API Collection

Making onscreen content available to Siri and Apple Intelligence

Enable Siri and Apple Intelligence to respond to a person's questions and action requests for your app's onscreen content.

iOS 18.2+ | iPadOS 18.2+ | Mac Catalyst 18.2+ | macOS 15.2+ | tvOS 18.2+ | visionOS 2.2+ | watchOS 11.2+ | Xcode 16.2+

Overview

When a person asks a question about onscreen content or wants to perform an action on it, Siri and Apple Intelligence will be able to retrieve the content to respond to the question and perform the action. If the user explicitly requests it, Siri and Apple Intelligence will be able to send content to supported third-party services. For example, someone could view a website and use Siri to provide a summary by saying or typing a phrase like "Hey Siri, what's this document about?"

> **Note**
>
> Siri's personal context understanding, onscreen awareness, and in-app actions are in development and will be available with a future software update.

Create an app entity and associate it with the user activity object

To integrate your app's onscreen content with current and upcoming personal intelligence features of Siri and Apple Intelligence, explicitly provide the onscreen content using the App Intents framework. Describe the content with an `AppEntity` — you might be able to reuse existing app entity code. Then, tell the system about the content when it becomes visible:

1. Create an app entity identifier using `EntityIdentifier` .
2. Associate the identifier with the current `NSUserActivity` by setting the activity's `appEntityIdentifier` property.

To remove the association between the user activity and your app entity, set the user activity's `appEntityIdentifier` property to `nil`.

The following code snippet from the Making your app's functionality available to Siri sample code project shows how a photo-viewing app might provide a photo to Siri and Apple Intelligence by creating an app entity identifier for the `asset` app entity that represents a photo, and associating it with the user activity:

```
MediaView(
    image: image,
    duration: asset.duration,
    isFavorite: asset.isFavorite,
    proxy: proxy
)
.userActivity(
    "com.example.apple-samplecode.AssistantSchemasExample.ViewingPhoto",
    element: asset.entity
) { asset, activity in
    activity.title = "Viewing a photo"
    activity.appEntityIdentifier = EntityIdentifier(for: asset)
}
```

Make the app entity transferable

Associating an `AppEntity` with the `NSUserActivity` provides Siri and Apple Intelligence with your app's onscreen content to offer personalized intelligence assistance. To go one step further and enable Siri and Apple Intelligence to further process the provided onscreen content and respond to a person's explicit request to send the content as an attachment to other services, including third parties:

1. Update your app entity to conform to the `Transferable` protocol.
2. In your `Transferable` implementation, provide image, PDF, rich text, or plain text representations. To increase compatibility with third-party services, provide several representations that best fit your content. For example, an email client might represent an email as rich text, plain text, and a PDF. For more on adopting `Transferable`, refer to CoreTransferable.

Provide additional context to the system with an assistant schema

To enable Siri and Apple Intelligence to further process the provided onscreen content and provide a better response in iOS 18, make sure that the app entity that you associate with an `NSUserActivity` conforms to one of the assistant schemas in the list below.

> **Note**
>
> The listed requests below are examples and not exhaustive. Actual functionality depends on factors such as the features provided by Siri and Apple Intelligence, the functionality offered by third-party services, or the phrase a person uses.

Domain	Schema	Swift macro	Example request
Browser	`tab`	`@AppEntity(schema: .browser.tab)`	A person might ask Siri questions about the web page.
Document reader	`document`	`@AppEntity(schema: .reader.document)`	A person might ask Siri to explain the conclusion of a document.
File management	`file`	`@AppEntity(schema: .files.file)`	A person might ask Siri to summarize file content.
Mail	`message`	`@AppEntity(schema: .mail.message)`	A person might ask Siri to provide a summary.
Photos	`asset`	`@AppEntity(schema: .photos.asset)`	A person might ask Siri about things to do with an object in a photo.
Presentations	`document`	`@AppEntity(schema: .presentation .document)`	A person might ask Siri to suggest a creative title for a presentation.
Spreadsheets	`document`	`@AppEntity(schema: .spreadsheet.document)`	A person might ask Siri to give an overview of the spreadsheet's data.

Domain	Schema	Swift macro	Example request
Word processor	document	`@AppEntity(schema: .wordProcessor .document)`	A person might ask Siri to suggest additional content for a text document.

Topics

System protocols

`protocol` `AppEntityAnnotatable`

A protocol that framework types adopt to enable you to provide content to system experiences.

See Also

Siri and Apple Intelligence

Integrating actions with Siri and Apple Intelligence

Create app intents, entities, and enumerations that conform to assistant schemas to tap into the enhanced action capabilities of Siri and Apple Intelligence.

App intent domains

Make your app's actions and content available to Siri and Apple Intelligence with assistant schemas.

{} Making your app's functionality available to Siri

Add app intent schemas to your app so Siri can complete requests, and integrate your app with Apple Intelligence, Spotlight, and other system experiences.