Core Location Manager Changes in iOS 8

The `CLLocationManager`, introduced in iPhone OS 2, has always worked the same way: Create, delegate, start, wait.

```
// Import CoreLocation framework
// Add <CLLocationManagerDelegate> conformance

// Create a location manager
self.locationManager = [[CLLocationManager alloc] init];
// Set a delegate to receive location callbacks
self.locationManager.delegate = self;
// Start the location manager
[self.locationManager startUpdatingLocation];

// Wait for location callbacks
- (void)locationManager:(CLLocationManager *)manager didUpdateLocations:(NSArray *)locations
{
    NSLog(@"%@", [locations lastObject]);
}
```

The only big change has been a different delegate method to listen for updates, but the old method continues to work. There was always the frustration of it failing silently when you forgot to set its delegate or mistyped the delegate method's signature, but these mistakes are well known and well documented.

Location Manager fails to start in iOS 8

In iOS 8, this code doesn't just fail, it fails silently. You will get no error or warning, you won't ever get a location update and you won't understand why. Your app will never even ask for permission to use location. I'm experienced with Core Location and it took me 30 minutes to figure out the new behavior. Baby devs are going to be even more frustrated by this change.

iOS 8 requirements

In iOS 8 you need to do two extra things to get location working: Add a key to your Info.plist and request authorization from the location manager asking it to start. There are two Info.plist keys for the new location authorization. One or both of these keys is required. If neither of the keys are there, you can call `startUpdatingLocation` but the location manager won't actually start. It won't send a failure message to the delegate either (since it never started, it can't fail). It will also fail if you add one or both of the keys but forget to explicitly request authorization.

So the first thing you need to do is to add one or both of the following keys to your Info.plist file:

- `NSLocationWhenInUseUsageDescription`
- `NSLocationAlwaysUsageDescription`

Both of these keys take a string which is a description of why you need location services. You can enter a string like "Location is required to find out where you are" which, as in iOS 7, can be localized in the InfoPlist.strings file.

Next you need to request authorization for the corresponding location method, WhenInUse or Background. Use one of these calls:

```
[self.locationManager requestWhenInUseAuthorization]
[self.locationManager requestAlwaysAuthorization]
```

Here's a full implementation of a view controller with a location manager asking for WhenInUse (foreground) authorization. This is a basic example which doesn't take into account whether the user has already denied or restricted location.

```
#import "ViewController.h"
@import CoreLocation;

@interface ViewController () <CLLocationManagerDelegate>
@property (strong, nonatomic) CLLocationManager *locationManager;
@end

@implementation ViewController

- (void)viewDidLoad
{
    [super viewDidLoad];

    // ** Don't forget to add NSLocationWhenInUseUsageDescription in MyApp-Info.plist and give it a string

    self.locationManager = [[CLLocationManager alloc] init];
    self.locationManager.delegate = self;
    // Check for iOS 8. Without this guard the code will crash with "unknown selector" on iOS 7.
    if ([self.locationManager respondsToSelector:@selector(requestWhenInUseAuthorization)]) {
        [self.locationManager requestWhenInUseAuthorization];
    }
    [self.locationManager startUpdatingLocation];
}

// Location Manager Delegate Methods
- (void)locationManager:(CLLocationManager *)manager didUpdateLocations:(NSArray *)locations
{
    NSLog(@"%@", [locations lastObject]);
}

@end
```

Authorization Types

There are two kinds of authorization that you can request, WhenInUse and Always. WhenInUse allow the app to get location updates only when the app is in the foreground. Always authorization allows the app to receive location updates both when the app is in the foreground and in the background (suspended or terminated). If you want to use any background location updates, you need Always authorization. An app can include plist keys for either one or both authorization types. If an app needs foreground location for it's main functionality, but has some secondary functionality that needs background location add both keys. If the app needs background functionality as a main requirement, only all Always.

Receiving Always authorization will also give you WhenInUse but not vice-versa. If your app can function with just foreground location but you have some parts that need Always (e.g. optional geofencing functionality) you would add both keys to the plist and start by requesting WhenInUse, then later asking for Always authorization. Unfortunately, this isn't as simple as just calling `[self.locationManager requestAlwaysAuthorization]` at some later point (after getting WhenInUse authorization). Unless the user has never been asked for authorization (`kCLAuthorizationStatusNotDetermined`) you have to ask them to change it manually in the Settings.

Here are the location types that you need Always authorization to use:

- Significant Location Change
- Boundary Crossing (Geofences)
- Background Location Updates (e.g. Fitness, Navigation apps)
- iBeacons
- Visited Locations (iOS 8+)
- Deferred Location Updates

All these location types have the power to wake the app from suspended or terminated when a location event occurs. If you have WhenInUse authorization, these services will work, but only when the app is in the foreground. There is one exception to this rule. In iOS 8, your app can request a local notification for when the user crosses a set location boundary (geofence). When the user crosses the boundary the system sends a local notification from your app but without waking your app. If the user chooses to open the notification your app will then open, receive the boundary information and be able to use foreground location services as normal.

Authorization Statuses

There are two new authorization statuses in iOS 8. Here's a description of all statuses:

- `kCLAuthorizationStatusNotDetermined` User hasn't yet been asked to authorize location updates
- `kCLAuthorizationStatusRestricted` User has location services turned off in Settings (Parental Restrictions)
- `kCLAuthorizationStatusDenied` User has been asked for authorization and tapped "No" (or turned off location in Settings)
- `kCLAuthorizationStatusAuthorized` User has been asked for authorization and tapped "Yes" on iOS 7 and lower.
- `kCLAuthorizationStatusAuthorizedAlways = kCLAuthorizationStatusAuthorized` User authorized background use.
- `kCLAuthorizationStatusAuthorizedWhenInUse` User has authorized use only when the app is in the foreground.

Note that the pre-iOS 8 status "Authorized" is the same code as the iOS 8 status "Always". This means that if an app has received location permission before the user updates to iOS 8, the app will automatically receive Always (background location) authorization. If the app hasn't been built to work with the iOS 8 APIs, on first run (on iOS 8) it will ask for Always authorization by default with the message:

> "Allow YourApp to access your location even when you are not using the app?"

The system may ask a second question at some later stage (saying that it's been using your location in the background) if the app does use location in the background. This question will pop up as an alert, generally appearing in the Springboard. Obviously it's sensible to update your code for iOS 8 to prevent the user from denying permissions they regard as too permissive. Paranoid users may go through the location settings changing all permissions to "When In Use", so you should be prepared to re-request "Always" if it's necessary for your app.

Adding Always Authorization

If you have WhenInUse but later need Always authorization, you will need to ask the user to go to the settings and change it manually. You can use the new iOS 8 URL String `UIApplicationOpenSettingsURLString` to open directly in the correct place in the Settings (this code won't work on iOS 7):

```
- (void)requestAlwaysAuthorization
{
    CLAuthorizationStatus status = [CLLocationManager authorizationStatus];

    // If the status is denied or only granted for when in use, display an alert
    if (status == kCLAuthorizationStatusAuthorizedWhenInUse || status == kCLAuthorizationStatusDenied) {
        NSString *title;
        title = (status == kCLAuthorizationStatusDenied) ? @"Location services are off" : @"Background location is not enabled";
        NSString *message = @"To use background location you must turn on 'Always' in the Location Services Settings";
        UIAlertView *alertView = [[UIAlertView alloc] initWithTitle:title
                                                            message:message
                                                           delegate:self
                                                  cancelButtonTitle:@"Cancel"
                                                  otherButtonTitles:@"Settings", nil];
        [alertView show];
    }
    // The user has not enabled any location services. Request background authorization.
    else if (status == kCLAuthorizationStatusNotDetermined) {
        [self.locationManager requestAlwaysAuthorization];
    }
}

- (void)alertView:(UIAlertView *)alertView clickedButtonAtIndex:(NSInteger)buttonIndex
{
    if (buttonIndex == 1) {
        // Send the user to the Settings for this app
        NSURL *settingsURL = [NSURL URLWithString:UIApplicationOpenSettingsURLString];
        [[UIApplication sharedApplication] openURL:settingsURL];
    }
}
```

If the status is `kCLAuthorizationStatusRestricted`, the user probably can't do anything to change that setting and should not be alerted. You can tie this method to some user action (such a a button to turn on geofencing) so that the user knows that it is their action that has caused the request for location. Disable the button if the status is Restricted.

Adding Authorization Types

If you upload an app to the store which uses one authorization type but then later decide that you need another, you can add the second plist key. Here are the rules for changing authorization type (these are mostly guesses):

- The system will preserve authorization if the accepted mode is still supported by the new key
- If you support WhenInUse and add Always, WhenInUse will still work. Always will need to be requested via Settings.
- If you support Always and add WhenInUse, both should work
- If you support both and remove Always it will probably depend on which one the user has granted
- If you support both and remove WhenInUse it will probably work as long as Always had been granted
- If you remove Always and add WhenInUse, WhenInUse will work probably automatically
- If you remove WhenInUse and add Always, app will probably prompt for permission again

Authorization and Maps

It doesn't end with `CLLocationManager`. If you are using MapKit to show maps you will need to add code to request authorization before the map can show the user's location. There are two ways to show the user's location on a map, with the `MKMapView` property `showsUserLocation` or by using an `MKUserTrackingBarButtonItem`. Location will work with either authorization type but only actually needs WhenInUse to function. If you are setting `showsUserLocation` in code, set the plist key and then request authorization before that call:

```
self.locationManager = [[CLLocationManager alloc] init];
self.locationManager.delegate = self;

// TODO: Add NSLocationWhenInUseUsageDescription in MyApp-Info.plist and give it a string

// Check for iOS 8
if ([self.locationManager respondsToSelector:@selector(requestAlwaysAuthorization)]) {
    [self.locationManager requestWhenInUseAuthorization];
}
self.mapView.showsUserLocation = YES;
```

Authorization with MKUserTrackingBarButtonItem

If you are using an `MKUserTrackingBarButtonItem`, you probably won't be changing `showsUserLocation` in code (since that's what the button does) but you still need an opportune place to call `requestWhenInUseAuthorization`. If the arrow button is tapped while there is no authorization, it will spin but never activate location. To make sure this doesn't happen, you have to check the current authorization type and either request authorization or tell the user to go to the settings and turn on location services. The best place to do this is in the `MKMapView` delegate method `mapViewWillStartLocatingUser:` which is called when `showsUserLocation` changes to `YES`. This code is iOS 8+.

```
- (void)viewDidLoad
{
    [super viewDidLoad];

    // Add a user tracking button to the toolbar
    MKUserTrackingBarButtonItem *trackingItem = [[MKUserTrackingBarButtonItem alloc] initWithMapView:self.mapView];
    [self.toolbar setItems:@[trackingItem]];
    // Need a delegate to receive -mapViewWillStartLocatingUser:
    self.mapView.delegate = self;

    self.locationManager = [[CLLocationManager alloc] init];
    self.locationManager.delegate = self;
}
```

```
// MKMapViewDelegate Methods
- (void)mapViewWillStartLocatingUser:(MKMapView *)mapView
{
    // Check authorization status (with class method)
    CLAuthorizationStatus status = [CLLocationManager authorizationStatus];

    // User has never been asked to decide on location authorization
    if (status == kCLAuthorizationStatusNotDetermined) {
        NSLog(@"Requesting when in use auth");
        [self.locationManager requestWhenInUseAuthorization];
    }
    // User has denied location use (either for this app or for all apps
    else if (status == kCLAuthorizationStatusDenied) {
        NSLog(@"Location services denied");
        // Alert the user and send them to the settings to turn on location
    }
}
```

Another thing to note is that you can call either `request...Authorization`, but unless the status is `NotDetermined` the call will do nothing. If authorization has already been granted there's no need to call this method, but there's also no harm in calling it.

> If the current authorization status is anything other than `kCLAuthorizationStatusNotDetermined`, this method does nothing and does not call the `locationManager:didChangeAuthorizationStatus:` method.

Here's what `request...Authorization` does in various cases of `kCLAuthorizationStatus`:

- `kCLAuthorizationStatusNotDetermined` Show an alert with your description and buttons to Accept/Cancel
- `kCLAuthorizationStatusRestricted` Nothing happens and location services don't start
- `kCLAuthorizationStatusDenied` Nothing happens and location services don't start.
- `kCLAuthorizationStatusAuthorized` Location services start without an alert
- `kCLAuthorizationStatusAuthorizedAlways` Location services start without an alert
- `kCLAuthorizationStatusAuthorizedWhenInUse` Location services start (Always included WhenInUse authorization)

Conclusion

This big change in the location manager can be summed up as "the location manager used to automatically ask for permission when it was asked to start (if it needed it) but now you must explicitly ask for that permission at a time of choosing". This gives more control to the developer over when to request permission, but also makes it more complex. To have a working location manager in iOS 8 you have to add appropriate plist keys and ask the location manager for authorization to receive location updates.

For such a small change to authorization, there are a surprising number of edge cases. Hopefully this post helped you to understand how you can use location services while giving your users a sense of control over how their location is used.

More Information

- WWDC 2014 Session 706 "What's New in Core Location"
- Location and Maps Programming Guide
- CLLocationManager Class Reference
- Locate Me Sample Code

Tweet 18