All Technologies

App Intents
Essentials
App Intents updates
Making actions and content discoverable ...
Creating your first app intent
Adopting App Intents to support system e...
Accelerating app interactions with App Int...
Siri and Apple Intelligence
Integrating actions with Siri and Apple Int...
Making onscreen content available to Siri ...
App intent domains
Making your app's functionality availabl...
Visual intelligence
Integrating your app with visual intelligence
Visual Intelligence
IntentValueQuery
Interactive Snippets
Displaying static and interactive snippets
SnippetIntent
Other system experiences

Filter /

App Intents / Making your app's functionality available to Siri

Sample Code

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.

Download

iOS 26.0+ | iPadOS 26.0+ | macOS 26.0+ | Xcode 26.0+

Overview

Using this sample app, people can keep track of photos and videos they capture with their device and can use Siri to access app functionality. To make its main functionality available to Siri, the app uses the App Intents framework.

Related sessions from WWDC24

Session 10133: Bring your app to Siri

Make app functionality available to Siri

This sample uses App intent domains to make the `AppEnum`, `AppEntity`, and `AppIntent` implementations available to Siri as shown in the following example:

```
@AppEnum(schema: .photos.assetType)
enum AssetType: String, AppEnum {
    case photo
    case video

    static let caseDisplayRepresentations: [AssetType : DisplayRepresentation]  = [
        .photo: "Photo",
        .video: "Video"
    ]
}
```

Make data available in Spotlight

People can use Spotlight to search for data the sample contains. To enable this functionality, the sample defines an app entity that conforms to `IndexedEntity`:

```
@AppEntity(schema: .photos.asset)
struct AssetEntity: IndexedEntity {

    // MARK: Static

    static let defaultQuery = AssetQuery()

    // MARK: Properties

    let id: String
    let asset: Asset

    @Property(title: "Title")
    var title: String?

    var creationDate: Date?
    var location: CLPlacemark?
    var assetType: AssetType?
    var isFavorite: Bool
    var isHidden: Bool
    var hasSuggestedEdits: Bool

    var displayRepresentation: DisplayRepresentation {
        DisplayRepresentation(
            title: title.map { "\($0)" } ?? "Unknown",
            subtitle: assetType?.localizedStringResource ?? "Photo"
        )
    }
}
```

Make app entities shareable

By adopting the `Transferable` protocol, this sample makes the data it describes as app entities more shareable and allows other apps to understand its data formats. For example, the sample's `AssetEntity` implements `Transferable` to make it easy to share a photo as a PNG image with Siri or the share sheet:

```
extension AssetEntity: Transferable {

    struct AssetQuery: EntityQuery {
        @Dependency
        var library: MediaLibrary

        @MainActor
        func entities(for identifiers: [AssetEntity.ID]) async throws -> [AssetEntity] {
            library.assets(for: identifiers).map(\.entity)
        }

        @MainActor
        func suggestedEntities() async throws -> [AssetEntity] {
            // Suggest the first three assets in the photo library.
            library.assets.prefix(3).map(\.entity)
        }
    }

    static var transferRepresentation: some TransferRepresentation {
        DataRepresentation(exportedContentType: .png) { entity in
            try await entity.asset.pngData()
        }
    }
}
```

Make onscreen content available to Siri and Apple Intelligence

When the user asks a question about onscreen content or wants to perform an action on it, Siri and Apple Intelligence can retrieve the content to respond to the question and perform the action. If the user explicitly requests it, Siri and Apple Intelligence can send content to supported third-party services. For example, someone could view a photo and use Siri to describe things a person can do with an identified object in the photo by saying or typing a phrase like "Hey Siri, what can I do with the object in this photo?" To integrate onscreen content with current and upcoming personal intelligence features of Siri and Apple Intelligence, the sample's `AssetEntity` conforms to the `Transferable` protocol and the `.photos.asset` schema. When a person views a photo, the app associates the asset entity with an `NSUserActivity` to make the photo available to Siri and Apple Intelligence:

```
var body: some View {
    // ...
    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)
    }
    // ...
```

For more information about making onscreen content available to Siri and Apple Intelligence, refer to Making onscreen content available to Siri and Apple Intelligence.

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.

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.

App intent domains

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