión de tokens...
- Quality of Service.
- Agrupado de notificaciones.

Apple Push Notification service

- Best effort.
- Almacena la última notificación si el dispositivo está offline.
- Envía la notificación cuando el dispositivo está online de nuevo.

Pasos a seguir

1. Configurar la app:

1. Activar Push Notifications en App ID y Xcode.

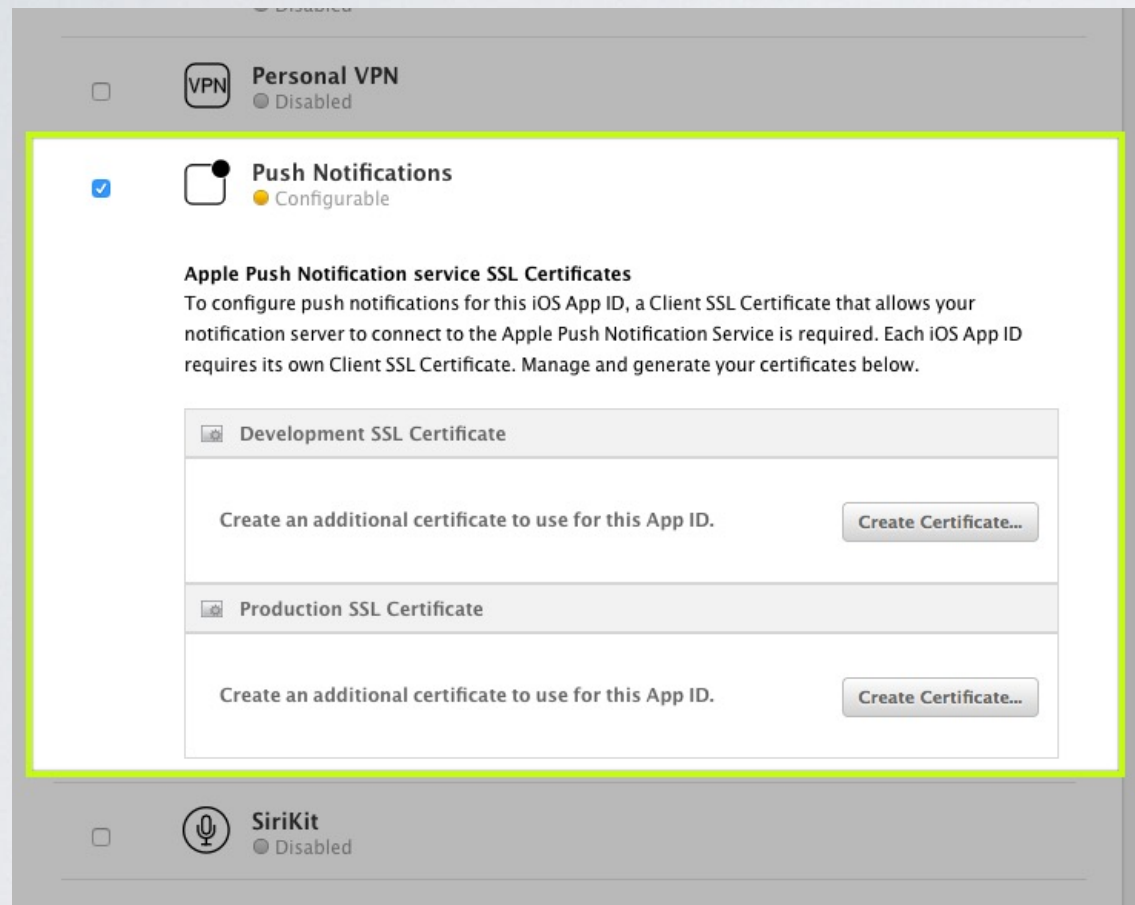
2. Crear certificados para notificaciones push.

2. Pedir permiso al usuario (igual que notificaciones locales).

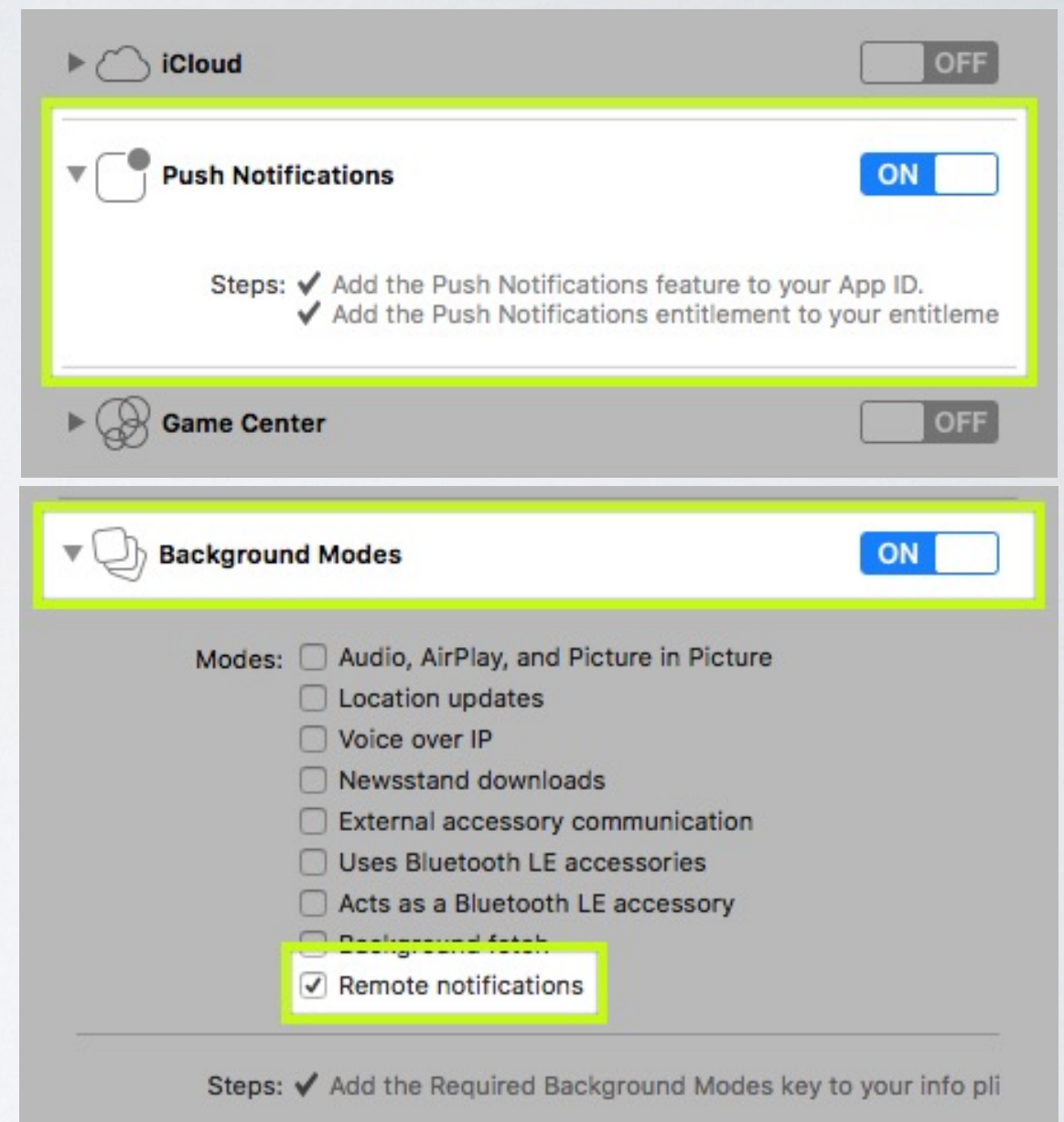
3. Registrar dispositivo para notificaciones push.

4. Manejar notificaciones.

Habilitar Push Notifications




Apple Member Center
App ID



Xcode
Target Capabilities

Crear certificados

 **Developer**

TechnologiesResourcesProgramsSupportMember Center

Search Developer

Certificates, Identifiers & Profiles

iOS Apps

Certificates

All

Pending

Development

Distribution

Identifiers

App IDs

Pass Type IDs

Devices

All

Provisioning Profiles

All

Development

Distribution

11 Certificates Total

Name	Type	Expires
	iOS Development	Oct 12, 2013
	APNs Development iOS	Jun 12, 2013
	APNs Production iOS	Jul 17, 2013
	APNs Production iOS	Sep 27, 2013
	iOS Distribution	Oct 12, 2013
	APNs Development iOS	Jul 17, 2013
	APNs Development iOS	Nov 12, 2013
	APNs Production iOS	Nov 12, 2013
	APNs Production iOS	Nov 15, 2013
	APNs Production iOS	Mar 28, 2014
	APNs Production iOS	Apr 07, 2014

+

Copyright © 2013 Apple Inc. All rights reserved. [Terms of Use](#) | [Privacy Policy](#)

Crear certificados

Certificates

- App IDs
- Pass Type IDs
- Devices
 - All
- Provisioning Profiles
 - All
 - Development
 - Distribution

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. A separate certificate is required for each app you develop.

Distribution


- ☐ **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 (Production)**
Establish connectivity between your notification server and the Apple Push Notification service production environment. A separate certificate is required for each app you distribute.
- ☐ **Pass Type ID Certificate**
Sign and send updates to passes in Passbook.

Intermediate Certificates

To use your certificates, you must have the intermediate signing certificate in your system keychain. This is automatically installed by Xcode. However, if you need to reinstall the intermediate signing certificate click the link below:

 [Worldwide Developer Relations Certificate Authority](#)

Crear certificados

 Technologies Resources Programs Support Member Center

Certificates, Identifiers & Profiles

iOS Apps

Certificates

All

Pending

Development

Distribution

Identifiers

App IDs

Pass Type IDs

Devices

All

Provisioning Profiles

All

Development

Distribution


Add iOS Certificate

Select Type

Request

Generate

Approval



About Creating a Certificate Signing Request (CSR)

To manually generate a Certificate, you need a Certificate Signing Request (CSR) file from your Mac. To create a CSR file, follow the instructions below to create one using Keychain Access.

Create a CSR file.

In the Applications folder on your Mac, open the Utilities folder and launch Keychain Access.

Within the Keychain Access drop down menu, select Keychain Access > Certificate Assistant > Request a Certificate from a Certificate Authority

- In the Certificate Information window, enter the following information:
- In the User Email Address field, enter your email address
- In the Common Name field, create a name for your private key (eg. John Doe Dev Key)
- In the Request is group, select the "Saved to disk" option
- Click Continue within Keychain Access to complete the CSR generating process.

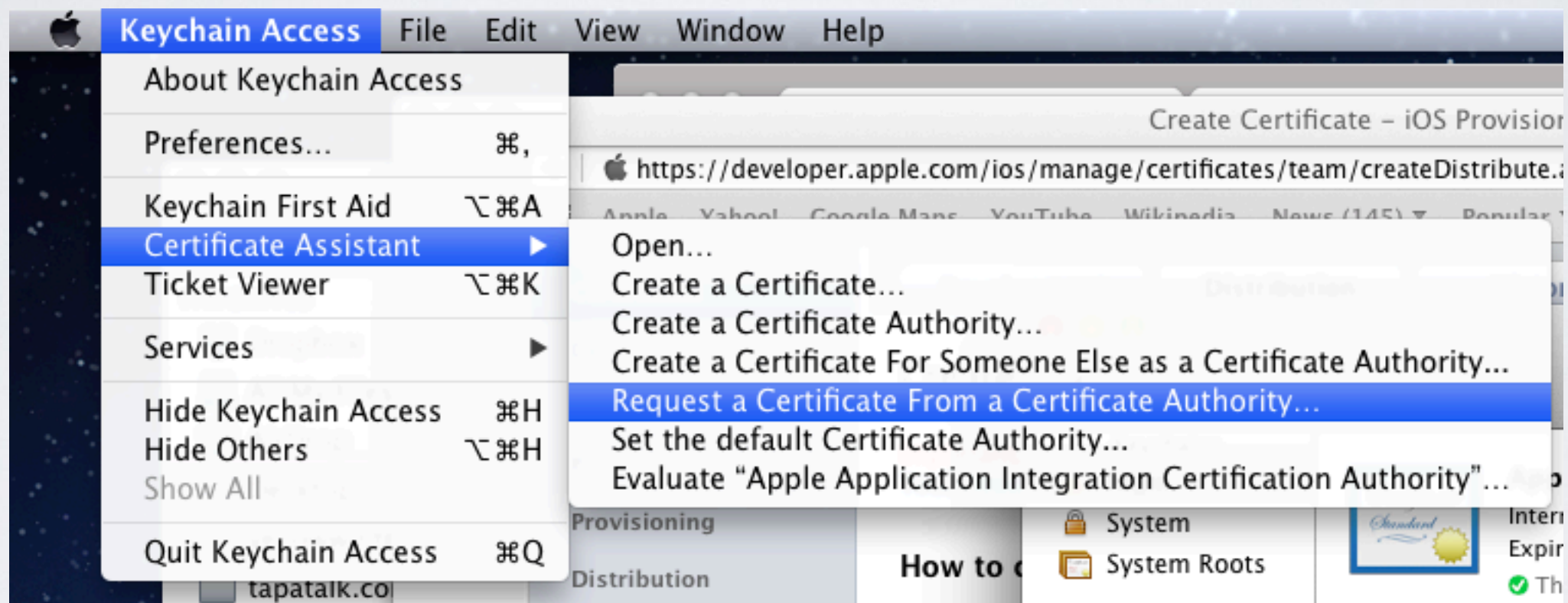
Cancel

Back

Continue

Copyright © 2013 Apple Inc. All rights reserved. [Terms of Use](#) | [Privacy Policy](#)

Crear certificados



Crear certificados

Application Certificate Assistant

Certificate Information

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

User Email Address: user@mydomain.com


Common Name: John Smith

CA Email Address:

Request is: ☐ Emailed to the CA
☒ Saved to disk
☐ Let me specify key pair information

Continue

Crear certificados

 **Developer**

TechnologiesResourcesProgramsSupportMember Center

Search Developer

Certificates, Identifiers & Profiles

iOS Apps

Certificates

AllPendingDevelopmentDistribution

Identifiers

App IDSPass Type IDs

Devices


All

Provisioning Profiles

AllDevelopmentDistribution

Add iOS Certificate

Select TypeRequestGenerateDownload

 **Generate your certificate.**

With the creation of your CSR, Keychain Access simultaneously generated a public and private key pair. Your private key is stored on your Mac in the login Keychain by default and can be viewed in the Keychain Access application under the "Keys" category. Your requested certificate will be the public half of your key pair.

Upload CSR file.
Select .certSigningRequest file saved on your Mac.

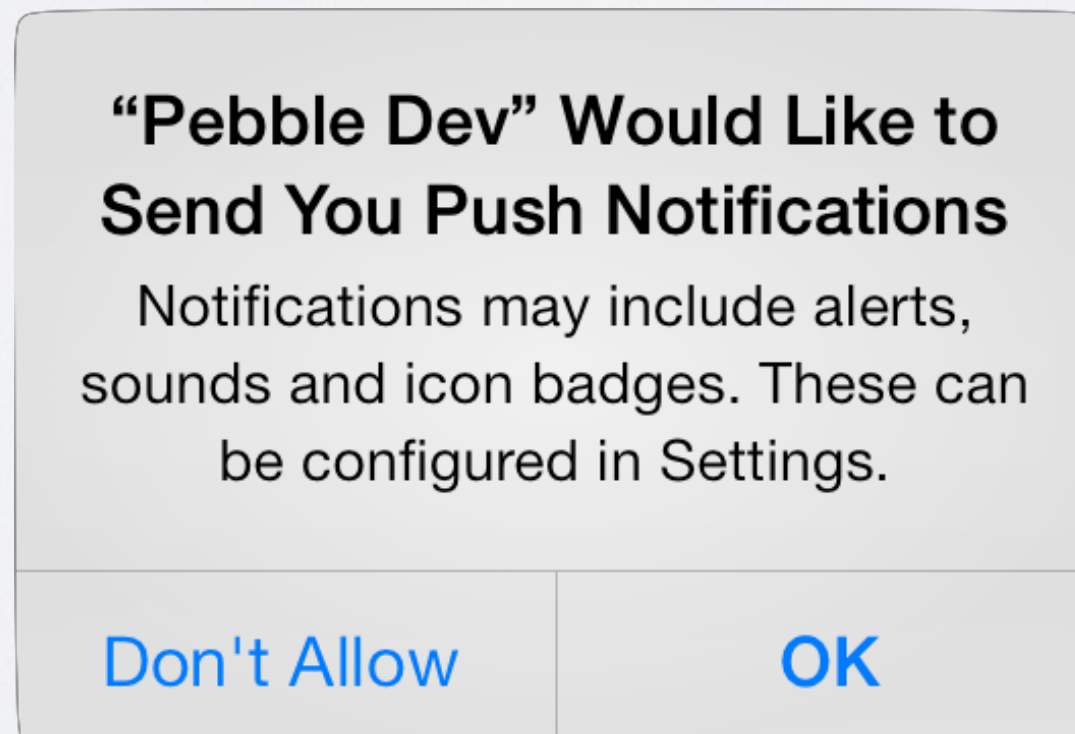
Choose File...

CancelBackGenerate

Copyright © 2013 Apple Inc. All rights reserved. [Terms of Use](#) | [Privacy Policy](#)

Registro de notificaciones

La app **DEBE** pedir permiso al usuario para mostrar notificaciones, indicándole los tipos de notificaciones que habrá.



Registro de notificaciones

```
let notificationCenter = UNUserNotificationCenter.current()

// Request notification permissions
notificationCenter.requestAuthorization(options: [.alert, .sound, .badge]) {
    (granted, error) in
    //Parse errors and track state
    if granted {
        application.registerForRemoteNotifications()
    }
}
```

Registro de notificaciones

```
func application(_ application: UIApplication,  
didRegisterForRemoteNotificationsWithDeviceToken deviceToken:  
Data) {  
    print("My token is: \(deviceToken as NSData)");  
}
```

```
func application(_ application: UIApplication,  
didFailToRegisterForRemoteNotificationsWithError error: Error) {  
    print("Failed to get token, error: \(error)");  
}
```


Manejar notificaciones recibidas

- Protocolo **UNUserNotificationCenterDelegate**.
userNotificationCenter:
didReceiveNotificationResponse:
- Si la app está en segundo plano: se muestra la notificación.
 - Usuario toca el botón por defecto e inicia la app.
 - Se llama al delegado.
- Si la app está en primer plano: sólo se llama al delegado.
- Background fetch: se reciben en **UIApplicationDelegate**.

Manejo de notificaciones: En primer plano

```
func userNotificationCenter(_ center: UNUserNotificationCenter,  
    willPresent notification: UNNotification,  
    withCompletionHandler completionHandler:  
        @escaping (UNNotificationPresentationOptions) -> Void) {  
  
    // Called when a notification is delivered to a foreground app.  
    completionHandler(.alert)  
  
}
```

Manejo de notificaciones:

Acciones

```
func userNotificationCenter(_ center: UNUserNotificationCenter,  
    didReceive response: UNNotificationResponse,  
    withCompletionHandler completionHandler: @escaping () -> Void) {  
  
    // Called to let your app know which action was  
    // selected by the user for a given notification.  
  
    // App will terminate if we fail to call this  
    completionHandler()  
}
```


Manejo de notificaciones: App arrancada de cero

```
func application(_ application: UIApplication, didFinishLaunchingWithOptions
    launchOptions: [UIApplicationLaunchOptionsKey: Any]?) -> Bool {

    if let remotePayload = launchOptions?[UIApplicationLaunchOptionsKey.remoteNotification] {
        NSLog("Launching from remote");
        NSLog("Remote Notification received While in background: \(remotePayload)");
    }

    return true
}
```

Manejo de notificaciones: Background Fetch

```
func application(_ application: UIApplication,  
    didReceiveRemoteNotification userInfo: [AnyHashable : Any],  
    fetchCompletionHandler completionHandler:  
    @escaping (UIBackgroundFetchResult) -> Void) {  
  
    // Handle notification  
    completionHandler(.newData)  
}
```

Testing: Nomad

NOMAD

World-Class Command Line Utilities For iOS Development



```
$ gem install nomad-cli
```


Testing: Houston



HOUSTON

Send push notifications from the command line. Simply provide your credentials, construct your message, and 3...2...1... blastoff.

```
$ apn push "<token>" -c /path/to/cert.pem -m "Hello!"
```

Instalación de nomad-cli

```
$ sudo gem install nomad-cli
```

Si falla intentando instalar *nokogiri*:

```
$ xcode-select --install
```

```
$ brew install libxml2
```

```
$ sudo gem install nokogiri -- --use-system-libraries  
--with-xml2-include=/usr/local/opt/libxml2/include/  
libxml2
```

y finalmente:

```
$ sudo gem install nomad-cli
```


Usando Houston

Ejemplo:

```
$ apn push "<e8b226d6 ... a0f8c3cc>" -c  
pushcert.pem -m "Hello world!"
```

Necesitamos generar el fichero pushcert.pem.

Certificado para testing

 **Developer**

TechnologiesResourcesProgramsSupportMember Center

Search Developer

Certificates, Identifiers & Profiles

iOS Apps

Certificates

All

Pending

Development

Distribution

Identifiers

App IDs

Pass Type IDs

Devices

All

Provisioning Profiles

All


Development

Distribution

iOS Certificates

12 Certificates Total

Name	Type	Expires
	APNs Development iOS	Nov 12, 2013
	APNs Development iOS	Jun 12, 2013
	APNs Production iOS	Jul 17, 2013
	APNs Production iOS	Sep 27, 2013
	iOS Distribution	Oct 12, 2013
	iOS Development	Oct 12, 2013
	APNs Development iOS	Jul 17, 2013
	APNs Production iOS	Nov 12, 2013
	APNs Production iOS	Nov 15, 2013
	APNs Production iOS	Mar 28, 2014
	APNs Production iOS	Apr 07, 2014
	APNs Production iOS	Apr 08, 2014



Name:

Type: APNs Production iOS

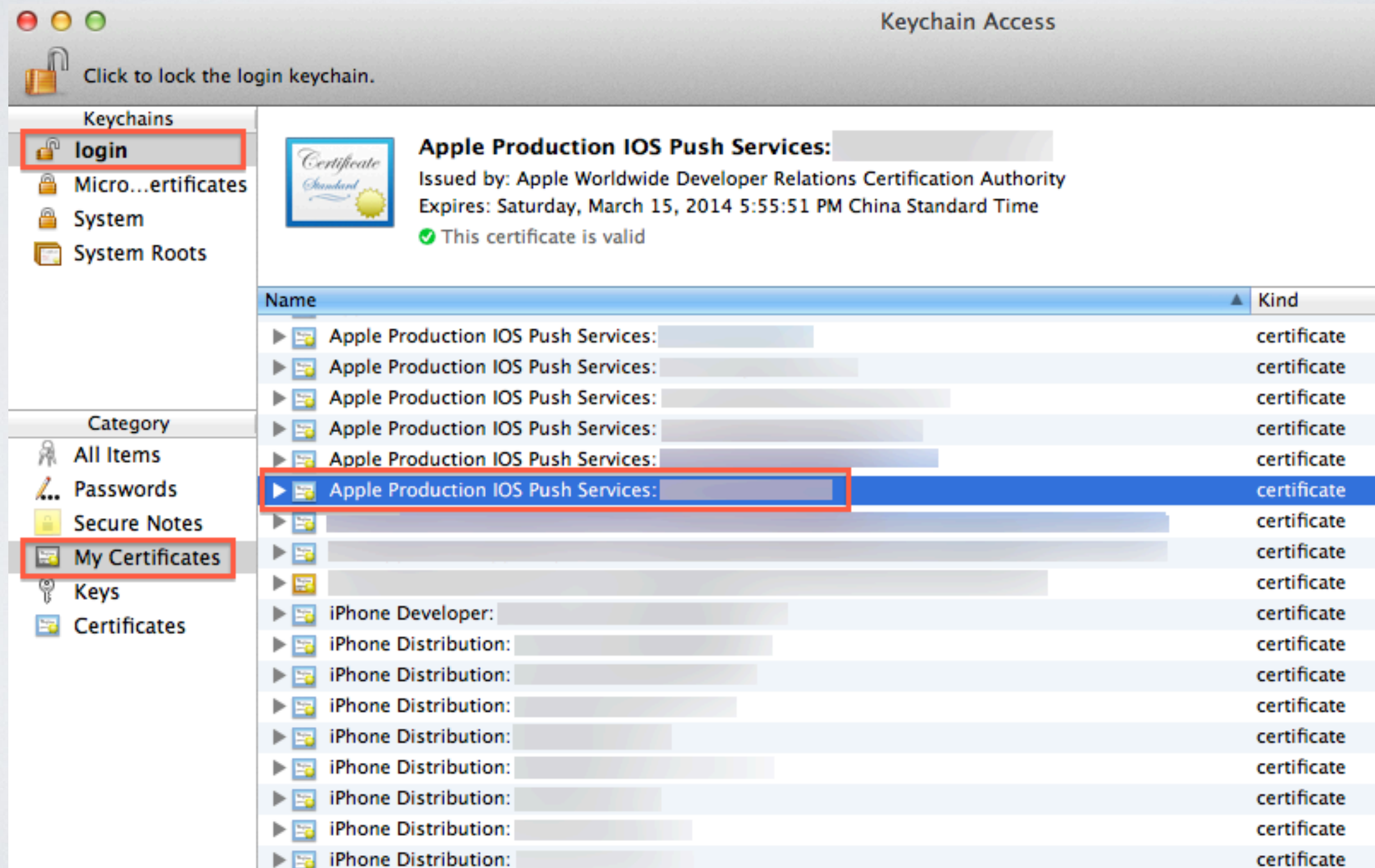
Expires: Mar 28, 2014

Revoke

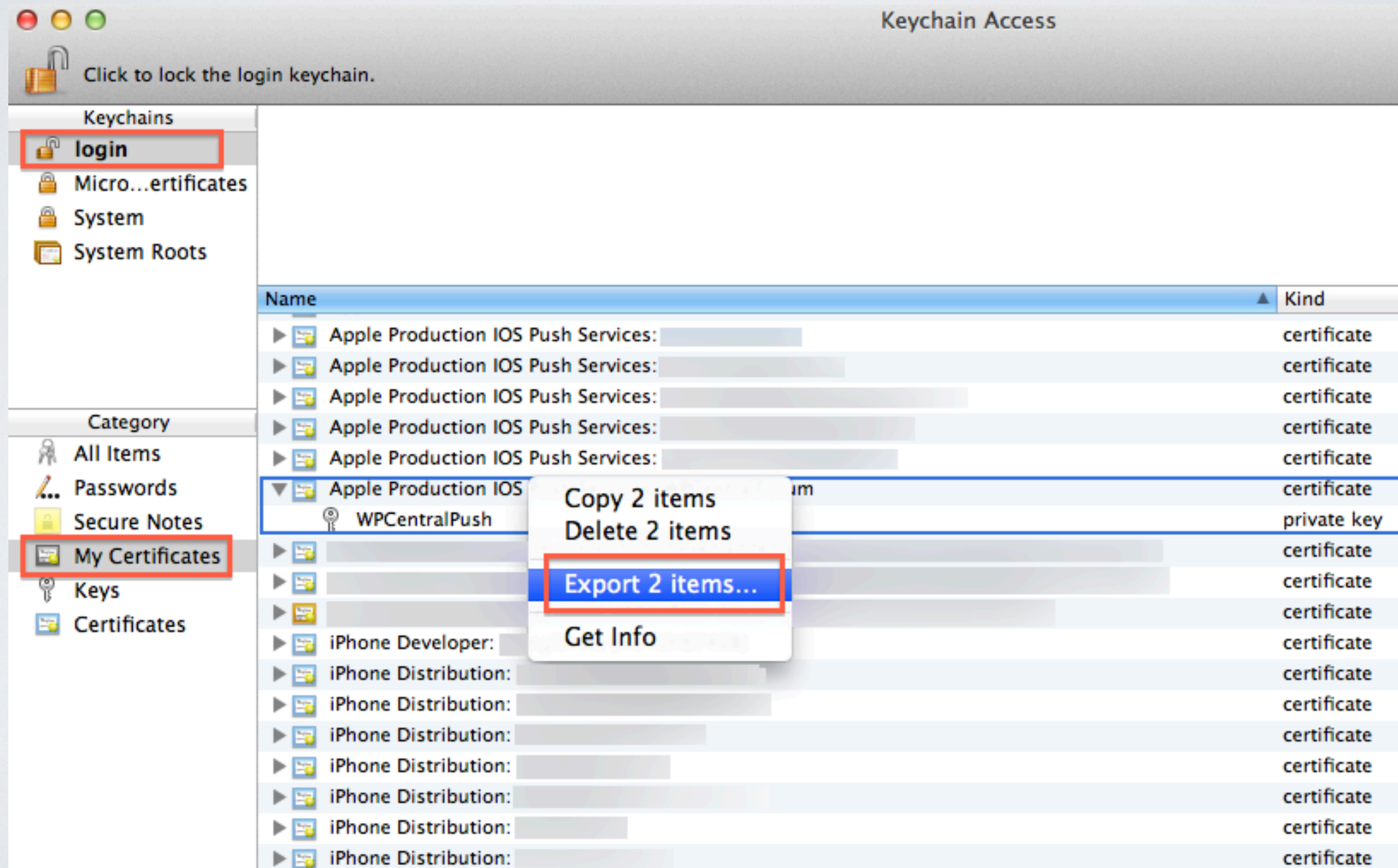
Download

Copyright © 2013 Apple Inc. All rights reserved. [Terms of Use](#) | [Privacy Policy](#)

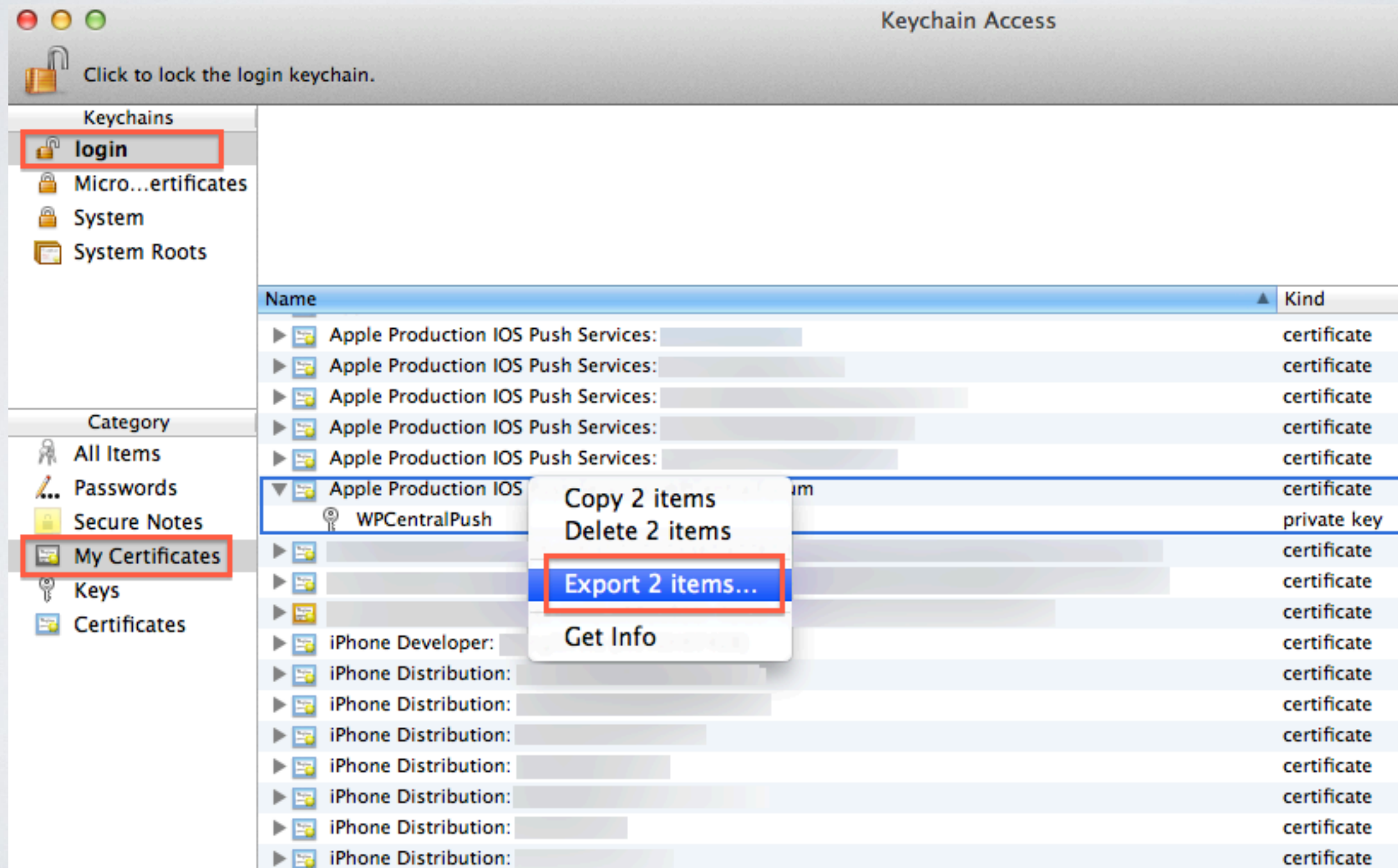
Certificado para testing



Certificado para testing



Certificado para testing



Certificado para testing

- Guardar certificado con extensión **p12**.
- Desde Terminal:

```
openssl pkcs12 -in pushcert.p12 -out  
pushcert.pem -nodes -clcerts
```
- **pushcert.pem** es el certificado que necesitas para enviar notificaciones push de pruebas.

Payload de una notificación

```
{  
  "aps": {  
    "alert": string,  
    "badge": number,  
    "sound": string,  
    "content-available": bool (1/0),  
    "actions": [action, action...],  
    "category": string  
  },  
  app data...  
}
```


Payload de una notificación: diccionario “aps”

Parámetro	Tipo	Descripción
alert	string/dictionary	Si se incluye, se ve la alerta estándar.
badge	number/0	Valor del badge.
sound	string	Sonido a ser reproducido.
content-available	string	Con valor 1 indica contenido disponible.
actions	array	Acciones.
category	string	Identificador de categoría.

Payload de una notificación: diccionario “alert”

Parámetro	Tipo	Descripción
body	string	Mensaje de la alerta.
loc-key	string	Key de texto localizado para el mensaje de la alerta.
loc-args	array	Valores que rellenan los huecos del string de loc-key.
action-loc-key	string	Key de texto localizado para la acción principal.

Payload de una notificación: diccionario en array “actions”

Parámetro	Tipo	Descripción
id	string	Identificador de la acción.
title	string	Título de la acción.
loc-key	string	Key de texto localizado para el título de la acción.
loc-args	array	Valores que rellenan los huecos del string de loc-key.

Payload de una notificación

Ejemplo

```
{  
  "aps": {  
    "alert": "Hello world!",  
    "badge": 2,  
    "sound": "default",  
  },  
  "foo": "bar"  
}
```

Enviar payload con Houston

Usar parámetro -P (**P**ayload) en vez de -m (**m**essage).

Ejemplo:

```
$ apn push "<e8b226d6 ... a0f8c3cc>" -c  
pushcert.pem -P '{ "aps": ... }'
```