What’s New in Core Location
Privacy, authorization, and a dash of new API

Adam Driscoll, Core Location Engineer
Andrea Guzzo, Core Location Engineer
Allow "Travel Eats" to access your location while you are using the app?
Your location is used to display restaurants around you.

- Don’t Allow
- Allow
- Only While Using the App
- Always Allow
- Don’t Allow

 Afterwards, open the app and enjoy your meal.
Allow "Travel Eats" to access your location while you are using the app? Your location is used to display restaurants around you.

- Don't Allow
- Allow
- Only While Using the App
- Always Allow
- Don't Allow
Always Authorization: Deferred
WhenInUse Authorization: Empowered
Temporary Authorization: New
Ranging: In Context
Allow “Travel Eats” to access your location?
Your location is used to find restaurants around you.

Allow Once
Don’t Allow
Allow While in Use
// Requesting an Authorization prompt

// CLLocationManagers need to have a long lifecycle, so an App Delegate var is ideal

```swift
var locationManager = CLLocationManager()
```

// If you do not need Always authorization, request a prompt for WhenInUse authorization

```swift
locationManager.requestWhenInUseAuthorization()
```

// -- OR --

// If you do need Always authorization, request a prompt for that

```swift
locationManager.requestAlwaysAuthorization()
```
// Requesting an Authorization prompt

// CLLocationManagers need to have a long lifecycle, so an App Delegate var is ideal
var locationManager = CLLocationManager()

// If you do not need Always authorization, request a prompt for WhenInUse authorization
locationManager.requestWhenInUseAuthorization()

// -- OR --

// If you do need Always authorization, request a prompt for that
locationManager.requestAlwaysAuthorization()
Allow "Travel Eats" to access your location?
Your location is used to find restaurants around you.

Allow While in Use

Allow Once
Don't Allow
Provisional Always Authorization

The User Perspective

- User picked While In Use
- Settings shows While In Use

What Your App Sees

- App requested Always
- Delegate receives Always
- Start using Always powers
Provisional Always Authorization

The User Perspective

An event occurs

User prompted: Allow Always?

Start using Always powers

What Your App Sees
Allow “Travel Eats” to access your location even when you are not using the app?

Your location is used to find restaurants around you. If you always allow access, you’ll never miss nearby Top Eats.

Keep Only While Using
Change to Always Allow
Provisional Always Authorization

The User Perspective

An event occurs

User prompted
Allow Always?

What Your App Sees

Start using Always Powers
Provisional Always Authorization

The User Perspective

An event occurs

User prompted
Allow Always?

Start using
Always Powers

What Your App Sees

Agreement

Always
Deferred Always Authorization Details

Nothing until the user says “Always”

Prompt will display later

One shot only

Request can be made
• Directly
• As an upgrade
Event Delivery Details

Delivered if app is granted Always

Not delivered if

• User granted WhenInUse instead
• User has not yet been prompted
• Event replaced by a newer event
• Just stale
Other Platforms

tvOS does not support Always

watchOS does not need Always*

macOS does not have Always or WhenInUse

iPad Apps for Mac may use either Always or WhenInUse
Always Authorization: Deferred
WhenInUse Authorization: Empowered
Temporary Authorization: New
Ranging: In Context
Allow “Travel Eats” to access your location?
Your location is used to find restaurants around you.

Allow Once
Don’t Allow
Allow While in Use
<table>
<thead>
<tr>
<th>Function</th>
<th>Always</th>
<th>WhenInUse</th>
</tr>
</thead>
<tbody>
<tr>
<td>Receive Location</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Range Beacons</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Continue Location in the Background</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Start Location in the Background</td>
<td>✔️</td>
<td></td>
</tr>
<tr>
<td>Use Significant Location Change Monitoring</td>
<td>✔️</td>
<td></td>
</tr>
<tr>
<td>Use Region Monitoring</td>
<td>✔️</td>
<td></td>
</tr>
<tr>
<td>Use Visit Monitoring</td>
<td>✔️</td>
<td></td>
</tr>
</tbody>
</table>
## Authorization, iOS 13

Will Core Location API yield events when my app is **Always** or **WhenInUse**?

<table>
<thead>
<tr>
<th></th>
<th>Always</th>
<th>WhenInUse</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>In use?</strong></td>
<td><img src="check-icon.png" alt="Check" /></td>
<td><img src="check-icon.png" alt="Check" /></td>
</tr>
<tr>
<td><strong>Not in use?</strong></td>
<td><img src="check-icon.png" alt="Check" /></td>
<td></td>
</tr>
</tbody>
</table>
When is an App In Use?

Foreground

Entered Foreground

Entered Background

In Use

Not In Use

Entered Foreground

Entered Background

In Use

Not In Use
When is an App In Use?

Continuous Background Updates

Background Mode: Location updates

- `startUpdatingLocation`
- `allowsBackgroundLocationUpdates = true`
- `allowsBackgroundLocationUpdates = false`

Entered Foreground

In Use

Entered Background

Still In Use

Entered Foreground

Not In Use
When is an App In Use?

Complications on the Watch

Always In Use
When is an App In Use?

Local notification bootstrap

Post a location-triggered notification

Notification becomes relevant

User taps through

Entered Foreground

Entered Background

In Use

Not In Use
### WhenInUse Authorization, iOS 13

<table>
<thead>
<tr>
<th>Will Core Location API yield events when my app is</th>
<th>Always</th>
<th>WhenInUse</th>
</tr>
</thead>
<tbody>
<tr>
<td>In use?</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Not in use?</td>
<td>✔️</td>
<td></td>
</tr>
</tbody>
</table>
Always Authorization: Deferred
WhenInUse Authorization: Empowered
Temporary Authorization: New
Ranging: In Context
Allow "Travel Eats" to access your location?
Your location is used to find restaurants around you.

Allow Once
Don't Allow

Allow While in Use
Authorization States, iOS 12

- .notDetermined
- .authorizedAlways
- .authorizedWhenInUse
- .denied
Allow Once
Temporary authorization

Grants authorization temporarily

Yields `authorizedWhenInUse`

Request authorization again next time

Very sensitive to end-of-use
Temporary Authorization Case Study

Foreground

Entered Foreground

In Use

Not In Use

Entered Background

Entered Foreground

In Use

Not In Use

Entered Background
Temporary Authorization Case Study

Foreground

Entered Foreground

Request Authorization

Entered Background

Entered Foreground

Not In Use

Entered Background

Not In Use
Temporary Authorization Case Study

Foreground

Entered Foreground

In Use

Entered Background

Not In Use

Entered Foreground

In Use

Entered Background

Not In Use
Temporary Authorization Case Study

Foreground

Entered Foreground

In Use

Entered Background

Not In Use

Entered Foreground

In Use

Entered Background

Not In Use
Temporary Authorization Case Study
Continuous Background Updates

startUpdatingLocation
allowsBackgroundLocationUpdates = true
allowsBackgroundLocationUpdates = false

Request Authorization
Entered Foreground
Entered Background
Entered Foreground
Entered Background
In Use
Not In Use
Temporary Authorization Case Study
Continuous foreground updates
Always Authorization: Deferred
WhenInUse Authorization: Empowered
Temporary Authorization: New
Ranging: In Context
Beacon Ranging Overview

API Changes

Example
Beacon Ranging

Introduced in iOS 7 as extension of the Region Monitoring API

Region Monitoring to initiate ranging

Region Monitoring no longer requires Always Authorization
Beacon Ranging API, iOS 12

open func startRangingBeacons(in region: CLBeaconRegion)

open func stopRangingBeacons(in region: CLBeaconRegion)
Beacon Ranging API Updates

- CLBeacon
- CLBeaconRegion
  - uuid
  - *
  - *
- CLRegion
- CLBeacon
  - uuid
  - major
  - minor
- CLBeacon
  - uuid
  - major
  - minor
- CLBeacon
  - uuid
  - major
  - minor
let beaconRegion = CLBeaconRegion(beaconIdentityConstraint: constraint, identifier: myBeaconRegion)
Example: Beacon Ranging in a museum

Install beacons in the exhibition rooms

Determine when a visitor is in an exhibition room

Find which objects are nearby

Location-triggered notification at arrival
When in Use

Monitor for BeaconRegion

Stop Ranging

Start Ranging

Enter Region

Exit Region

Beacons Found

Status Update
When in Use

Monitor for BeaconRegion

Stop Ranging
Start Ranging
Beacons Found
Status Update
Enter Region
Exit Region

UUID: MyMuseum
Major: PaintersRoom
Minor: SomePainting
// Monitor for the beacon-defined region

self.locationManager.requestWhenInUseAuthorization()

// Create a new constraint and add it to the dictionary.
let constraint = CLBeaconIdentityConstraint(uuid: uuid)

/*
By monitoring for the beacon before ranging, the app is more
energy efficient if the beacon is not immediately observable.
*/

let beaconRegion = CLBeaconRegion(beaconIdentityConstraint: constraint,
                                    identifier: uuid.uuidString)

self.locationManager.startMonitoring(for: beaconRegion)
When in Use

- Stop Ranging
- Start Ranging
- Monitor for BeaconRegion
- Enter Region
- Exit Region
- Beacons Found

Status Update
/ React to entering/exiting the beacon-defined region

```swift
func locationManager(_ manager: CLLocationManager, didDetermineState state: CLRegionState, for region: CLRegion)
{
    let beaconRegion = region as? CLBeaconRegion
    if state == .inside {
        // Start ranging when inside a region.
        manager.startRangingBeacons(satisfying: beaconRegion!.beaconIdentityConstraint)
    } else {
        // Stop ranging when not inside a region.
        manager.stopRangingBeacons(satisfying: beaconRegion!.beaconIdentityConstraint)
    }
}
```
When in use

Monitor for BeaconRegion

Enter Region

Start Ranging

Beacons Found

Status Update

Exit Region

Stop Ranging
// Status update for what beacons are in proximity
func locationManager(_ manager: CLLocationManager,
    didRange beacons: [CLBeacon],
    satisfying beaconConstraint: CLBeaconIdentityConstraint)
{
    /* Beacons are categorized by proximity. */
    for range in [CLProximity.immediate, .near, .far, .unknown] {
        let proximityBeacons = beacons.filter { $0.proximity == range }
        // TODO: Do something with the matching beacons
    }
}

On First Launch

- UserNotification as Location-based Trigger

When in Use

- Start Ranging
- Stop Ranging
- Monitor for BeaconRegion

Enter Region

Exit Region

Beacons Found

Status Update
// Setup a location-triggered user notification

let center = CLLocationCoordinate2D(latitude: 37.335400, longitude: -122.009201)
let region = CLCircularRegion(center: center, radius: 2000.0, identifier: "Museum")

region.notifyOnEntry = true
region.notifyOnExit = false

let trigger = UNLocationNotificationTrigger(region: region, repeats: false)
Recap

Beacon Ranging

API Updates

Example: Ranging for beacons in a museum
Summary

We have some new authorization behaviors
Test them
Ranging opens a new way to deliver location-aware experiences
<table>
<thead>
<tr>
<th>Event</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core Location Lab</td>
<td>Wednesday, 11:00</td>
</tr>
<tr>
<td>Designing for Privacy</td>
<td>Wednesday, 2:00</td>
</tr>
<tr>
<td>Core Location Lab</td>
<td>Friday, 1:00</td>
</tr>
</tbody>
</table>

More Information

developer.apple.com/wwdc19/705