Sample Code

Integrating CarPlay with Your Music App

Configure your music app to work with CarPlay by displaying a custom UI.

Download

iOS 14.0+ | Xcode 15.0+

Overview

CarPlay Music is a sample music app that demonstrates how to display a custom UI from a CarPlay–enabled vehicle. CarPlay Music integrates with the CarPlay framework by implementing the `CPNowPlayingTemplate` and `CPListTemplate`. This sample's iOS app component provides a logging interface to help you understand the life cycle of a CarPlay app, as well as a music controller.

Configure the Sample Code Project

To configure the sample code project, perform the following:

1. Complete the steps in Requesting CarPlay Entitlements to request the CarPlay audio entitlement and configure the Xcode project.
2. Create a key and developer token for the MusicKit service. For more information, see Getting Keys and Creating Tokens.
3. Update the `developerToken` variable in the `AppleMusicAPIController.swift` file to use the developer token you create.

Handle Communication with CarPlay

The app is responsible for adding and removing the root view controller of the CarPlay window in response to connections and disconnections.

The following code shows an example implementation of setting a root template:

```
var tabTemplates = [CPTemplate]()

if let playlists = MediaPlayerUtilities.searchForPlaylistsInLocalLibrary(withPredica

    let listItems = playlists.compactMap { (playlist) -> CPListItem? in
        let listItem = CPListItem(text: playlist.name, detailText: "")
        listItem.handler = { playlistItem, completion in
            AppleMusicAPIController.playWithItems(items: playlist.items.compactMap({
                return item.playbackStoreID
            }))
            completion()
        }
        return listItem
    }

    var playlistTemplate: CPListTemplate!

    if #available(iOS 15.0, *) {
        let configuration = CPAssistantCellConfiguration(
                                position: .top,
                                visibility: .always,
                                assistantAction: .playMedia)
        playlistTemplate = CPListTemplate(
                                title: "Playlists",
                                sections: [CPListSection(items: listItems)],
                                assistantCellConfiguration: configuration)

    } else {
        playlistTemplate = CPListTemplate(
                                title: "Playlists",
                                sections: [CPListSection(items: listItems)])

    }

    playlistTemplate.tabImage = UIImage(systemName: "list.star")

    tabTemplates.append(playlistTemplate)
}
```

```
tabTemplates.append(genresTemplate())
tabTemplates.append(settingsTemplate())

self.carplayInterfaceController!.delegate = self
self.carplayInterfaceController!.setRootTemplate(CPTabBarTemplate(templates: tabTemp
```

Prepare for App Selection

Like other Music apps, CarPlay Music is eligible to participate in App Selection to improve its interactions with Siri. This allows the system to automatically select the app for playing music on the device. See `Improving Siri Media Interaction and App Selection`.

The following code demonstrates how to declare an app as eligible for App Selection:

```
let context = INMediaUserContext()
context.numberOfLibraryItems = MPMediaQuery.songs().items?.count
AppleMusicAPIController.sharedController.prepareForRequests { (success) in
    if success {
        context.subscriptionStatus = .subscribed
    } else {
        context.subscriptionStatus = .notSubscribed
    }
    context.becomeCurrent()
}
```

Listen for Changes with the Music Player

A good way to ensure an app UI updates automatically in response to changes is to listen for changes in the Now Playing item, as well as in the playing state. CarPlay Music uses the `applicationMusicPlayer`, so it subscribes to the `MPMusicPlayerControllerPlaybackStateDidChange` and `MPMusicPlayerControllerNowPlayingItemDidChange` notifications.

```
self.playbackObserver = NotificationCenter.default.addObserver(
    forName: .MPMusicPlayerControllerPlaybackStateDidChange,
    object: nil,
    queue: .main) {
    notification in
    MemoryLogger.shared.appendEvent(
        "MPMusicPlayerControllerPlaybackStateDidChange: \(MPMusicPlayerController.ap
}

self.nowPlayingItemObserver = NotificationCenter.default.addObserver(
    forName: .MPMusicPlayerControllerNowPlayingItemDidChange,
    object: nil,
    queue: .main) {
    notification in
    MemoryLogger.shared.appendEvent("MPMusicPlayerControllerNowPlayingItemDidChange'
}
```

See Also

Audio

`class CPNowPlayingTemplate`

A shared system template that displays Now Playing information.