Article

Requesting access to protected resources

Provide a purpose string that explains to a person why you need access to protected resources on their device.

Overview

Modern devices collect and store a wealth of sensitive information about people who use them. Many apps rely on this kind of data, and the device hardware that generates it, to do useful work. For example, a navigation app needs a person's GPS coordinates to locate the person on a map. But not all apps need access to all data. The same navigation app doesn't need access to a person's health history, camera interface, or Bluetooth peripherals.

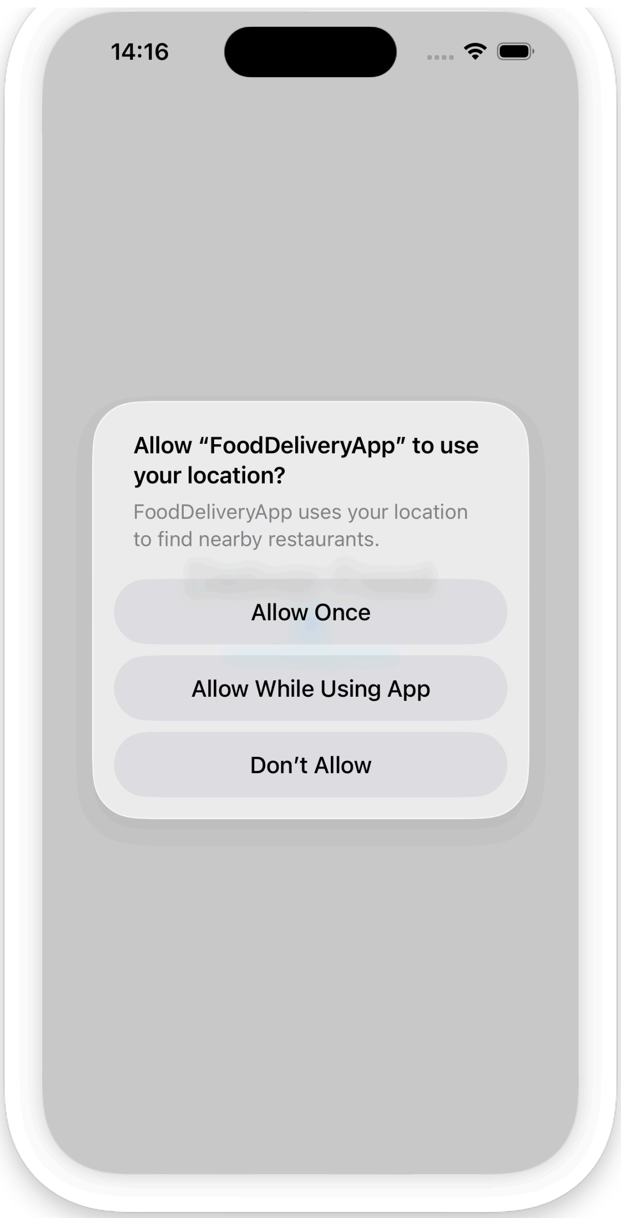
Ensure your app accesses only what it needs to do its job. To support this principle, Apple's operating systems restrict access to protected data and resources by default. Apps can request access on a case-by-case basis, providing an explanation for why they need access. The person who uses the app decides whether to grant or deny the request.

Tip

In addition to asking people for permission to access a resource, in some cases, you also need to separately declare your intent to do so by adding an entitlement to your app, as described in `Entitlements`.

Provide a purpose string

The first time your app attempts to access a protected resource, the system prompts the person using the app for permission. In the following example, an iOS app called FoodDeliveryApp, which provides a food delivery service, generates a prompt requesting access to the person's location:


14:16
Allow "FoodDeliveryApp" to use your location?
FoodDeliveryApp uses your location to find nearby restaurants.
Allow Once
Allow While Using App
Don't Allow

If the person grants permission, the system remembers the person's choice and doesn't prompt again. If the person denies permission, the access attempt that initiates the prompt, and any further attempts, fail in a resource-specific way. For the particular case of access to location data, the person can choose to allow access for one session only by tapping Allow Once.

The system automatically generates the prompt's title, which includes the name of your app. You supply a message called a *purpose string* or a *usage description* — in this case, "Your location allows you to view restaurants in delivery range of your address." — to indicate the reason that your app needs the access. Accurately and concisely explaining to the person why your app needs access to sensitive data, typically in one complete sentence, lets the person make an informed decision and improves the chances that they grant access.

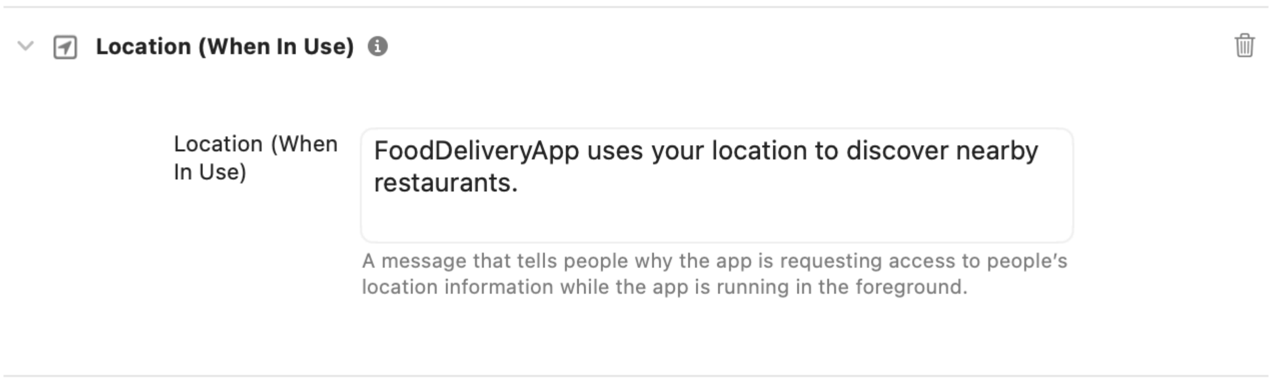
To provide a purpose string, follow these steps in Xcode:

1. Navigate to the Signing and Capabilities editor for your app.

2. Click the Add (+) button to add a capability.

3. Choose the protected resource you want to add; in this case, it's "Location (When in Use)".

4. Enter the purpose string in the text field.

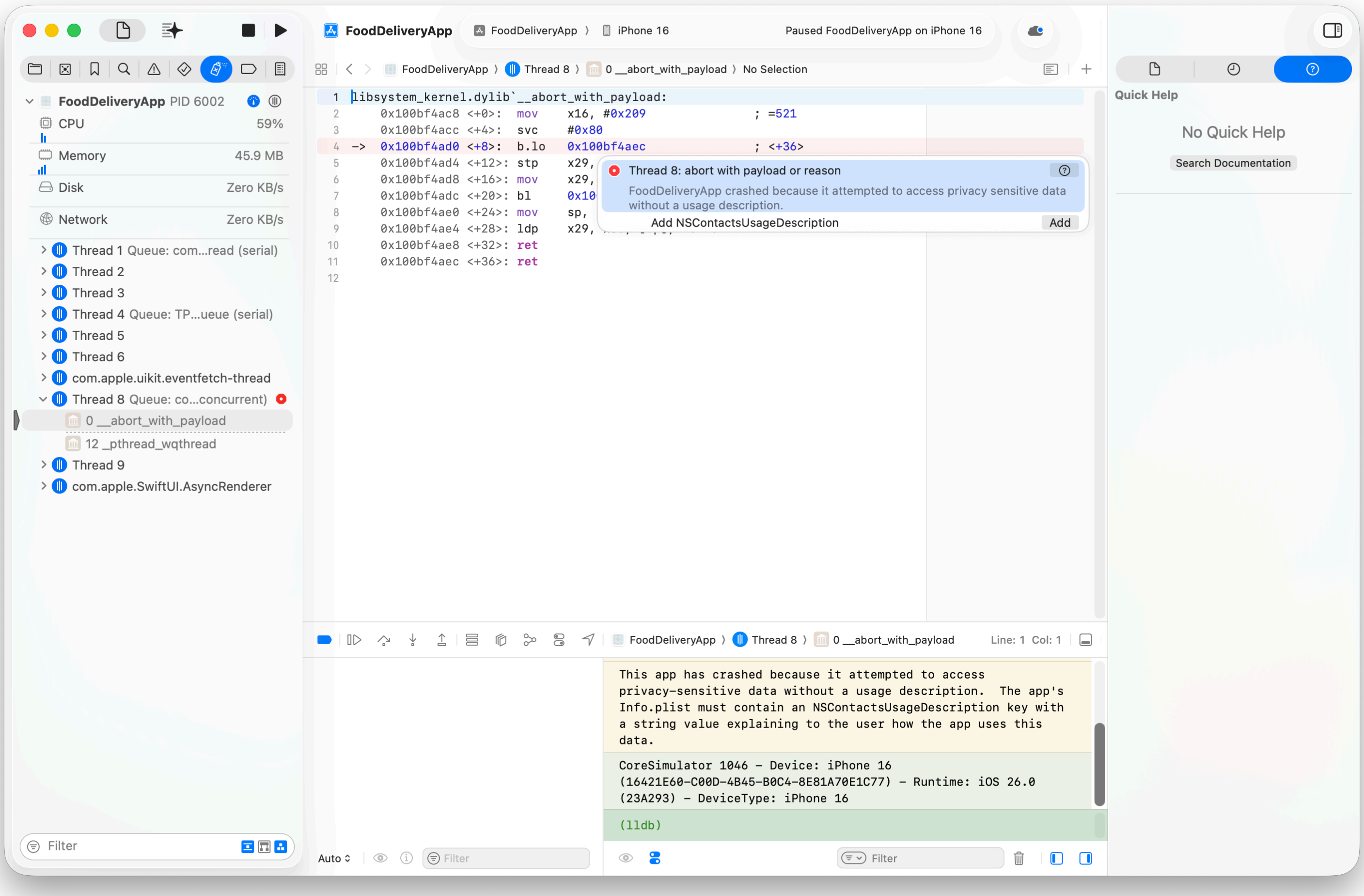


Location (When In Use)

Location (When In Use): FoodDeliveryApp uses your location to discover nearby restaurants.

A message that tells people why the app is requesting access to people's location information while the app is running in the foreground.

Always provide a valid purpose string in the Signing and Capabilities editor if your app uses a protected resource. If you don't, attempts to access the resource fail, and might cause your app to crash. Xcode detects when your app crashes for this reason and reports an issue, telling you to add the purpose string to your app. Click the Add button to provide the purpose string.



```
libsystem_kernel.dylib`__abort_with_payload:
    0x100bf4ac8 <+0>:  mov    x16, #0x209               ; =521
    0x100bf4acc <+4>:  svc    #0x80
->  0x100bf4ad0 <+8>:  b.lo   0x100bf4aec               ; <+36>
    0x100bf4ad4 <+12>: stp    x29,
    0x100bf4ad8 <+16>: mov    x29,
    0x100bf4adc <+20>: bl     0x10
    0x100bf4ae0 <+24>: mov    sp,
    0x100bf4ae4 <+28>: ldp    x29,
    0x100bf4ae8 <+32>: ret
    0x100bf4aec <+36>: ret
```

Thread 8: abort with payload or reason

FoodDeliveryApp crashed because it attempted to access privacy sensitive data without a usage description.

Add NSContactsUsageDescription — Add

```
This app has crashed because it attempted to access
privacy-sensitive data without a usage description.  The app's
Info.plist must contain an NSContactsUsageDescription key with
a string value explaining to the user how the app uses this
data.

CoreSimulator 1046 – Device: iPhone 16
(16421E60-C00D-4B45-B0C4-8E81A70E1C77) – Runtime: iOS 26.0
(23A293) – DeviceType: iPhone 16

(lldb)
```

FoodDeliveryApp
FoodDeliveryApp 〉 iPhone 16
Paused FoodDeliveryApp on iPhone 16
FoodDeliveryApp 〉 Thread 8 〉 0 __abort_with_payload 〉 No Selection
FoodDeliveryApp PID 6002
CPU 59%
Memory 45.9 MB
Disk Zero KB/s
Network Zero KB/s
Thread 1 Queue: com...read (serial)
Thread 2
Thread 3
Thread 4 Queue: TP...ueue (serial)
Thread 5
Thread 6
com.apple.uikit.eventfetch-thread
Thread 8 Queue: co...concurrent)
0 __abort_with_payload
12 _pthread_wqthread
Thread 9
com.apple.SwiftUI.AsyncRenderer
Filter
Quick Help
No Quick Help
Search Documentation
FoodDeliveryApp 〉 Thread 8 〉 0 __abort_with_payload
Line: 1 Col: 1
Auto
Filter
Filter

Xcode adds a build setting to your app that configures the purpose string as the value for a `Information Property List`; in this example, the key is `NSLocationWhenInUseUsageDescription`, so the build setting is `INFOPLIST_KEY_NSLocationWhenInUseUsageDescription`. For more information on configuring information property list values using build settings, see Managing your app's information property list values.

If your app supports multiple locales, in addition to providing a purpose string in the Signing and Capabilities editor, localize the purpose string for each locale you support. Create a string catalog file called `InfoPlist.xcstrings`, and build your app to populate the string catalog with keys for the usage description strings in your app. Add the translations for your usage description strings to the localizations in the string catalog. For more information, see Localizing and varying text with a string catalog.

Adhere to the requirements for purpose strings

To give people useful, concise information about why you're requesting access to protected resources, make sure each purpose string you provide is valid by checking the following:

- The purpose string isn't blank and doesn't consist solely of whitespace characters.
- The purpose string is shorter than 4,000 bytes. Typical purpose strings are one complete sentence, but you can provide additional information to help a person make the right decision about sharing personal information.
- The purpose string has the proper type that the corresponding key requires, typically a string.
- The purpose string provides a description that's accurate, meaningful, and specific about why the app needs to access the protected resource.

Adhere to these requirements for every purpose string in your app, including localized purpose strings.

App Review checks for the use of protected resources, and rejects apps that contain code accessing those resources without a purpose string. For example, an app accessing location might receive the following information from App Review about the requirement that an `NSLocationWhenInUseUsageDescription` key be present:

```
ITMS-90683: Missing purpose string in Info.plist.
Your app's code references one or more APIs that access sensitive user
data, or the app has one or more entitlements that permit such access.
The Info.plist file for the "{app-bundle-path}" bundle should contain a
NSLocationWhenInUseUsageDescription key with a user-facing purpose string
explaining clearly and completely why your app needs the data.
If you're using external libraries or SDKs, they may reference APIs that
require a purpose string. While your app might not use these APIs, a
purpose string is still required. For details, visit:
https://developer.apple.com/documentation/uikit/protecting_the_user_s_privacy/reques
```

To resolve this issue, provide a purpose string that explains why the app needs access to this sensitive information, or remove the code that's accessing the resource.

Note

If you’re using external libraries or SDKs, they may reference APIs that require a purpose string. Although your app might not use these APIs, a purpose string is still necessary for App Review. You can contact the developer of the library or SDK to request information about which protected resources the developer uses and their purpose, or request that the developer release a version of their code that doesn’t contain the APIs. You’re responsible for all access of protected resources, including external SDK and library access.

Check for authorization

Many system frameworks that provide access to protected resources have dedicated APIs for checking and requesting authorization to use those resources. This model allows you to adjust your app’s behavior depending on the current access it has. For example, if a person denies your app permission to do something, you can remove related elements from your user interface.

Because a person can change authorization at any time using Settings, always check the authorization status of a feature before accessing it. In cases without a dedicated API, prepare your app to gracefully handle access failures.

Reset authorization access

When your app attempts to access a protected resource after its first attempt, the system remembers the person’s permission choice and doesn’t prompt again. To prompt the person again, you need to reset access to these resources on your device or system.

To reset permission access to a protected resource in iOS apps, tap Settings > General > Transfer or Reset iPhone > Reset > Reset Location & Privacy on your device.

Important

Using Reset Location & Privacy resets location and privacy settings for all services on your device.

To reset permissions for a particular service in macOS apps, run the `tccutil reset <service name>` command in Terminal. For example, to reset all permissions for AppleEvents, type:

```
$ tccutil reset AppleEvents
```

This command resets authorization access for all apps using the protected resource. You can similarly specify Camera, Calendar, Reminders, or other services to reset them individually.

See Also

Supporting Privacy

Encrypting Your App’s Files

Protect the user’s data in iOS by encrypting it on disk.