CarPlay Audio App Programming Guide

March 2017

Contents

Introduction

CarPlay is a smarter, safer way to use your iPhone in the car. CarPlay takes the things you want to do with your iPhone while driving and puts them right on your car's built-in display. In addition to getting directions, making calls, and sending and receiving messages, CarPlay supports audio apps, messaging apps and automaker apps that you’ve downloaded to your iPhone.

This guide describes how to create a CarPlay audio app.

CarPlay audio app development requires iOS 7.1 or later and Xcode 5.1 or later.

CarPlay Audio Apps

Users download CarPlay audio apps from the App Store and use them on iPhone like any other app. When an iPhone with your audio app is connected to a CarPlay vehicle compatible with your app, your app's icon appears on the CarPlay home screen.

CarPlay audio apps are designed to look and feel like the built-in Music app. CarPlay audio apps work by providing metadata to populate the provided user interface, and by responding to user actions such as track selection or media control.

iOS manages the display of your app on the CarPlay screen and handles the interface with the car. Your app does not have to worry about screen resolutions, or input hardware such as touchscreens, knobs, or steering wheel controls.

Audio apps must meet the basic requirements defined in *CarPlay APIs Addendum to the iOS Developer Program License Agreement*.

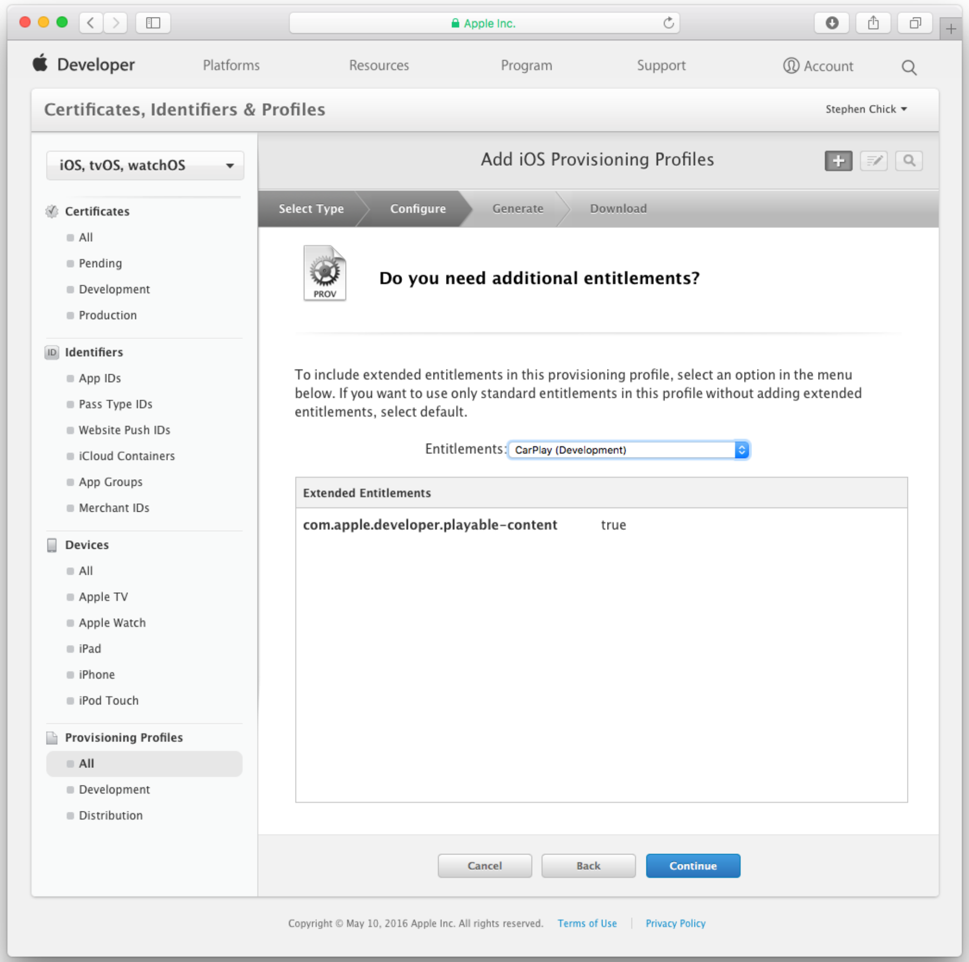
Also see *Human Interface Guidelines for CarPlay Apps*.

Development Environment

CarPlay audio app entitlement

Audio apps require a CarPlay audio app entitlement. To obtain the entitlement, sign and return the *CarPlay APIs Addendum to the iOS Developer Program License Agreement* and let Apple know the Apple Developer Program Team ID that you intend to use when publishing your app on the App Store. Apple will create a CarPlay entitlement and contact you.

Once the entitlement is created, log in to your account on the Apple Developer website and create a new Provisioning Profile that includes the CarPlay audio app entitlement.



Do you need additional entitlements?

To include extended entitlements in this provisioning profile, select an option in the menu below. If you want to use only standard entitlements in this profile without adding extended entitlements, select default.

Entitlements: CarPlay (Development)

Extended Entitlements	
com.apple.developer.playable-content	true

Copyright © May 10, 2016 Apple Inc. All rights reserved. Terms of Use | Privacy Policy

After you have created a new Provisioning Profile that supports CarPlay, be sure to import it into Xcode. The Simulator will not recognize your CarPlay audio app without the presence of a Provisioning Profile that supports CarPlay.

In your Xcode project, you will also need to create an Entitlements.plist file in your project, if you don't have one already, and include a key for the CarPlay audio app entitlement as a Boolean:

```
<key>com.apple.developer.playable-content</key>
<true/>
```

In Xcode, make sure that your target project setting CODE_SIGN_ENTITLEMENTS is set to the path of your Entitlements.plist file.

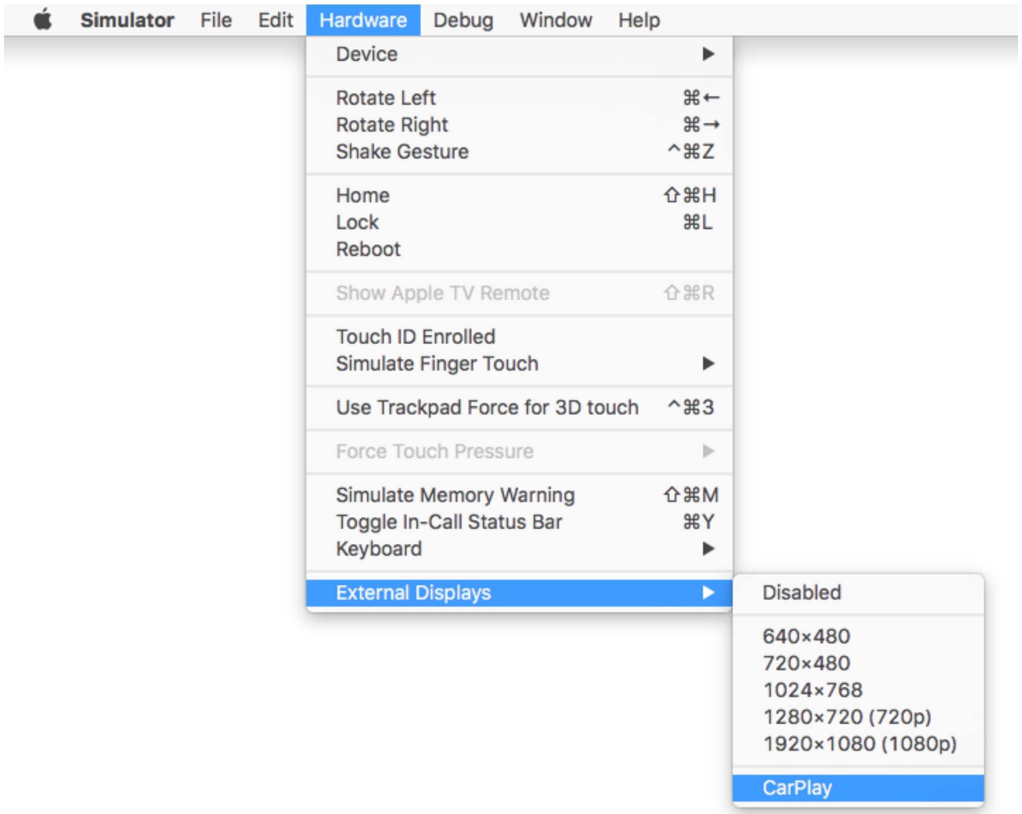
Simulator

To activate CarPlay support in the Simulator, type:

```
defaults write com.apple.iphonesimulator CarPlay -bool YES
```

from the Terminal, then quit and launch the Simulator.

From the Simulator menu, select Hardware, External Displays and CarPlay to show the CarPlay screen. The CarPlay screen will be blank until your app is run.



While the Simulator is useful during development, certain CarPlay features are not available in the Simulator and you should not rely on it as the sole method to develop your app. Test your app on real hardware to observe interaction with other CarPlay features such as the built-in Music app, interruptions from phone calls, and audio mixing with turn-by-turn spoken directions.

It is highly recommended to develop audio apps using a car or aftermarket head unit that supports wireless CarPlay. This will allow you to run CarPlay wirelessly while iPhone is simultaneously connected to Xcode on your Mac using a Lightning to USB cable.

CarPlay Audio App Design

Complete the following tasks to add CarPlay support to your audio app:

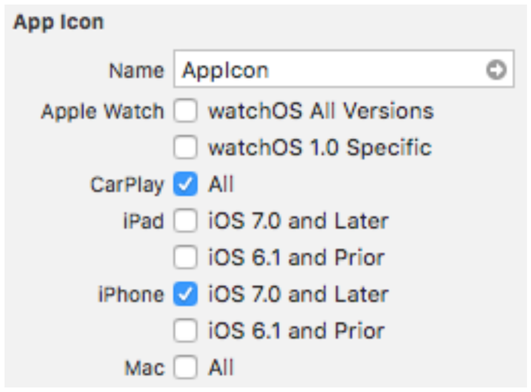
Task	
Include the CarPlay audio app entitlement	• Add com.apple.developer.playable-content to Entitlements.plist.
Show an app icon on the CarPlay home screen	• Add a CarPlay-specific app icon asset.
Present a hierarchical list to navigate and select audio content	• Implement MPPlayableContentDataSource and assign the MPPlayableContentManager dataSource property. • Implement MPPlayableContentDelegate and assign the MPPlayableContentManager delegate property.
Show media information in the Now Playing screen	• Update MPNowPlayingInfoCenter with information about the currently playing audio.
Handle remote control events	• Support MPRemoteCommandCenter events.

Show an app icon on the CarPlay home screen

Provide app icon assets for rendering in CarPlay. Your CarPlay icon should look similar to your app icon displayed on iPhone.

Your app icon will not appear on the CarPlay home screen if the app is missing the CarPlay audio app entitlement.

To specify app icon assets, turn on the CarPlay app icon asset in Xcode and drag your icon into the image well for CarPlay.

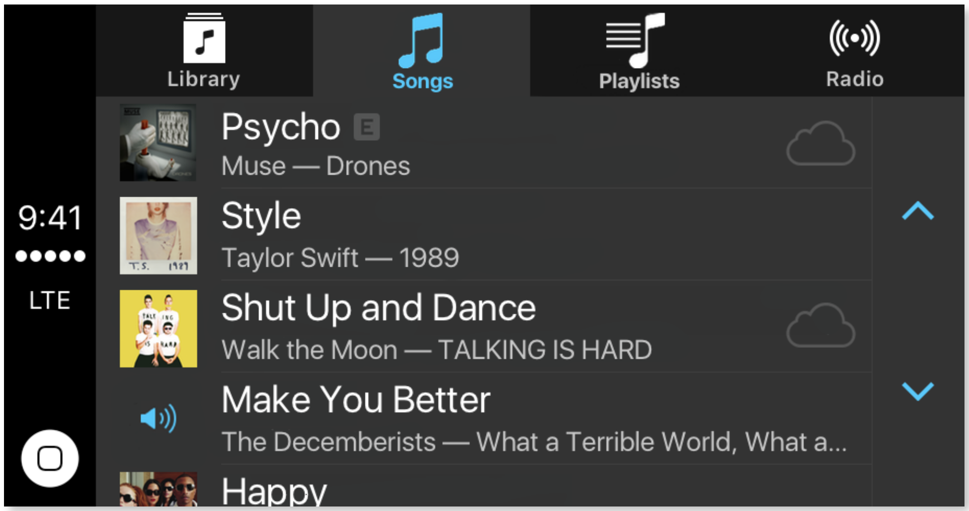


App Icon

Name	AppIcon
Apple Watch	☐ watchOS All Versions
	☐ watchOS 1.0 Specific
CarPlay	✓ All
iPad	☐ iOS 7.0 and Later
	☐ iOS 6.1 and Prior
iPhone	✓ iOS 7.0 and Later
	☐ iOS 6.1 and Prior
Mac	☐ All

Present a hierarchical list to navigate and select audio content

CarPlay audio apps provide a list that allows the user to navigate and select audio content. The list may be as simple as a short, static list of audio content, or a dynamic list with content arranged in a navigable hierarchy. Starting with iOS 10, audio apps can also implement tabs that appear across the top of the screen. Audio playback starts when the user explicitly selects an item to play.



To provide a list of audio content, support the MPPlayableContentManager data source protocol (MPPlayableContentDataSource) and delegate (MPPlayableContentDelegate). When your app is launched, set the data source protocol and delegate using sharedContentManager. For details, see MPPlayableContentDataSource and MPPlayableContentDelegate below.

If you provide a list, the root level list must not be empty. If your root level list is dynamically generated, maintain a cache or present default content so that some information is shown even when the network is unavailable. Remember that users are more likely to encounter poor network reception while driving, and if they encounter an empty list when your app is launched, it will look like your app is not working. Starting with iOS 10, a loading screen will be shown while your app is generating content, and will show an error screen after a period of time if your app does not load content. The user will have the ability to retry fetching content when they are presented with this error screen.

If you provide a list, never start audio playback until the user explicitly selects a playable item. Users expect to be able to browse your list of audio content while the previous audio source, such as the car's FM radio or another audio app is still playing.

MPPlayableContentDataSource
The data source protocol is invoked when CarPlay needs to present a list of audio content. You receive an NSIndexPath which indicates which list in the hierarchy is being requested. Your app responds with the number of items in the requested list and an MPContentItem for each item in the list.

MPContentItem contains metadata for an item in the list. You should always include an identifier and a title. You must also specify whether the item is a container or a playable item. Starting with iOS 8.3, every MPContentItem must have either the container or playable property set to YES. If they are both set to NO then the item will not be functional. Container items are drawn with a chevron icon on the edge indicating that they lead to the next level in the hierarchy, and the view will be animated appropriately when the item is selected. If the item is playable, you are expected to play audio when the item is selected.

The following is a list of properties you can set with each MPContentItem.

Property	Description
identifier	A unique identifier for the item. This is required for the MPPlayableContent APIs to reference and update content accordingly.
title	The title of the item. Usually this is a track name if representing playable content, or a container's name such as a playlist or album.
subtitle	The subtitle of the item. If the subtitle were representing a song, it would represent the artist or composer.
artwork	The artwork for the item.
playbackProgress	Represents the playback progress of the item. You must implement childItemsDisplayPlaybackProgressAtIndexPath in MPPlayableContentDataSource in order to display playback progress. Otherwise, the framework will assume playback progress is not supported for any content items.
explicitContent	The item contains explicit content which may not be suitable for some audiences. Available in iOS 10 and later.
streamingContent	The item is not local to the device and initiating playback for this item will incur data usage. Available in iOS 10 and later.
container	The item is a container of other items, such as media content or other containers. Examples of containers include playlists, albums, and radio stations.
playable	The item is a playable content, and will load the Now Playing screen and initiate playback when tapped.

MPPlayableContentDelegate
The delegate is called when the user selects a playable item.

Your delegate will be informed of which item in the list was selected. Your app is expected to start playing audio at this time. When you are done, call the provided completion handler and the view will automatically transition to the Now Playing screen.

You may optionally return an error in response to the delegate. If you return an error, an error message will be displayed as an alert on the screen.

Important: Your error message should be informational and must never direct the user to pick up their iPhone. For example, if audio could not be played because of a preference or configuration error, don't display an error message that directs the user to adjust settings on their iPhone. Whenever possible, avoid using error alerts since they prevent the user from performing other tasks in CarPlay.

Limitations on depth of hierarchy and list length
The maximum permitted depth of the navigable hierarchy is 5 levels deep. Starting with iOS 9, iOS will not allow you to create hierarchies deeper than 5 levels.

The maximum permitted number of items in a list was unlimited in earlier versions of iOS. Starting with iOS 9, CarPlay can also operate in limited list mode. When CarPlay is operating in limited list mode, all lists are limited to a fixed number of items and anything longer will be truncated. The vehicle determines if and when limited list mode is activated. For example, a vehicle may choose to enter limited list mode when the vehicle exceeds a certain speed, or when the vehicle is shifted out of Park to Drive. The vehicle may enter and exit limited list mode multiple times while your app is running.

You can determine whether limited list mode is currently activate by obtaining the MPPlayableContentManagerContext and checking if contentLimitsEnforced returns YES. You can obtain the MPPlayableContentManagerContext via the MPPlayableContentManager context property.

You can also use the enforcedContentItemsCount property if you want to know the maximum number of items that are permitted in a list. For example, in iOS 9, this value will be 12 when CarPlay is operating in limited list mode. Similarly, the enforcedContentTreeDepth property is the maximum number of levels permitted in the navigable hierarchy. In iOS 9, this value is always 5. The limitations may change in future versions of iOS so you should not rely on the exact values returned by the context.

Updating data
You can directly update an MPContentItem that was already provided by the data source.

To update an item, keep a reference to the MPContentItem before responding to contentItemAtIndexPath and later use setTitle, for example, to update the title. The updated title will automatically appear on the CarPlay display.

If you are changing multiple items, or multiple properties of an item, you should call MPPlayableContentManager beginUpdates prior to updating MPContentItem objects and endUpdates after you have finished updating them. That way, the items will refresh on the screen all at once.

Tabs
Starting in iOS 10, audio apps can use a combination of horizontal tabs and lists to better display content in CarPlay. If you add a keyword to your Info.plist file, items in the root level list that are marked as (container = YES) will be shown as a tab on the CarPlay screen. Make sure that each item in the root level provides template artwork so that each tab has an icon representing the tab's contents. To adopt tabs in CarPlay, add the following to your Info.plist file:

```
<key>UIBrowsableContentSupportsSectionedBrowsing</key>
<true/>
```

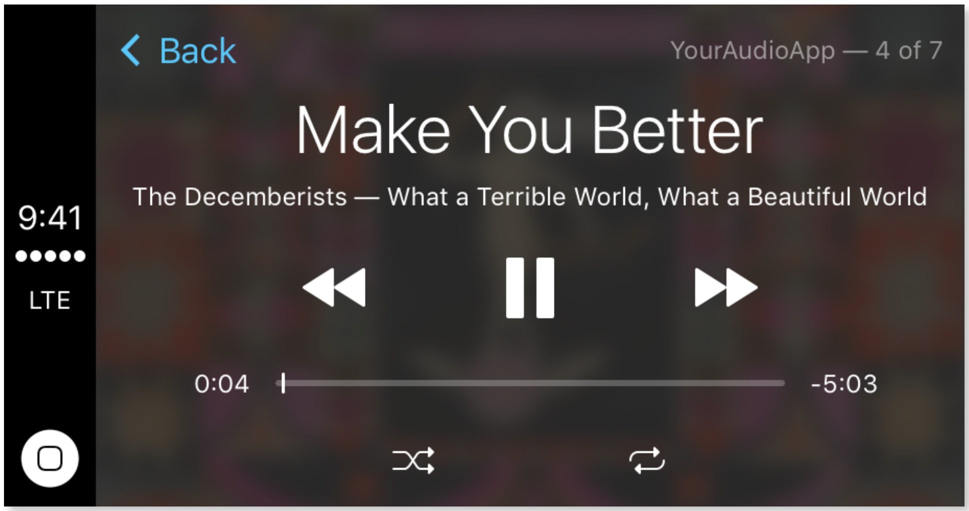
Indicating the currently playing item
Starting in iOS 10, audio apps can show a speaker icon next to the item that is currently playing. When initiating playback, set the nowPlayingIdentifiers property in MPPlayableContentManager with the identifiers of related MPContentItem objects. When the user navigates the list hierarchy, a speaker icon will appear next to MPContentItem objects you designated as currently playing.

Dedicated audio apps that play a single stream
Starting in iOS 10, dedicated audio players that stream audio from a single source and don't require navigation or selection may choose to immediately transition to the Now Playing screen and play content at launch. To initiate playback at launch, add the following to your Info.plist file:

```
<key>UIBrowsableContentSupportsImmediatePlayback</key>
<true/>
```

Show media information in the Now Playing screen

All CarPlay apps populate the Now Playing screen with information about the current audio content.


Back
YourAudioApp — 4 of 7
Make You Better
The Decemberists — What a Terrible World, What a Beautiful World
9:41
LTE
0:04
-5:03

MPNowPlayingInfoCenter
Use MPNowPlayingInfoCenter to populate the CarPlay Now Playing screen with information. MPNowPlayingInfoCenter allows you to specify metadata such as title, artist, elapsed time, playback queue, playback rate and album artwork.

When running your app in the Simulator, the media playback features are not the same as running on iPhone. On iPhone running CarPlay, the play button in the CarPlay Now Playing screen is automatically managed by iOS and changes state based on whether audio is playing. In contrast, the play button in the Simulator relies on the state of the playback rate in Now Playing Info Center. You should ensure that Now Playing Info Center always has the correct playback rate. When playback is stopped, be sure to set the MPNowPlayingInfoPropertyPlaybackRate property to 0.0.

Apple recommends you always supply a MPMediaItemPropertyPersistentID property.

If your app has the concept of a playback queue, set MPNowPlayingInfoPropertyPlaybackQueueIndex and MPNowPlayingInfoPropertyPlaybackQueueCount. Starting in iOS 10, this information is shown on the Now Playing screen.

If your app plays live streaming content, set MPNowPlayingInfoPropertyIsLiveStream to YES.

Handle remote control events

All CarPlay apps must respond to remote control events.

MPRemoteCommandCenter
MPRemoteCommandCenter allows you to receive user remote control events such as play, pause, stop, next track or previous track. Remote control events may be triggered by Siri, by the user manipulating the CarPlay display, or by hard keys such as those on a steering wheel.

If your app supports older UIEvent remote control commands (subtype UIEventSubTypeRemoteControl) you should no longer do so. Adopt MPRemoteCommandCenter instead.

CarPlay does not show all supported remote control commands in the Now Playing screen, but you can suggest which controls should be shown by registering a handler for that command. To register handlers for specific commands, use addTarget in MPRemoteCommand.

Some commands such as rating, like, dislike and bookmark can have additional properties such as custom, localized title strings. If multiple commands are supported, they may automatically be incorporated into an action sheet.

The following remote control events are supported in CarPlay's Now Playing screen:

Remote control event	
pauseCommand	
playCommand	
togglePlayPauseCommand	
nextTrackCommand	
previousTrackCommand	
skipForwardCommand	
skipBackwardCommand	
seekForwardCommand	
seekBackwardCommand	
likeCommand	
dislikeCommand	
stopCommand	Available in iOS 10 and later.
bookmarkCommand	Available in iOS 10 and later.
changeRepeatModeCommand	Available in iOS 10 and later.
changeShuffleModeCommand	Available in iOS 10 and later.

Best Practices

The following section describes best practices for CarPlay audio apps. Also see *Human Interface Guidelines for CarPlay Apps*.

Test on real hardware
While most development can be performed in the Simulator, you will want to test your app on iPhone connected to real CarPlay hardware.

By testing on real hardware you will have a full appreciation of how your app interacts with CarPlay and other vehicle functions. Some features such as the built-in Music app, and app usage while iPhone is locked simply aren't available in the Simulator.

CarPlay is available in many vehicles, or can be purchased as an aftermarket kit for use on a desktop in combination with a 12 V power supply and a set of speakers. For a list of available CarPlay vehicles, see http://www.apple.com/ios/carplay/available-models/.

Test under poor network conditions
Remember that users are more likely to encounter areas of poor network reception while driving. On a short drive, the user may enter and exit areas of network coverage multiple times. Design your app to gracefully handle variable network conditions.

For example, if your audio content list is dynamically generated, maintain a cache or have default content so you can display some information when your app is launched, even when the network is unavailable.

If you are unable to play audio due to poor connectivity, give appropriate user feedback. See Transition to the Now Playing screen for some examples.

Provide a navigable hierarchy of audio information
CarPlay allows your app to display a navigable, hierarchical list of information. You should provide information relevant to your app, such as radio stations, albums, artists or titles.

CarPlay is designed for the driver to use their iPhone in the car with minimal effort. The information you provide should be relevant and succinct.

Only include information directly related to the task at hand and only if applicable to the car experience. Do not attempt to recreate every feature of your app.

Do not provide excessive amounts of information. Minimize the number of items in each list. Also, minimize the depth, or number of levels of hierarchy, that the user needs to navigate to complete a task.

Provide icons in the navigable hierarchy to improve the appearance and usability of your app. Icons must be provided with a transparent background and should include high resolution assets for vehicles with high resolution displays.

Apple recommends that you provide easy, one-touch access to a Shuffle, Play my last or Recommended feature at the top level of your hierarchy so users can quickly start listening, without having to navigate through a complex hierarchy of options.

Populate all relevant item properties
Starting in iOS 10, MPContentItem contains properties that provide useful indications to the user, such as showing whether content is explicit or not stored locally on iPhone.

Set the explicitContent property when you have explicit content, as users may have children or other passengers who do not wish to listen to explicit content.

Set the streamingContent property for content that is not locally stored on iPhone but streamed over the network. Many users have cellular data usage limits, so be mindful when streaming content over the cellular network.

Transition to the Now Playing screen
When the user selects a playable item such as a radio station or song title to request audio playback, CarPlay should transition to the Now Playing screen as soon as content is available for playback. However, it may take several seconds for audio to actually start playing due to buffering or network conditions. Apple recommends that you let the selected item remain highlighted with an animated spinner to indicate that the content is loading. iOS automatically draws the spinner and it will remain until your app returns the completion handler. However, if your app does not return the completion handler for a long period of time, iOS will eventually timeout and display an error screen.

After transitioning to the Now Playing screen, your app may not have complete information about the properties that need to be displayed. In this case, you should provide some interim Now Playing information so the user knows what is happening. For example, you could partially populate the Now Playing Info Center with properties that you already know and later update all properties with real data, for example after you have loaded metadata from a server.

Audio playback at app Launch
Most apps should only open an audio session when the user explicitly starts playing audio by selecting an item from your navigable list. Don't open an audio session immediately after your app launches or resumes since the user may just be browsing your app while listening to other media, such as another app on iPhone or the FM radio in the vehicle.

However, if your app shows no list and simply plays audio from a single audio source when launched, you can choose to open an audio session immediately after your app launches. In this case, you must also set UIBrowsableContentSupportsImmediatePlayback to YES in your app's Info.plist.

Be sure to update all relevant Now Playing properties when audio starts or stops, including the currently playing item and properties such as the playback rate and elapsed playback time. By providing all information, iOS can show accurate information in lists and in the Now Playing screen.

If the user was listening to your app when they exited the car, iOS may attempt to automatically start audio playback when the user comes back in to the car. Make sure your app supports this use case.

Audio playback after interruptions
While users are operating your app, they may choose to perform a task that interrupts audio playback. For example, they may receive an incoming phone call or the user may activate Siri to make a request.

After the temporary audio interruption, your app should resume audio where it left off without requiring user intervention.

Alerts and modal behavior
When using CarPlay, iPhone's screen remains turned off by default and iPhone may be in an inaccessible location, such as in the car's glove compartment or even the trunk. You must avoid any behavior that locks out CarPlay functionality or requires user intervention on iPhone while driving.

If your app displays alerts or other messages on iPhone's screen, these should not interfere with CarPlay or interrupt media playback. Be sure to test your app thoroughly so that login messages, error alerts, update notices, advertisements or any other content shown on iPhone do not unnecessarily disrupt media playback in CarPlay. If iPhone's screen is not visible, the user won't know why media playback has stopped.

If there are unavoidable situations where CarPlay simply won't work, you should provide appropriate feedback to the user. See Error conditions below.

Error conditions
If your app cannot fulfill user requests to interact with your app using CarPlay, you can provide feedback within CarPlay so the user understands what is happening. Do not rely on error messages displayed on iPhone itself.

In some cases you can provide feedback by returning an appropriate error and a localized description so it is displayed as an alert in CarPlay. Alternatively, you can provide brief feedback by populating your app content with an unplayable item whose text describes the error condition.

Use alerts sparingly and only when absolutely necessary to communicate important information to the user. Never display a message that directs the user to perform an operation on iPhone itself. Always assume iPhone is physically inaccessible while CarPlay is active.

For example, although it's recommended that your app is immediately usable after obtaining it from the App Store, some apps may require a login or setup process that can only be done on iPhone. In this case, you can provide feedback to the user in CarPlay so they know that a setup step is required, but the feedback should never suggest that the user pick up and manipulate iPhone.

Think carefully about how to handle temporary network failures. If the user is driving in a region with intermittent network coverage you don't want to present unnecessary alerts.

Working while iPhone is locked
Starting with iOS 8.3, CarPlay may be used while iPhone is locked.

When using CarPlay for the first time, the user is prompted whether they would like to allow this behavior when connected to the vehicle again.

One side effect of this is that your app may be launched while iPhone is locked, so you need to ensure that your app works correctly when iPhone is locked.

You should test your app, particularly the initial launch sequence, if you are performing any functions that require iPhone to be unlocked. For example, if you are using the keychain to store data, you need to make sure it works when iPhone is locked.

Your app will not be able to access any of the following when launched or running while iPhone is locked:

- Files saved with NSFileProtectionComplete or NSFileProtectionCompleteUnlessOpen
- Keychain items with a kSecAttrAccessible attribute of kSecAttrAccessibleWhenPasscodeSetThisDeviceOnly, kSecAttrAccessibleWhenUnlocked or kSecAttrAccessibleWhenUnlockedThisDeviceOnly
- SQLite databases created with SQLITE_OPEN_FILEPROTECTION_COMPLETE or SQLITE_OPEN_FILEPROTECTION_COMPLETEUNLESSOPEN

For more information see *Protecting Data Using On-Disk Encryption* in *App Programming Guide for iOS*.

Test new installs
Before you submit your app to the App Store, you should do a run test in CarPlay with a new install on a clean iPhone to make sure the installation and setup process works as expected. Pay close attention to any sign-in requirements.

Publishing Your CarPlay Audio App

When you are ready to publish your CarPlay audio app on the App Store, follow the same process as for any iOS app and use iTunes Connect to submit your app.

Revision History

This table describes the changes to CarPlay Audio App Programming Guide.

Release	Notes
March 2017	• Editorial updates.
June 2016	• Starting with iOS 10, CarPlay audio apps can immediately begin playback on app launch, which will launch to the Now Playing screen. • Updated documentation on new properties that can be set in MPNowPlayingInfoCenter for CarPlay for iOS 10, such as live streaming, explicit content, or playback queue index. • Updated new properties for MPContentItem in iOS 10, which include the ability to show explicit content and streaming content. • Introduced nowPlayingIdentifiers for MPPlayableContentManager for iOS 10, which allow the UI to display what content is currently playing. • Updated descriptions on how errors are handled, and when the CarPlay screen will show a timeout for root level content. • In Transition to the Now Playing screen, updated the description on what action to take when starting audio playback.