Instance Method

application(_:continue:restorationHandler:) Deprecated

Tells the delegate that the data for continuing an activity is available.

iOS 8.0–26.0 Deprecated | iPadOS 8.0–26.0 Deprecated | Mac Catalyst 13.1–26.0 Deprecated | tvOS 9.0–26.0 Deprecated | visionOS 1.0–26.0 Deprecated

```
@MainActor
optional func application(
    _ application: UIApplication,
    continue userActivity: NSUserActivity,
    restorationHandler: @escaping ([any UIUserActivityRestoring]?) -> Void
) -> Bool
```

Deprecated

Use UIScene lifecycle and scene(_:continue:) from UISceneDelegate instead.

Parameters

`application`
The shared app object that controls and coordinates your app.

`userActivity`
The activity object containing the data associated with the task the user was performing. Use the data to continue the user's activity in your iOS app.

`restorationHandler`

A block to execute if your app creates objects to perform the task the user was performing. Calling this block is optional and you can copy this block and call it at a later time. When calling a saved copy of the block, you must call it from the app's main thread. This block has no return value and takes the following parameter:

restorableObjects

An array of objects that conform to `UIUserActivityRestoring` that represent the objects you created or fetched in order to perform the operation. The system calls the `restoreUserActivityState(_:)` method of each object in the array to give it a chance to perform the operation.

Return Value

`true` to indicate that your app handled the activity or `false` to let iOS know that your app didn't handle the activity.

Discussion

The app calls this method when it receives data associated with a user activity, for example, when the user transfers an activity from a different device using Handoff.

Implement this method to update your iOS app so that the user can continue the activity from where they left off. If you don't implement this method or if your implementation returns `false`, iOS tries to create a document for your app to open using a URL.

Calling the block in the `restorationHandler` is optional and only needed when specific objects are capable of continuing the activity.

This method isn't called if either `application(_:willFinishLaunchingWithOptions:)` or `application(_:didFinishLaunchingWithOptions:)` returns `false`.

Handling Activities from SiriKit

This method is called whenever your app is launched to handle a SiriKit intent. Update your app's user interface based on the `userActivity` parameter. Your app should seamlessly continue the interaction that began in Siri.

By default, the intent provides an `NSUserActivity` object whose `interaction` property contains both the originating intent and your response. You can add additional, app-specific information by creating a new `NSUserActivity` object in your intent's `confirm` or `handle` method and adding your data to the activity's `userInfo` dictionary.

When continuing activities from SiriKit:

- Look for the intent specified in the `interaction` property. Resume handling this intent in your app.
- Avoid accidentally repeating actions (such as making double payments). For example, check the `INInteraction` object's `intentResponse` property to see if the action has already been completed.

Intents may launch your app under the following circumstances:

- Some intents always launch the app after the intent is successfully handled (for example, intents with a `continueInApp` response code).
- Your intent's `handle` and `confirm` methods launch the app when you resolve the intent with a `failureRequiringAppLaunch` (or similar) response code.
- The user can always decide to launch the app at any point in a Siri transaction.

See Also

Continuing user activity and handling quick actions

~~`func application(UIApplication, willContinueUserActivityWithType: String) -> Bool`~~

Tells the delegate if your app takes responsibility for notifying users when a continuation activity takes longer than expected.

Deprecated

~~`func application(UIApplication, didUpdate: NSUserActivity)`~~

Tells the delegate that the activity was updated.

Deprecated

~~`func application(UIApplication, didFailToContinueUserActivityWithType: String, error: any Error)`~~

Tells the delegate that the activity couldn't be continued.

Deprecated

~~`func application(UIApplication, performActionFor: UIApplicationShortcutItem, completionHandler: (Bool) -> Void)`~~

Tells the delegate that the user selected a Home screen quick action for your app, except when you've intercepted the interaction in a launch method.