Class

NSUserActivity

A representation of the state of your app at a moment in time.

iOS 8.0+ | iPadOS 8.0+ | Mac Catalyst 13.1+ | macOS 10.10+ | tvOS 9.0+ | visionOS 1.0+ | watchOS 2.0+

```
class NSUserActivity
```

Mentioned in

- Implementing Handoff in Your App
- Creating a user activity object

Overview

An `NSUserActivity` object provides a lightweight way to capture the state of your app and put it to use later. Create this object to capture information about what a person was doing, such as viewing app content, editing a document, viewing a web page, or watching a video. When the system launches your app and an activity object is available, your app can use the information in that object to restore itself to an appropriate state. Spotlight also uses these objects to improve search results for people. To allow people to continue an activity on another device, see Implementing Handoff in Your App.

Siri

If SiriKit needs to launch your app for any reason, it creates a user activity object and assigns an appropriate doc://com.apple.documentation/documentation/sirikit/ininteraction object to its `interaction` property. Your app can use the interaction information to configure itself and

display information related to the interaction started by SiriKit. You can also provide SiriKit with a custom user activity object containing additional data that you want passed to your app.

In iOS 15 and later, a person can share content they're viewing by asking Siri to "share this". Apps built with Mac Catalyst provide the same capability with an `NSSharingServicePickerToolbarItem` in the toolbar. You can use `activityItemsConfiguration` or `activityItemsConfigurationSource` to provide shareable content. In iOS, if both of those properties are `nil`, Siri uses the `webpageURL` property of your app's current user activity as a fallback value.

Quick Note

Quick Note on macOS and iOS can link to any app content represented as an `NSUserActivity`. To appear as a link, the content must be the app's current activity, and provide at least one of the following identifiers:

`webpageURL`
: An `https:` URL, ideally in a canonical form that's consistent every time a person visits the same content.

`persistentIdentifier`
: A string that uniquely identifies the content in this domain. The identifier should identify the same content across devices.

`targetContentIdentifier`
: A string that uniquely identifies the content in this domain, but also allows disambiguating between multiple scenes of an app. The identifier should identify the same content across devices.

To work well with Quick Note, content must adhere to the following guidelines:

- The activity `title` should be clear and concise. This text describes the content of the link, like "Photo taken on July 27, 2020" or "Conversation with Maria". Use nouns for activity titles.
- Keep the app's current activity up to date, using `becomeCurrent()` and `resignCurrent()`.
- Linkable identifiers (listed above) must be stable and consistent for the same content. When you link from a note to a document in an app, and later revisit that document, the system shows an indicator linking back to the note. The system compares identifiers to check that the document is the same as the original source of the link.
- Maintain support for activities provided by your app, and support navigating to linked content indefinitely. Links added to notes are important to people, who may feel that a broken link indicates data loss.
- Gracefully handle attempts to navigate to an activity that points to content that doesn't exist. For example, you can redirect to the new location of moved content, or show an error message.

This situation may happen with shared notes, when a person links to content that exists only on another person's device.

Search results

If your `NSUserActivity` objects contain information that a person might want to search for later, set the `isEligibleForSearch` property to `true`. When you enable search, Spotlight indexes your user activity objects and considers them during subsequent on-device searches. For example, if a person viewed information about a particular restaurant in your app, you'd enable search for the corresponding user activity object. Subsequent searches for restaurants using Spotlight could then include the results obtained from your user activity object.

In addition to on-device searches, you can contribute URLs accessed by your app with the global Spotlight search engine. Sharing a URL helps Spotlight improve its own search results for other people. To contribute a URL, put the URL in the `webpageURL` property of your activity object and set the `isEligibleForPublicIndexing` property to `true`.

Important

Your app must maintain a strong reference to any activity objects that you use for search results.

Employ user activity objects to record user-initiated activities, not as a general-purpose indexing mechanism of your app's data. To index all of your app's content, and not just the content touched by people, use the APIs of the Core Spotlight framework.

Topics

Creating a user activity object

Creating a user activity object

Identify key user interactions and include the information to restore them later.

`init(activityType: String)`

Creates a user activity object with the specified type.

~~`convenience init()`~~

Creates a user activity object using the first activity type declared in the app's information property list file.

Deprecated

Accessing activity information

`var activityType: String`

The user activity object's activity type.

`var title: String?`

An optional, user-visible title for this activity, such as a document name or web page title.

`var requiredUserInfoKeys: Set<String>?`

A set of keys that represent the minimal information about the activity that should be stored for later restoration.

`var userInfo: [AnyHashable : Any]?`

A dictionary containing app-specific state information needed to continue an activity on another device.

`func addUserInfoEntries(from: [AnyHashable : Any])`

Adds the contents of the specified dictionary to the user info dictionary.

`var targetContentIdentifier: String?`

A string that identifies the user activity's content.

`var needsSave: Bool`

A Boolean value that indicates whether the state of the activity needs to be updated.

`var contentAttributeSet: CSSearchableItemAttributeSet?`

A set of properties that describe the activity.

`var keywords: Set<String>`

A set of localized keywords that can help users find the activity in search results.

`var persistentIdentifier: NSUserActivityPersistentIdentifier?`

A value used to identify the user activity.

`typealias NSUserActivityPersistentIdentifier`

The type that defines a persistent identifier value for a user activity.

`var appClipActivationPayload: APActivationPayload?`

An object containing the payload information that launches an App Clip.

Specifying the activity's eligibility

```
var isEligibleForHandoff: Bool
```

A Boolean value that indicates whether the activity can be continued on another device using Handoff.

```
var isEligibleForSearch: Bool
```

A Boolean value that indicates whether the activity should be added to the on-device index.

```
var isEligibleForPublicIndexing: Bool
```

A Boolean value that indicates whether the activity can be publicly accessed by all iOS users.

```
var expirationDate: Date?
```

The date after which the activity is no longer eligible for Handoff or indexing.

Registering and invalidating user activities

```
func becomeCurrent()
```

Marks the activity as currently in use by the user.

```
func resignCurrent()
```

Marks this activity object as inactive without invalidating it.

```
func invalidate()
```

Invalidates an activity and marks it as no longer eligible for continuation.

Deleting saved user activities

```
class func deleteAllSavedUserActivities(completionHandler: () -> Void)
```

Deletes all user activities created by your app.

```
class func deleteSavedUserActivities(withPersistentIdentifiers: [NSUser
ActivityPersistentIdentifier], completionHandler: () -> Void)
```

Deletes user activities created by your app that have the specified persistent identifiers.

Accessing the delegate

```
var delegate: (any NSUserActivityDelegate)?
```

The user activity object's delegate.

`protocol NSUserActivityDelegate`

The interface through which a user activity instance notifies its delegate of updates.

Working with continuation streams

`var supportsContinuationStreams: Bool`

A Boolean value that determines whether the continuing app can request streams to be opened back to the originating app.

`func getContinuationStreams(completionHandler: (InputStream?, OutputStream?, (any Error)?) -> Void)`

Requests streams back to the originating app.

Continuing web browsing

`var webpageURL: URL?`

The URL of the webpage to load in a browser to continue the activity.

`var referrerURL: URL?`

The URL of the webpage that linked to the webpage URL.

Donating to Siri Shortcuts

`var isEligibleForPrediction: Bool`

A Boolean value that determines whether Siri can suggest the user activity as a shortcut to the user.

`var suggestedInvocationPhrase: String?`

A phrase suggested to the user when they create a shortcut.

`var shortcutAvailability: INShortcutAvailabilityOptions`

A set of defined contexts in which an intent or activity might be relevant to a user.

Continuing Siri interactions

`var interaction: INInteraction?`

The SiriKit interaction object to use when configuring your app.

Retrieving NFC tag data

`var ndefMessagePayload: NFCNDEFMessage`

The NDEF message read by the system in the background.

Processing barcodes

`var detectedBarcodeDescriptor: CIBarcodeDescriptor?`

The barcode that the system scanner passes in.

Sharing map item information

`var mapItem: MKMapItem!`

Attaches the specified map item to a user activity object.

Working with media

`var externalMediaContentIdentifier: String?`

A unique identifier from the app's media content catalog for the currently displayed media item.

Managing type-safe access to user info

`func setTypedPayload<T>(T) throws`

Encodes the specified payload into the user activity's user info dictionary.

`func typedPayload<T>(T.Type) throws -> T`

Decodes the user activity's user info dictionary as an instance of the specified type.

`enum TypedPayloadError`

An enumeration that describes the error types for getting and setting a typed payload.

Working with ClassKit

`var isClassKitDeepLink: Bool`

A Boolean value that indicates whether a user activity represents a ClassKit context.

`var contextIdentifierPath: [String]?`

The identifier path associated with a user activity generated by an app that adopts ClassKit.

Identifying activity types

`let NSUserActivityTypeBrowsingWeb: String`

An activity that continues from Handoff or a universal link.

`let TVUserActivityTypeBrowsingChannelGuide: String`

An activity for viewing your app's channel guide.

Reporting errors

`var NSUserActivityConnectionUnavailableError: Int`

The user activity couldn't be continued because a required connection wasn't available.

`var NSUserActivityErrorMaximum: Int`

The end of the range of error codes reserved for user activity errors.

`var NSUserActivityErrorMinimum: Int`

The start of the range of error codes reserved for user activity errors.

`var NSUserActivityHandoffFailedError: Int`

The data for the user activity wasn't available.

`var NSUserActivityHandoffUserInfoTooLargeError: Int`

The user info dictionary was too large to receive.

`var NSUserActivityRemoteApplicationTimedOutError: Int`

The remote application failed to send data within the specified time.

Instance Properties

`var appEntityIdentifier: EntityIdentifier?`

The identifier of an app entity that you associate with the user activity.

Instance Methods

`func widgetConfigurationIntent<Intent>(of: Intent.Type) -> Intent?`

Default Implementations

Relationships

Inherits From

`NSObject`

Conforms To

`AppEntityAnnotatable`
`CVarArg`
`Copyable`
`CustomDebugStringConvertible`
`CustomStringConvertible`
`Equatable`
`Hashable`
`NSItemProviderReading`
`NSItemProviderWriting`
`NSObjectProtocol`

See Also

Host App Interaction

`protocol` `NSUserActivityDelegate`

The interface through which a user activity instance notifies its delegate of updates.