Article

Configuring your app to use location services

Prepare your app to start collecting location data.

Overview

The location data available on most Apple devices gives you additional context and information to incorporate into your app's content. You might use that data to show someone's physical location on a map and help them navigate their environment. Or you might apply location data to a list of restaurants and shops to eliminate choices that aren't nearby. You might also use location data to alert someone when they're near a particular device or geographic region. For all of these use cases and more, the Core Location framework provides access to the location data you need.

When you add code to support Core Location, plan for situations where location data isn't available. The system also requires apps to gain permission to use location data, and prevents apps from acquiring locations without permission. If location data is unavailable for any reason, make the best app experience you can without it. Disable features that rely on location data or provide alternatives to get the behavior you need.

Important

Location data is sensitive information, and it's important for you to secure any location data you collect. Encrypt any location data you store on disk or send over the network. In addition, provide a clear privacy policy that explains how you use someone's location data.

Check the availability of services your app uses

Always verify that location services are available before you attempt to use them. Services might be unavailable for many different reasons, including:

- The device is in Airplane mode.
- The device doesn't have the required hardware.
- The device doesn't support a specific service.
- Your app doesn't have authorization to use the service.

If a service isn't available, disable any app-specific features that rely on that service. Disabling features in advance is a more reliable approach than using a service and responding to errors.

The `CLLocationManager` class provides methods to determine the availability of each service. Call the appropriate method for a given service immediately before you try to use that service. For example, an app that offers compass heading information might call the `headingAvailable()` method before starting the service. If your app uses multiple services, call the appropriate method for each service.

```
// Check if heading data is available.
if CLLocationManager.headingAvailable() {
    locationManager.startUpdatingHeading()
} else {
    // Disable compass features.
}
```

If your app can't function without specific location services, declare those requirements in advance using your app's Information Property list. For more information, see Declare the device capabilities your app requires below.

Start receiving location updates and authorization status changes

You can request location updates in your code immediately. If the system hasn't requested authorization for your app, it does so when your code starts iterating over the asynchronous stream of updates. Because location data is sensitive personal information, the owner of a device controls which apps give access. They can grant or deny access to apps separately, and can change an app's access at any time in system settings.

> **Tip**
>
> Begin requesting location updates where your app uses location data, such as from the view that displays location-related data. Don't make requests at launch time or from a part of your app that isn't connected to location, unless absolutely necessary. Someone might not fully understand why your app is asking for authorization and deny the request.

Location updates and authorization status changes arrive in an asynchronous fashion. Check for both the presence of a location update and authorization status changes within the loop. The loop doesn't terminate unless you explicitly use `return`, `break`, or throw an exception.

```
// Obtain an asynchronous stream of updates.
let stream = CLLocationUpdate.liveUpdates()

// Iterate over the stream and handle incoming updates.
for try await update in stream {
     if update.location != nil {
          // Process the location.
     } else if update.authorizationDenied {
          // Process the authorization denied state change.
     } else {
          // Process other state changes.
     }
}
```

If your app always needs authorization, it needs to create and hold a privileged session offered by `CLServiceSession`. This class provides a single opportunity to upgrade from "While using" to "Always."

Declare the device capabilities your app requires

Core Location generates location updates using a combination of Wi-Fi, cellular, and GPS hardware, and it generates compass updates using magnetometer hardware. For location updates, Core Location doesn't use every piece of hardware every time. You specify the level of precision you want in your `CLLocationManager` object, and Core Location turns on the hardware it needs to deliver that data in the most power-efficient way.

If your app can't function without a particular piece of hardware, add the `UIRequiredDeviceCapabilities` key to your app's Information Property List. The presence of this key tells the App Store to prevent the installation of your app on devices without the specified hardware or capabilities. The value of the key is an array of strings, and you can include one of the following strings for location-related requirements: `location-services`, `gps`, or `magnetometer`.

Include the `gps` key only if you require the highest level of precision for location data. Typically, only navigation apps require that kind of precision, but other apps might also need it to ensure precise location is available when needed. If your app requires heading information, include the `magnetometer` key.

Don't include the `UIRequiredDeviceCapabilities` key if people can still use your app without location data. For example, don't include the key if your app uses location data to filter

search results for nearby restaurants. When location data isn't available, you can find alternatives to get you what you need or work without that data. For example, if you want to filter search results by location, you might prompt someone to enter a postal code or other geographic information explicitly.

Start the location services

After you perform the initial checks and verify the authorization status of your app, start the location services you need. Core Location offers several different ways to access location-related information:

- **Get the current location.** Offer navigation instructions, filter data sets based on location, share someone's location with friends, or perform other tasks that use someone's current location. See Getting the current location of a device.
- **Detect when the device enters or exits a geographical region.** Alert someone to points-of-interest, deliver location-sensitive reminders, and more. See Monitoring the user's proximity to geographic regions.
- **Determine the current compass heading.** Offer course-based navigation or display an onscreen compass. See Getting heading and course information.
- **Detect nearby iBeacon hardware.** Determine someone's proximity to Bluetooth devices. See Determining the proximity to an iBeacon device.

See Also

Essentials

Supporting live updates in SwiftUI and Mac Catalyst apps

Enable background events by adding lifecycle event support.

`class` `CLLocationManager`

The object you use to start and stop the delivery of location-related events to your app.

`class` `CLBackgroundActivitySession`

An object that manages a visual indicator that keeps your app in use in the background, allowing it to receive updates or events.

`struct` `CLLocationUpdate`

A structure that contains the location information the framework delivers with each update.

{} Adopting live updates in Core Location

Simplify location delivery using asynchronous events in Swift.

{} Monitoring location changes with Core Location

Define boundaries and act on user location updates.