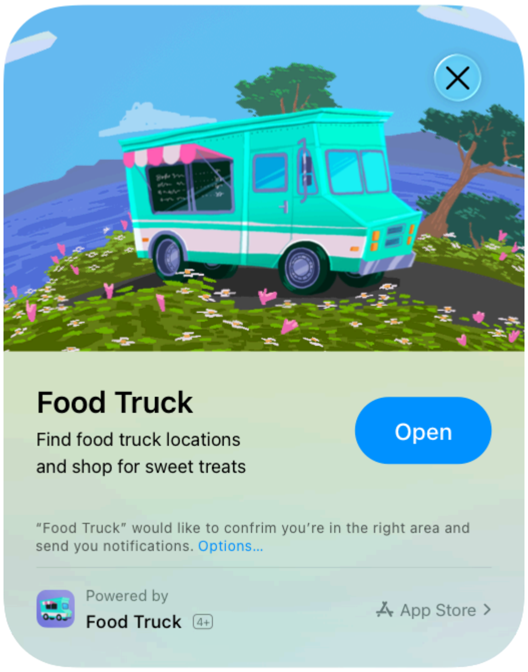
Article

Confirming a person’s physical location

Add code to quickly confirm a person’s physical location while respecting their privacy.

Overview

If you create an App Clip that people invoke at a physical location, you may need to confirm a person’s location before allowing them to perform a task. For a quick launch and to preserve user privacy, App Clips use a lightweight mechanism in which the system verifies that a person is at a specific, expected location. When you adopt this mechanism, and when people allow it, the *App Clip card* contains a note that tells people that the App Clip can verify their location. They can disable location verification by tapping the note on the App Clip card.


Food Truck

Find food truck locations and shop for sweet treats

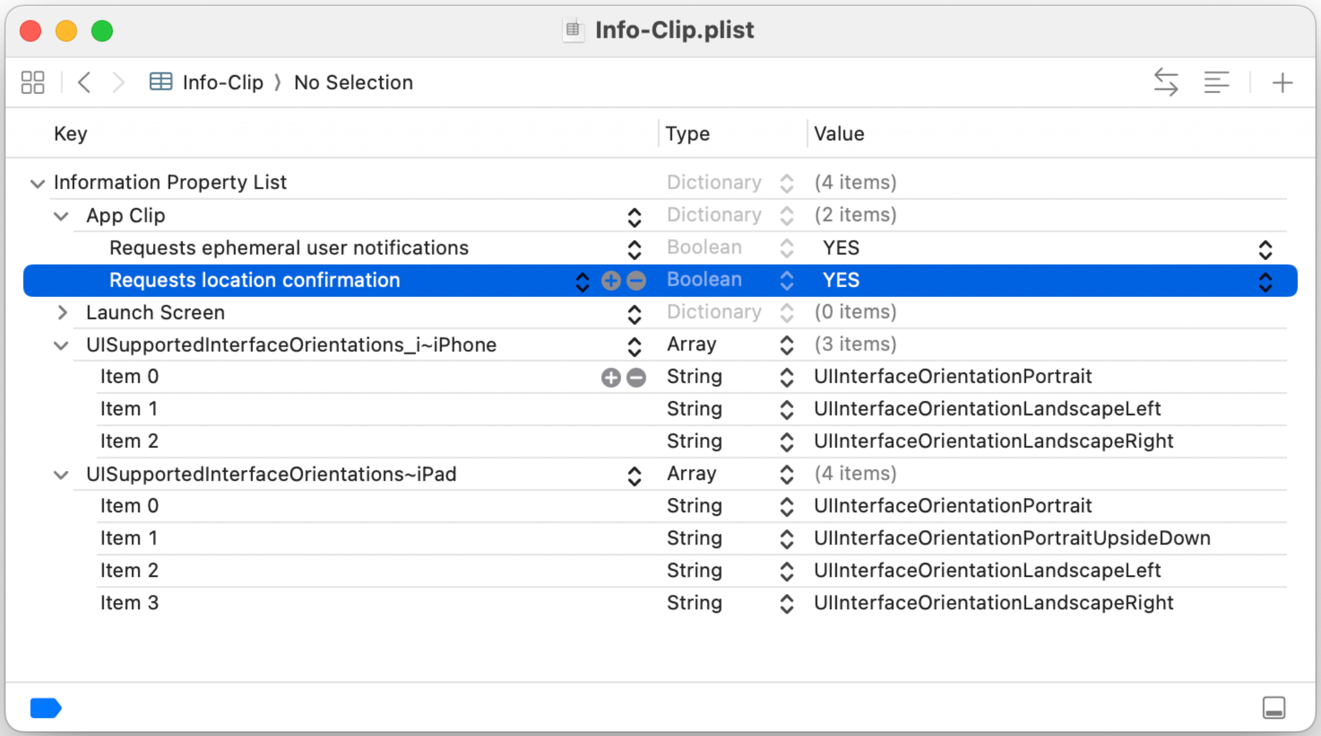
“Food Truck” would like to confrim you’re in the right area and send you notifications. Options…

Powered by Food Truck 4+

Enable your App Clip to verify a person’s location

To enable your App Clip to verify the person’s location, modify your App Clip’s `Info.plist` file:

1. Open your App Clip’s `Info.plist`, add the `NSAppClip` key, and set its type to `Dictionary`.
2. Add an entry to the dictionary with `NSAppClipRequestLocationConfirmation` as the key, select `Boolean` as its type, and set its value to `true`.



Key	Type	Value
Information Property List	Dictionary	(4 items)
App Clip	Dictionary	(2 items)
Requests ephemeral user notifications	Boolean	YES
Requests location confirmation	Boolean	YES
Launch Screen	Dictionary	(0 items)
UISupportedInterfaceOrientations_i~iPhone	Array	(3 items)
Item 0	String	UIInterfaceOrientationPortrait
Item 1	String	UIInterfaceOrientationLandscapeLeft
Item 2	String	UIInterfaceOrientationLandscapeRight
UISupportedInterfaceOrientations~iPad	Array	(4 items)
Item 0	String	UIInterfaceOrientationPortrait
Item 1	String	UIInterfaceOrientationPortraitUpsideDown
Item 2	String	UIInterfaceOrientationLandscapeLeft
Item 3	String	UIInterfaceOrientationLandscapeRight

Alternatively, open the `Info.plist` file in the property list editor and add the entry by selecting App Clip from the list of keys. This adds the `NSAppClip` key and the following entries of type `Boolean` to its dictionary: “Requests ephemeral user notifications” and “Requests location confirmation.” Per default, the value for both entries is NO. Change the value for “Requests location confirmation” to YES.

> Note
>
> Don’t add an entry for the `NSAppClipRequestLocationConfirmation` key to your full app’s `Info.plist` — functionality to confirm a person’s location is only available to App Clips. Instead, modify your full app’s code to request permission to access the location of a device and make use of the Core Location framework. For more information, refer to Getting the current location of a device.

Add code that verifies the physical location

After you modify your App Clip's `Info.plist` file, add code to provide the expected physical location information to the App Clip. To retrieve this information, encode an identifier in the URL that launches the App Clip, and use the identifier to look up the location information for a business in your database. Alternatively, encode the location information in the URL that launches the App Clip.

On launch, access the location information, use it to create a `CLCircularRegion` object with a radius of up to 500 meters, and pass it to the `confirmAcquired(in:completionHandler:)` function.

The following code verifies a person's location when they launch the App Clip. Make sure to update your user interface for each possible result, including the case where a person denies access to location services on their device.

```
import UIKit
import AppClip
import CoreLocation

class SceneDelegate: UIResponder, UIWindowSceneDelegate {

    var window: UIWindow?

    // Call the verifyUserLocation(_:) function in all applicable lifecycle callback

    func verifyUserLocation(_ activity: NSUserActivity?) {

        // Guard against faulty data.
        guard activity != nil else { return }
        guard activity!.activityType == NSUserActivityTypeBrowsingWeb else { return
        guard let payload = activity!.appClipActivationPayload else { return }
        guard let incomingURL = activity?.webpageURL else { return }

        // Create a CLRegion object.
        guard let region = location(from: incomingURL) else {
            // Respond to parsing errors here.
            return
        }

        // Verify that the invocation happened at the expected location.
        payload.confirmAcquired(in: region) { (inRegion, error) in
            guard let confirmationError = error as? APActivationPayloadError else {
                if inRegion {
                    // The location of the NFC tag matches a person's location.
                } else {
                    // The location of the NFC tag doesn't match the records,
                    // for example, if someone moved the NFC tag.
                }
                return
            }

            if confirmationError.code == .doesNotMatch {
                // The scanned URL isn't registered for the App Clip.
            } else {
                // A person denied location access, or the source of the
                // App Clip's invocation isn't an NFC tag or visual code.
            }
        }
    }

    func location(from url:URL) -> CLRegion? {
        let coordinates = CLLocationCoordinate2D(latitude: 37.334722,
                                                 longitude: 122.008889)
        return CLCircularRegion(center: coordinates,
                                radius: 100,
                                identifier: "Apple Park")
    }
}
```

For more information on how you can access the App Clip’s invocation URL, refer to Responding to invocations.

See Also

Launch

Responding to invocations

Add code to respond to invocations and offer a focused launch experience.

Associating your App Clip with your website

Enable the system to verify your App Clip to support invocations from your website and devices running iOS 16.3 or earlier.

Supporting invocations from your website and the Messages app

Display a Smart App Banner and the App Clip card on your website that people tap to launch your App Clip, and add support for invocations from the Messages app.

Launching another app’s App Clip from your app

Enable people to launch another app’s App Clip from your app with App Clip links and offer a rich preview of it with the Link Presentation framework.

`class APActivationPayload`

Information that’s passed to an App Clip on launch.

`NSAppClip`

A collection of keys that an App Clip uses to get additional capabilities.