


UIKit /  / application(_:handle:completionHandler:) **Deprecated**

Instance Method

application(_:handle:completionHandler:) **Deprecated**

Asks the delegate to handle the specified SiriKit intent directly.

iOS 11.0–14.0 **Deprecated** | iPadOS 11.0–14.0 **Deprecated** | Mac Catalyst 13.1–14.0 **Deprecated** |

tvOS 11.0–14.0 **Deprecated** | visionOS 1.0–1.0 **Deprecated**

```
@MainActor
optional func application(
    _ application: UIApplication,
    handle intent: INIntent,
    completionHandler: @escaping (INIntentResponse) -> Void
)
```

```
@MainActor
optional func application(
    _ application: UIApplication,
    handle intent: INIntent
) async -> INIntentResponse
```

Deprecated

Use `application(_:handlerFor:)` instead to provide an object to resolve, confirm, and handle intents in your app.

Parameters

application

The shared app object.

intent

The intent object that contains information about the SiriKit request. Use this object to identify what the user intends and what kind of response to provide.

completionHandler

The handler block to execute with your response. You must execute this handler at some point during your implementation of this method. This handler has no return value and takes the following parameter:

intentResponse

The response object you create to report the status of the request. The exact type of this object must correspond to the type of intent delivered. For example, if the `intent` parameter contains an `INStartWorkoutIntent` object, you must create an `INStartWorkoutIntentResponse` object. This parameter must not be `nil`.

Discussion

With this method, an app can handle an intent directly, rather than handling it in the app's Intent extension. You might use this method to implement workflows that you can't implement easily in your extension. For example, you might use it to start or manage a user's workout session. If your app isn't running, SiriKit launches your app in the background so that the Siri interface remains active.

Your Intents app extension is still responsible for resolving and confirming the intent details. Your extension's `handler(for:)` method must create a response object that resolves and confirms the intent details. In the response object's `handle(intent:completion:)` implementation, return a response object with a status code of `failureRequiringAppLaunch`. Upon receiving your response, SiriKit launches the app and calls `application(_:handle:completionHandler:)`. In your implementation of this app delegate method, handle the intent by performing the user's intended action if possible. Then call the provided completion handler with a response object that indicates if your app performed the intent or provides a reason it could not. For details about how to handle a specific intent, see the class reference for that intent in [SiriKit](#).

See Also

Deprecated

~~func application(UINavigationController, didRegister: UIUserNotificationSettings) →~~

Called to tell the delegate the types of local and remote notifications that can be used to get the user's attention.

Deprecated

~~func application(UINavigationController, didReceive: UILocalNotification)~~

Sent to the delegate when a running app receives a local notification.

Deprecated

~~func application(UINavigationController, didReceiveRemoteNotification: [Any Hashable : Any])~~

Called when your app has received a remote notification.

Deprecated

~~func application(UINavigationController, handleActionWithIdentifier: String?, for: UILocalNotification, completionHandler: () → Void)~~

Called when your app has been activated because user selected a custom action from the alert panel of a local notification.

Deprecated

~~func application(UINavigationController, handleActionWithIdentifier: String?, for: UILocalNotification, withResponseInfo: [AnyHashable : Any], completionHandler: () → Void)~~

Called when your app has been activated by the user selecting an action from a local notification.

Deprecated

~~func application(UINavigationController, handleActionWithIdentifier: String?, forRemoteNotification: [AnyHashable : Any], completionHandler: () → Void)~~

Tells the app delegate to perform the custom action specified by a remote notification.

Deprecated

~~func application(UINavigationController, handleActionWithIdentifier: String?, forRemoteNotification: [AnyHashable : Any], withResponseInfo: [Any Hashable : Any], completionHandler: () → Void)~~

Called when your app has been activated by the user selecting an action from a remote notification.

Deprecated

~~func application(UIApplication, handleOpen: URL) -> Bool~~

Asks the delegate to open a resource identified by URL.

Deprecated

~~func application(UIApplication, open: URL, sourceApplication: String?,
annotation: Any) -> Bool~~

Asks the delegate to open a resource identified by a URL.

Deprecated

~~func application(UIApplication, willChangeStatusBarOrientation:
UIInterfaceOrientation, duration: TimeInterval)~~

Tells the delegate when the interface orientation of the status bar is about to change.

Deprecated

~~func application(UIApplication, didChangeStatusBarOrientation:
UIInterfaceOrientation)~~

Tells the delegate when the interface orientation of the status bar has changed.

Deprecated

~~func application(UIApplication, willChangeStatusBarFrame: CGRect)~~

Tells the delegate when the frame of the status bar is about to change.

Deprecated

~~func application(UIApplication, didChangeStatusBarFrame: CGRect)~~

Tells the delegate when the frame of the status bar has changed.

Deprecated

~~func application(UIApplication, performFetchWithCompletionHandler: (
UIBackgroundFetchResult) -> Void)~~

Tells the app that it can begin a fetch operation if it has data to download.

Deprecated

~~func application(UIApplication, shouldSaveApplicationState: NSCoder) ->
Bool~~

Asks the delegate whether to preserve the app's state.

Deprecated