

W7D1 Lighthouse Lecture

NotificationCenter vs. User Notifications

- So we've heard about NotificationCenter.
- It uses the Observer Design Pattern to broadcast data from the system (or your own code) to your code in a completely decoupled way.
- The class used for this is called NotificationCenter.
- In contrast to NotificationCenter, there is a mechanism on iOS of user notifications. Don't confuse the 2.
- User notifications enable an app that is no longer running to alert the user about some event.
- User's can tap to launch the app from the notification or ignore them.

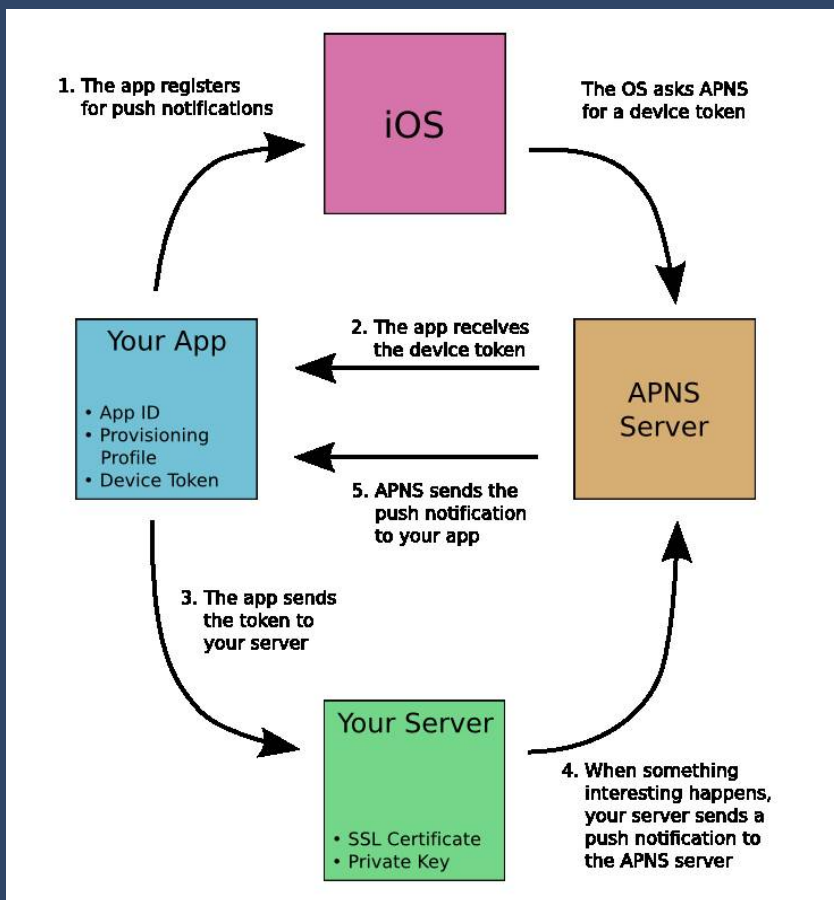
Local vs Remote User Notifications

- Let's say the user wants to setup a calendar event inside your app.
- User notifications can alert the user of this event when your app is not running.
- User notifications can be used to pass the user a message.
- Like out calendar event, or it could be a timer event, location geofencing event, or some important data change on a remote server.
- There are 2 types of user notifications on iOS: `local` and `remote`.
- Local notifications are scheduled and sent by the app itself (no internet needed).
- Local and remote notifications look and sound the same from the user's point of view.

Remote Notifications

- Remote notifications (aka Push Notifications) involve a remote server.
- Data may be sent from a dashboard to a remote service, some automated messaging setup, or it may originate from a user's device.
- Wherever it originates the message must be sent from a remote server to Apple's Push Notification Service (APNS).
- APNS in turn calls the device(s) and sends the payload *securely*.
- Most BAAS systems also include push notification support. This is true of CloudKit, Firebase, and Parse.
- But it is pretty easy to roll your own push notification service.

APNS



UILocalNotifications vs UNUserNotifications

- Note there are 2 Notification API's. UILocalNotifications, and the new system called UNUserNotifications.
- UNUserNotifications was introduced in iOS 10, watchOS 3, and tvOS 10.
- You might need to use both API's in an app that is backward compatible. :(
- UNUserNotifications is way more powerful. (prefer current API's over legacy ones).

Displaying Media & Attachments

- User Notifications can display alerts, and play default or custom sounds to the user, as well as badge the app icon with a number.
- `UNUserNotifications` supports media attachments with the `UNNotificationAttachment` object. This allows attaching audio, video and images to your notifications.
- These attachments can't be sent remotely. Instead you will use a notification service extension to modify your remote notification before it is delivered which allows you to pull in resources locally or remotely. (see [UNNotificationServiceExtension](#) for more details.)

Configuring UNUserNotifications

1. Request authorization using `requestAuthorization(options:completionHandler:)` (same for both local and remote notifications)
2. Create and configure a `UNMutableNotificationContent` object with the notification details.
3. Create a `UNCalendarNotificationTrigger`, `UNTimeIntervalNotificationTrigger`, or `UNLocationNotificationTrigger` object to describe the conditions under which the notification is delivered.
4. Create a `UNNotificationRequest` object with the content and trigger information.
5. Call the `addNotificationRequest:withCompletionHandler:` method to schedule the notification.



Demo Local Notification

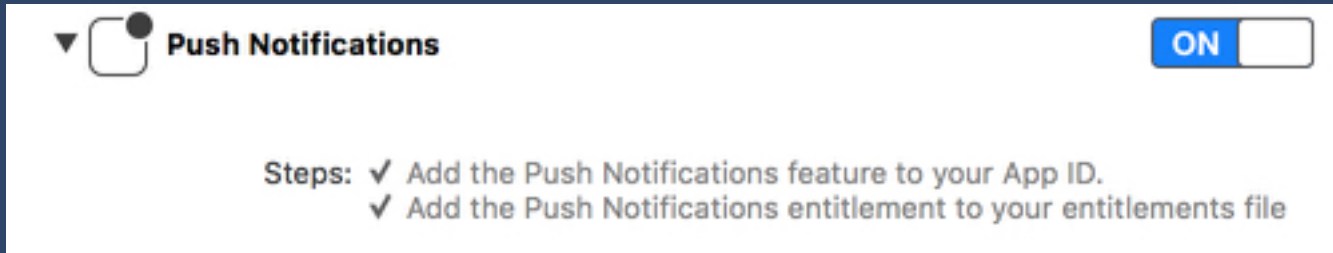
Remote Notifications Requirements

1. RN require a device, and an Apple Developer Membership.
2. We will fake out the server using an app called Pusher.
3. Simple install: `brew cask install pusher`

Setup Push Service

1. Make sure you App's Bundle Identifier is unique.
2. Enable Push Service in the Capabilities Tab of your Xcode project.

Enable Push Service Xcode





Push Service Certificate

1. Log in to <https://developer.apple.com/> and verify that your app has push notifications enabled (it will be amber).
 2. Edit the permissions and click the *Create Certificate...* button for the *Development* type. Follow the instructions.
 3. Add this downloaded certificate to the Mac keychain login.
 4. Restart Pusher. Find the entry in Pusher and add the token.
- Test!

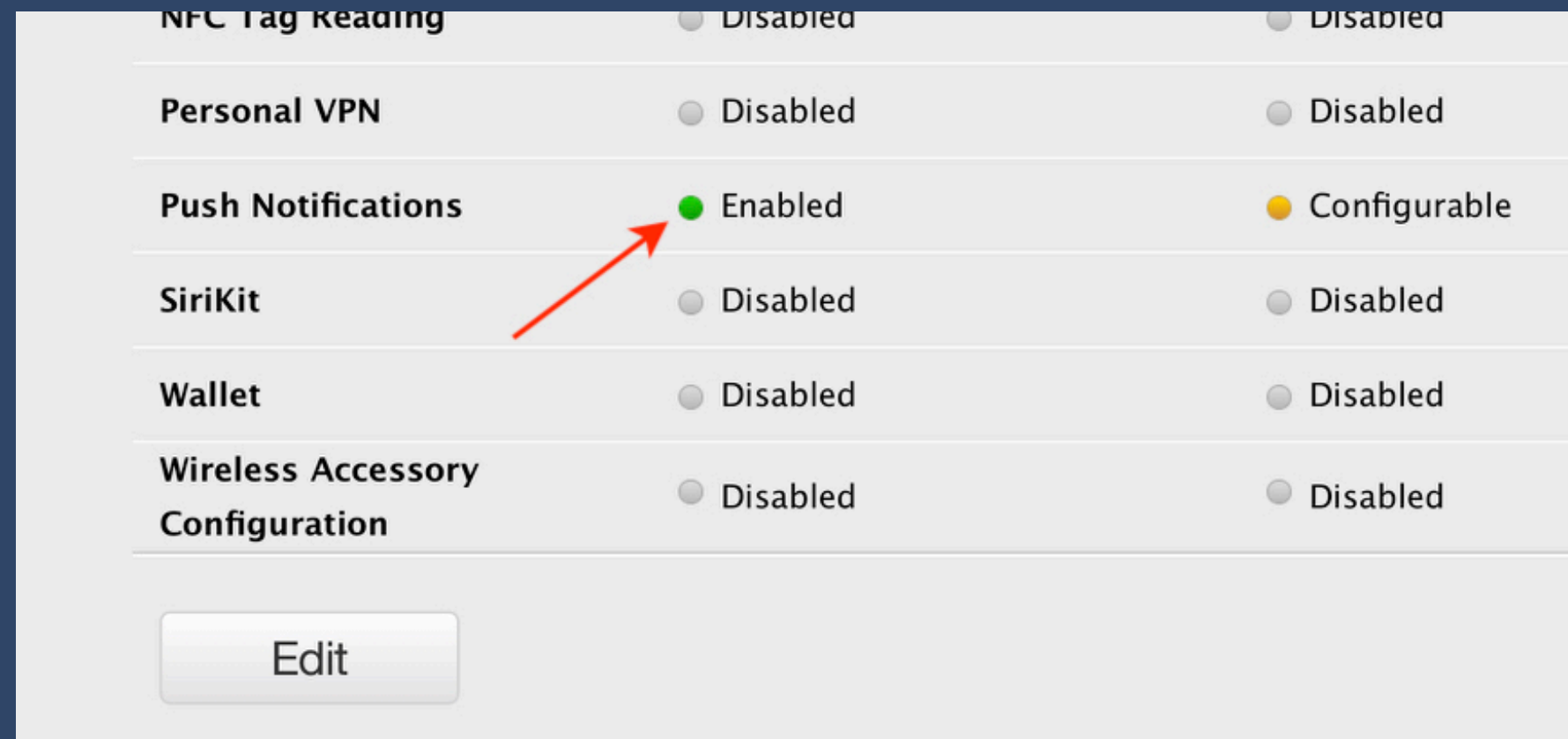
Create Certificate

Apple Push Notification service SSL Certificates

To configure push notifications for this iOS App ID, a Client SSL Certificate that allows your notification server to connect to the Apple Push Notification Service is required. Each iOS App ID requires its own Client SSL Certificate. Manage and generate your certificates below.

 Development SSL Certificate	
Create an additional certificate to use for this App ID.	Create Certificate...
 Production SSL Certificate	

Push Notifications Enabled



After you upload the certificate the amber light should turn to green.

Codes

```
import UIKit
import UserNotifications

@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate {

    var window: UIWindow?

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


More...

```
extension AppDelegate {  
    // Register to receive notifications  
  
    private func registerForNotifications(remote: Bool = false) {  
        UNUserNotificationCenter.current().requestAuthorization(options: [.alert, .sound, .badge]) {  
            (granted, error) in  
            print(#line, "Permission granted: \(granted)")  
            guard granted else { return }  
            self.checkNotificationSettings(remote: remote)  
        }  
    }  
}
```

More...

```
private func checkNotificationSettings(remote: Bool) {
    UNUserNotificationCenter.current().getNotificationSettings { (settings) in
        print(#line, "Settings: \(settings)")
        guard settings.authorizationStatus == .authorized else {
            // Good place to notify the user how not having notifications might affect their experience
            return
        }
        // Register for Remove Notifications
        if remote {
            DispatchQueue.main.async {
                UIApplication.shared.registerForRemoteNotifications()
            }
        }
    }
}
```

More...

```
// Remove notification Callbacks

func application(_ application: UIApplication,
                 didRegisterForRemoteNotificationsWithDeviceToken deviceToken: Data) {
    let tokenParts = deviceToken.map { data -> String in
        return String(format: "%02.2hhx", data)
    }
    let token = tokenParts.joined()
    print("Device Token: \(token)")
    print(#line, token)
    // you could send this token to a server, but don't rely on it because it will change if the user deletes the app for example
}

func application(_ application: UIApplication,
                 didFailToRegisterForRemoteNotificationsWithError error: Error) {
    print("Failed with error: \(error)")
}
}
```

References

- [Local and Remote Notification Programming Guide](#)
- [Background Notifications.](#)
- [Push Notifications Tutorial](#)

CloudTracker

REST

Requests