# Table of Contents

- Introduction ............................................................................................................. 3
- CarPlay Navigation Apps .......................................................................................... 4
  - CarPlay Navigation App Entitlement ..................................................................... 4
- Development Environment .......................................................................................... 5
  - Provisioning Profile ............................................................................................... 5
  - Simulator .................................................................................................................. 6
- CarPlay Navigation App Guidelines ............................................................................ 9
- CarPlay Navigation App UI Elements .......................................................................... 10
  - Base View ............................................................................................................... 10
  - Map Template ......................................................................................................... 11
  - List Template ......................................................................................................... 12
  - Grid Template ........................................................................................................ 13
  - Search Template .................................................................................................... 14
  - Trip Preview Panel .................................................................................................. 15
  - Route Guidance Panel ............................................................................................ 16
  - Navigation Alert Panel ........................................................................................... 17
  - Voice Control Template .......................................................................................... 18
  - Alert ........................................................................................................................ 19
- Icons and Images ......................................................................................................... 20
- CarPlay Framework Classes ......................................................................................... 22
  - CPInterfaceController ......................................................................................... 22
  - CPMapContentWindow .......................................................................................... 22
  - CPMapTemplate ...................................................................................................... 22
  - CPGridTemplate ..................................................................................................... 22
  - CPLISTemplate ....................................................................................................... 22
  - CPSearchTree ......................................................................................................... 22
  - CPVoiceControlTemplate ....................................................................................... 23
  - CPAler .................................................................................................................... 23
  - CPNavigationAlert ................................................................................................. 23
  - CPSessionConfiguration ......................................................................................... 23
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, VoIP calling apps, automaker apps, and navigation apps that you’ve downloaded to your iPhone.

This guide describes how to create a CarPlay navigation app.
CarPlay Navigation Apps

Users download CarPlay navigation apps from the App Store and use them on iPhone like any other app. When an iPhone with a CarPlay navigation app is connected to a CarPlay vehicle, the app icon appears on the CarPlay home screen. CarPlay navigation apps are not separate apps—you add CarPlay support to an existing navigation app.

CarPlay navigation apps are designed to look and feel like your app on iPhone, but with familiar UI elements that are similar to the built-in CarPlay apps.

Your app is responsible for drawing a map on the CarPlay screen, and presenting UI elements to the user. iOS manages the display of UI elements and handles the interface with the car. Your app does not need to manage the layout of UI elements for different screen resolutions, or support different input hardware such as touchscreens, knobs, or touch pads.

CarPlay navigation apps must meet the basic requirements defined in CarPlay Navigation APIs Addendum to the Apple Developer Program License Agreement, and must follow the CarPlay Navigation App Guidelines.

For general design guidance, see Human Interface Guidelines for CarPlay Apps.

CarPlay navigation apps cannot combine other CarPlay features such as audio, messaging, or VoIP calling into a single app.

CarPlay navigation apps require iOS 12 and Xcode 10.

CarPlay Navigation App Entitlement

CarPlay navigation apps require a CarPlay navigation app entitlement.

CarPlay navigation apps must be designed primarily to provide navigation services to a user.

Other kinds of apps, including apps designed primarily to locate points of interest, apps designed primarily to provide services other than navigation, and apps that don’t provide route guidance, are not eligible for the CarPlay navigation app entitlement.

To request a CarPlay navigation app entitlement, go to http://developer.apple.com/carplay and provide information about your app. You must also agree to the CarPlay Navigation APIs Addendum to the Apple Developer Program License Agreement.

Apple will review your request. If your app meets the criteria for a CarPlay navigation app, Apple will create a CarPlay navigation app entitlement and notify you.
Development Environment

The Xcode developer tools include everything you need to create apps for CarPlay. Like all apps on iPhone, CarPlay navigation apps use the iOS SDK.

Provisioning Profile

Once you have received the CarPlay navigation app entitlement, log in to your account on the Apple Developer website and create a new Provisioning Profile that includes the entitlement.

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

In Xcode, you will also need to create an `Entitlements.plist` file in your project, if you don't have one already. Create a key for the CarPlay navigation app entitlement as a boolean.

```xml
/keycom.apple.developer.carplay-maps</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

Simulator allows you to run your CarPlay navigation app in a second window. The second window acts as the car’s display and allows you to interact with it in a similar manner to when you are connected to a CarPlay system.

While Simulator is useful during development, certain CarPlay features are not available in Simulator and you should not rely on it as the sole method to develop your app.

Here are some examples of features that you cannot test in Simulator:

- **Testing while iPhone is locked.** You need to ensure that your app performs normally while iPhone is locked, since CarPlay is often used while iPhone is in a locked state. With wireless CarPlay, iPhone may be completely inaccessible and stored in a pocket or a bag.

- **Testing audio interactions.** You need to ensure that your app correctly plays voice prompts while other audio sources are playing. For example, when the vehicle’s FM radio is playing, voice prompts cause the FM radio volume level to be temporarily lowered, then raised after the voice prompt has finished playing.

It is highly recommended to develop and test navigation apps using a car or aftermarket system 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 is supported by default when you run Simulator. However, you should configure the Simulator with extra options when developing a CarPlay navigation app. To enable extra options, enter the following command in Terminal before launching Simulator.

```bash
defaults write com.apple.iphone simulator CarPlayExtraOptions -bool YES```
From the Simulator menu, select Hardware, External Displays, and CarPlay... to show the CarPlay screen.

Show CarPlay as a second window
If you configured the Simulator with extra options, you can modify the CarPlay screen width, height, and scale.

By default, the CarPlay screen is configured as a display that is 800 pixels wide and 480 pixels high. In this configuration CarPlay will use a scale of 2.0. This is typical of many CarPlay systems. At higher resolutions, CarPlay automatically uses a scale of 3.0.

It is recommended that you test your app using all of the following settings. In particular, you need to ensure that your base map renders correctly at all resolutions and scales.

<table>
<thead>
<tr>
<th></th>
<th>Width and Height</th>
<th>Scale</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum</td>
<td>748px x 456px</td>
<td>2.0</td>
</tr>
<tr>
<td>Minimum</td>
<td>748px x 456px</td>
<td>2.0</td>
</tr>
<tr>
<td>Standard</td>
<td>800px x 480px</td>
<td>2.0</td>
</tr>
<tr>
<td>Standard</td>
<td>800px x 480px</td>
<td>2.0</td>
</tr>
<tr>
<td>High resolution</td>
<td>1920px x 720px</td>
<td>3.0</td>
</tr>
<tr>
<td>High resolution</td>
<td>1920px x 720px</td>
<td>3.0</td>
</tr>
<tr>
<td>Portrait</td>
<td>768px x 1024px</td>
<td>2.0</td>
</tr>
<tr>
<td>Portrait</td>
<td>768px x 1024px</td>
<td>2.0</td>
</tr>
</tbody>
</table>
CarPlay Navigation App Guidelines

CarPlay navigation apps must adhere to the following guidelines.

1. **The base view must be used exclusively to draw a map.** Do not draw windows, alerts, panels, overlays, or user interface elements in the base view. For example, don’t draw lane guidance information in the base view. Instead, draw lane guidance information as a secondary maneuver using the provided template.

2. **Use each provided template for its intended purpose.** For example, maneuver images must represent a maneuver and cannot represent other content or user interface elements.

3. **Do not instruct the user to perform tasks on iPhone.** It’s not appropriate to ask the user to manipulate their iPhone while using CarPlay. If there is an error condition, such as a required log in, just inform the user of the error so they can take action when safe.

4. **Provide a way to enter panning mode.** If your app supports panning, you must include a button in the map template that allows the user to enter panning mode since drag gestures are not available in all vehicles. Drag gestures must only be used for panning the map.

5. **Immediately terminate route guidance when requested.** For example, if the user starts route guidance using the vehicle’s built-in navigation system, your app delegate will receive a cancelation notification and must immediately stop route guidance.

6. **Correctly handle audio.** Voice prompts must work concurrently with the vehicle’s audio system (such as FM radio) and your app should not needlessly activate audio sessions when there is no audio to play.

7. **Supported countries.** Ensure that your map is appropriate in each supported country.

8. **Be open and responsive to feedback.** Apple may contact you in the event that Apple or automakers have input to design or functionality.
CarPlay Navigation App UI Elements

CarPlay navigation apps are built from a fixed set of UI templates that iOS renders on the CarPlay screen.

Navigation apps are responsible for selecting which UI template to show on the screen (the controller), and providing the data to be shown inside the UI template (the model). iOS is responsible for rendering the information in CarPlay (the view). The exception is the base view where navigation apps are responsible for drawing their own map.

Base View

All navigation apps start with a base view. The base view is where the map is drawn. The base view must be used exclusively to draw a map, and may not be used to draw windows, alerts, panels, overlays, or other UI elements. Instead, navigation apps overlay UI elements such as the navigation bar and map buttons using the provided templates.

Navigation apps are designed to work with a variety of car input devices, and CarPlay does not support direct user interaction in the base view (apps do not directly receive tap or drag events).

Base View
**Map Template**

The Map Template is a control layer that appears as an overlay over the base view and allows you to present user controls.

The control layer consists of a navigation bar and map buttons. By default, the navigation bar appears when the user interacts with the app, and disappears after a period of inactivity. Your app can customize this behavior, including whether to hide the map buttons.

The navigation bar includes up to two leading buttons and two trailing buttons. You can customize the appearance of these buttons with icons or text.

The control layer may also include up to four map buttons. The map buttons are always shown as icons.

Navigation apps enter panning mode, zoom in or out, and perform other functions by responding to user actions on these buttons.

*Map Template with navigation bar and map buttons*
List Template

The List Template allows navigation apps to present a hierarchical list of menu items. It includes a navigation bar and a list.

The navigation bar includes a title, up to two leading buttons, and up to two trailing buttons. You can customize the appearance of these buttons with icons or text.

The list contains items, optionally separated into sections.

Each item in the list may include an icon, title, subtitle, and an optional disclosure indicator indicating the presence of a submenu.

The depth of the menu hierarchy may not exceed five levels. Also, some cars limit the total number of items that may be shown in a list.
**Grid Template**

The Grid Template allows navigation apps to present a grid of menu items. It includes a navigation bar and a grid.

The navigation bar includes a title, and up to two leading buttons and two trailing buttons. You can customize the appearance of these buttons with icons or text.

The grid may include up to eight items. Each item in the grid includes an icon and a title.
**Search Template**

The Search Template allows navigation apps to present a destination search screen. It includes a search field, a cancel button, a list of search results, and a keyboard view.

The keyboard is automatically shown in the appropriate language and is managed by iOS. Note that some cars limit when the keyboard may be shown.

Typing anything in the keyboard automatically enters text in the search field. Navigation apps are responsible for parsing the search text and updating the list of search results. Navigation apps must also respond when the user selects an item from the list of search results.

![Search Template](image)
**Trip Preview Panel**

The Trip Preview Panel allows navigation apps to present a list of route options for the selected destination.

Route information is shown in the Trip Preview Panel. Navigation apps provide information on each route, including a name, distance, and estimated travel time. If more than one route is available, a “More Routes...” button will be shown automatically, allowing the user to preview a different route.

Each time the user previews a route, navigation apps are responsible for updating the base view in the Map Template to provide a visual representation of the route.

A “Go” button is always shown. When the user selects “Go”, navigation apps should clear the Trip Preview Panel and begin route guidance to the destination.
**Route Guidance Panel**

CarPlay navigation apps show information about upcoming maneuvers in the route guidance panel.

The route guidance panel shows one maneuver at a time, but in cases where there are maneuvers in rapid succession, two maneuvers may be shown on the screen. The second maneuver may also be used to show lane guidance information for the first maneuver.

Each maneuver may include instruction text, a symbol, estimated remaining distance and time.

In addition to the route guidance panel, maneuvers may also be shown in notifications, or sent to vehicles that support the display of CarPlay metadata in their instrument cluster or heads up display.

In addition to upcoming maneuvers, navigation apps should continuously update travel estimates for the overall trip.
**Navigation Alert Panel**

The Navigation Alert Panel allows navigation apps to show a navigation alert panel that is overlaid on the map. If the navigation app is running in the background, high priority alerts will also appear over the active app in CarPlay.

Navigation alert panels include a title and a primary action button, and may also include an image, subtitle, and secondary action button. You can also specify a duration for which the alert should be visible before automatically being dismissed.

![Navigation Alert Panel Diagram](image-url)
Voice Control Template
The Voice Control Template allows navigation apps to show a voice control indicator.
Alert
Alerts allow you to present an alert or action sheet. Action sheets are recommended when you have more than two choices.
Icons and Images

Your app is responsible for providing icon and image assets for CarPlay, including an app icon, navigation and map button icons, maneuver symbols, alert icons, and other UI elements.

All assets must be provided in both 2x and 3x variants. Some CarPlay systems use a 2x scale, while higher resolution screens use a 3x scale.

Your CarPlay app icon must look similar to your app icon displayed on iPhone.

To provide icon and image assets, turn on CarPlay assets in Xcode.

Enable CarPlay app icon assets

Next, drag 2x and 3x assets into the CarPlay image wells.

CarPlay 2x and 3x image wells
Use the following sizes for guidance when preparing image assets.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Maximum sizes</th>
</tr>
</thead>
</table>
| CarPlay app icon | 180px x 180px (60pt x 60pt @ 3x)  
120px x 120px (60pt x 60pt @ 2x) |
| Primary maneuver symbol  
(symbol and instruction on one line) | 150px x 150px (50pt x 50pt @ 3x)  
100px x 100px (50pt x 50pt @ 2x) |
| Primary maneuver symbol  
(symbol and instruction on two lines) | 360px x 150px (120pt x 50pt @ 3x)  
240px x 100px (120pt x 50pt @ 2x) |
| Secondary maneuver symbol  
(when specifying CPManeuverDisplayStyleDefault,  
CPManeuverDisplayStyleLeadingSymbol, and  
CPManeuverDisplayStyleTrailingSymbol) | 360px x 150px (18pt x 18pt @ 3x)  
240px x 100px (18pt x 18pt @ 2x) |
| Secondary maneuver symbol  
(when specifying CPManeuverDisplayStyleSymbolOnly) | 360px x 54px (120pt x 18pt @ 3x)  
240px x 36px (120pt x 18pt @ 2x) |
CarPlay Framework Classes

CarPlay navigation apps use the following classes in the CarPlay framework to present a user interface on the CarPlay screen.

**CPInterfaceController**

CPInterfaceController manages all user interface elements that are overlaid on your map on the CarPlay screen. You don't need to create a CPInterfaceController since it is created for you. Use CPInterfaceController to present templates on the screen, including CPMapTemplate, CPGridTemplate, CPListTemplate, CPSearchTemplate, and CPVoiceControlTemplate, and to present alerts using CPAlert.

**CPMapContentWindow**

CPMapContentWindow represents the window where you will draw your map.

**CPMapTemplate**

CPMapTemplate presents controls that appear directly on the map, including the navigation bar and map buttons. CPMapTemplate is also used to manage turn by turn guidance, present navigation alerts, and manage the map's panning interface. Create a CPMapTemplate when didConnectCarInterface is called on the application delegate.

**CPGridTemplate**

CPGridTemplate presents a grid of menu items on the CarPlay screen. Populate the grid with an array of buttons defined using CPGridButton. Each grid button includes an image, a title, and a handler that is called when the user selects the button.

**CPListTemplate**

CPListTemplate presents a list of menu items on the CarPlay screen. Populate the list with an array of sections defined using CPListSection. Each section contains an array of items defined using CPLListItem. Each list item includes a title, and may also include a subtitle, an image, and a disclosure indicator. Use nil to omit optional parameters. Your delegate responds to user selection by listening for didSelectListItem.

**CPSearchTemplate**

CPSearchTemplate presents a destination search interface on the CarPlay screen that includes a list and a keyboard. Your delegate responds to keyboard input by listening to updatedSearchText, and provides an array of CPLListItem elements that contain matching destination entries. Respond to a destination selection by listening for selectedResult.
**CPVoiceControlTemplate**

CPVoiceControlTemplate presents a voice control indicator. CarPlay navigation apps must show the voice control template during audio input.

**CPAlert**

Use CPInterfaceController presentAlert: and CPAlert to display a modal alert on the CarPlay screen, either as a full screen alert or as an action sheet. Alerts can include up to three (3) action buttons. Your CPAlertAction handler responds to user selection on the action buttons. Use CPInterfaceController dismissAlertAnimated: to dismiss an alert without user interaction.

**CPNavigationAlert**

Use CPMapTemplate presentNavigationAlert: and CPNavigationAlert to display a navigation alert panel that is overlaid on the map. Navigation alerts include a primary action button, an optional secondary action button, and an optional timeout after which it will be automatically dismissed. Your CPAlertAction handler responds to user selection on the action buttons. Use CPMapTemplate dismissNavigationAlertAnimated: to dismiss a navigation alert without user interaction. Alerts with CPNavigationAlertPriorityHigh are shown even when the navigation app is running in the background.

**CPSessionConfiguration**

Use CPSessionConfiguration to receive notifications when the car limits user interface elements such as keyboard display and maximum list length. Limitations can change dynamically while navigation apps are running.

**CPTrip**

CPTrip represents a journey consisting of an origin, a destination, and up to three (3) route choices defined using CPRouteChoice. Create CPTrip objects to represent journeys to preview trip information on the CarPlay screen.

**CPManeuver**

CPManeuver represents a single instruction, such as an upcoming turn. It includes an instruction, an optional symbol, and initial estimates for remaining time and distance.

**CPNavigationSession**

CPNavigationSession represents an active route guidance session.
CarPlay Navigation App Development

**Startup**

When your app becomes active in CarPlay, you will receive a CPInterfaceController and a CPMapContentWindow through CPApplicationDelegate didConnectCarInterface.

Retain references to both the interface controller and the map content window throughout the duration of the CarPlay session.

```swift
self.interfaceController = interfaceController
self.carWindow = window
```

Next, create a new view controller and assign it to the window’s root view controller. You will use the view controller to manage your map content as the base view in the window.

```swift
let rootViewController = MyRootViewController()
window.rootViewController = rootViewController
```

Finally, create a map template and assign it as the root template.

```swift
let rootTemplate: CPMapTemplate = createRootTemplate()
self.interfaceController?.setRootTemplate(rootTemplate, animated: false)
```

Create a default set of navigation bar buttons and map buttons and assign them to the root map template. Specify navigation bar buttons by setting up the leadingNavigationBarButtons and trailingNavigationBarButtons arrays. Specify map buttons by setting up the mapButtons array.

If your CarPlay navigation app supports panning, one of the buttons you create must be a pan button that lets the user enter panning mode. The pan button is essential in vehicles that don’t support panning via the touch screen.

You can update the navigation bar buttons and map buttons dynamically based on the state of the app. For example, during active route guidance, you may choose to replace the default navigation bar buttons with an option to end route guidance.
**Handling the base map view**

CarPlay navigation apps draw their map in the base view. You are responsible for creating the base view and attaching it to the window provided when the CarPlay session starts. The base view must be used exclusively to draw a map, and may not be used to display other UI elements.

CarPlay supports vehicle screens of different shapes and resolutions, so your map needs be capable of being drawn in a wide variety of aspect ratios and scales. See the section on Simulator for more information on which screen resolutions and scales to test with.

CarPlay also supports different car input devices, including touch screens, touch pads, and rotating knob interfaces. CarPlay does not support direct user interaction in the map base view (your app will not receive direct touch events).

In order to handle zooming and panning, create action buttons in the map template. For example, to handle zooming you can create “zoom out” and “zoom in” map buttons and change the scale of your map in response to user action on the buttons.

![Create buttons in the map template, including those for pan and zoom](image)

If your CarPlay navigation app supports panning, one of the buttons must be a pan button that lets the user enter panning mode, and you must respond to the panning functions in CPMapTemplate.
Providing route guidance

All CarPlay navigation apps are expected to follow a standard flow for selecting a destination and getting route guidance.

Select destination. All route guidance starts with the user selecting a destination, whether that is the result of an on-screen search, voice command, or picking a category or destination from a list.

Preview. When a destination is selected, the user is shown a preview of the trip. At the same time, your map in the base view typically shows a visual representation of the trip. The preview also supports disambiguation when there are multiple matching destinations. For example, if the user chooses to navigate to a nearby park, the preview may show up to 12 nearby parks to choose from.

Select route and begin. Once the user has confirmed the destination, they may start route guidance. If there are multiple possible routes, you may present the routes as options for the user to choose from.

Show turn by turn. When the user starts route guidance, show real time information including upcoming maneuvers, and travel estimates (distance and time remaining) for the trip.

End navigation. Route guidance continues until the user arrives at the destination, or chooses to end route guidance.
Select destination

Use the template management methods in CPInterfaceController to present templates that allow the user to specify a destination.

To present a new template, use pushTemplate with a supported CPTemplate class such as CPGridTemplate, CPListTemplate, CPSearchTemplate, or CPVoiceControlTemplate.

For example, to implement a “Favorites” function with categories of interest represented by icons and names, you can use CPGridTemplate to show a grid of icons. Create a new CPGridTemplate and use pushTemplate to present it on the screen.

Show a grid of icons

When the user takes action on a template by selecting an item or canceling the selection, your delegate will be called with a completion handler that includes information about the action that was taken.

You may present multiple templates in succession to support hierarchical selection. For example, you can show a list template that includes list items which lead to additional sublists when selected. Be sure to set showsDisclosureIndicator to true for list items that support hierarchical browsing, and push a new list template when the list item is selected. However, hierarchical selections must never exceed 5 levels of depth.

When the user selects a destination, proceed to the next step and show trip previews.
Preview

When you are ready to show trip previews, use `CPMapTemplate showTripPreviews` to provide an array of up to 12 `CPTrip` objects.

Each `CPTrip` object represents a journey consisting of an origin, a destination, up to 3 route choices, and estimates for remaining time and distance.

Specify available routes using `CPRouteChoice`. Each route includes a summary and an optional description. The summary should describe the route for user disambiguation. For example, the summary may be “Fastest Route” or “Using Surface Streets.” The optional description provides additional information such as “Traffic is light.”

```swift
let routeChoice = CPRouteChoice(summaryVariants: ["Fastest Route"],
additionalInformationVariants: ["Traffic is light."])
```

Provide the summary and description as arrays of variable length strings in descending order of length. CarPlay will display the longest string that fits in the available space on the screen. The array of instructions must be provided in descending order of length (longest string first).

For each `CPTrip`, be sure to provide travel estimates using `CPMapTemplate updateEstimates:` and update the estimates if the remaining time or distance change.

You may also customize the names of the start, overview, and additional routes buttons shown in the trip preview panel.

Select route and begin

When the user selects a different route to preview, the delegate `selectedPreviewFor:` will be called. Respond by updating your map base view.

If the user decides to start a trip, the delegate `startedTrip:` will be called. Respond by starting route guidance. At this time, use `CPMapTemplate hideTripPreviews` to dismiss the trip preview panel.

```swift
mapTemplate.hideTripPreviews()
```

Next use `CPMapTemplate startNavigationSession` to start a navigation session for the selected trip and obtain a `CPNavigationSession` object that represents the navigation session.

```swift
let session = mapTemplate.startNavigationSession(for: trip)
```

While you are calculating initial maneuvers, set the navigation session pause state to `CPTripPauseReasonLoading` so that CarPlay can display the correct state.

```swift
session.pauseTrip(for: .CPTripPauseReasonLoading)
```

At this time, update the navigation bar buttons and map buttons to provide appropriate actions for the user to manage their route.
Show turn by turn

During turn by turn guidance, show route guidance information by updating `upcoming Maneuvers` with information on upcoming turns. Each CPManeuver represents a single maneuver and may include a symbol, an instruction, and estimates for remaining time and distance.

![Show a maneuver in the route guidance panel](image)

**Symbol.** If the maneuver has an associated symbol, such as a turn right arrow, provide an image using `symbolSet`. The symbol will be shown in the route guidance card and any related notifications. You must provide two image variants using CPImageSet—one is used for rendering the symbol on light backgrounds, the other is used for rendering the symbol on dark backgrounds.

**Instruction.** Provide an instruction using `instructionVariants` which is an array of strings. Use the array to provide variants of different lengths so that CarPlay can display the instruction that best fits in the available space on the screen. For example, if the maneuver requires you to turn right on the street named “Solar Circle” you may choose to provide 3 instruction variants “Turn Right on Solar Circle,” “Turn Right on Solar Cir.,” and “Turn Right”. CarPlay will display the instruction with the longest string length that fits in the available space. The array of instructions must be provided in descending order of length (longest string first). You may optionally provide `attributedInstructionVariants` to include embedded images in the instruction. This is useful if you need to display special symbols, such as a highway symbol, as part of the instruction. Note that other text attributes including text size and fonts will be ignored. If you provide `attributedInstructionVariants`, always provide `text-only instructionVariants` since CarPlay vehicles may not always support attributed strings.

Maintain at least one upcoming turn in the `maneuvers` array at all times. In cases where there are two maneuvers in quick succession, provide a second maneuver which may be shown on the screen simultaneously.

If you provide a second maneuver, you can customize its appearance by specifying a symbol style. In CMapTemplateDelegate, return a CPManeuverDisplayStyle for the maneuver when requested. The display style only applies to the second maneuver.
If your app provides lane guidance information, you must use the second maneuver to show lane guidance. Create a second maneuver containing symbolSet with dark and light images that occupy the full width of the guidance panel (maximum size 120pt x 18pt), provide an empty array for instructionVariants, and in the CMapTemplateDelegate, return a symbol style of CPManeuverDisplayStyleSymbolOnly for the maneuver.

Show a maneuver with lane guidance information

Your app is responsible for continuously updating estimates for remaining time and distance for each maneuver, and for the overall trip. Use CPNavigationSession updateEstimates: to update estimates for each maneuver, and CMapTemplate updateEstimates to update overall estimates for the trip. Only update the values when significant changes occur, such as when the number of remaining minutes changes.

If you need to display an alert related to the map or navigation, create a CPNavigationAlert and use CMapTemplate present to show it. Navigation alerts can be configured to automatically disappear after an fixed interval. They may also be shown as a notification, even when your app is not in the foreground.

For each maneuver and navigation alert, specify whether it should be shown as a CarPlay notification when your app is running in the background. Respond to the shouldShowNotificationFor delegate call to specify the maneuver or navigation alert behavior. In the case of a maneuver, you can optionally include updating travel estimates as part of the notification.
End navigation

When route guidance is paused, canceled, or finished, call the appropriate method in CPNavigationSession.

In some cases, CarPlay route guidance may be canceled by the system. For example, if the car’s native navigation system starts route guidance, CarPlay route guidance automatically terminates. In this case, your delegate will receive mapTemplateDidCancelNavigation and you should end route guidance immediately.
Audio prompts

CarPlay navigation apps should use the following audio configuration when playing prompts for upcoming maneuvers.

- Set AVAudioSessionCategory to AVAudioSessionCategoryPlayback. For details, see the documentation for AVAudioSessionCategoryPlayback.
- Set AVAudioSessionMode to AVAudioSessionModeVoicePrompt. For details, see the documentation for AVAudioSessionModeVoicePrompt.

Note that AVAudioSessionModeVoicePrompt is only available when the audio category is set to AVAudioSessionCategoryPlayback.

Audio prompts in CarPlay navigation apps are played over a separate audio channel so they can be mixed with other audio sources in the car, including the car’s own audio sources such as FM radio.

Specify both of the following options.

- AVAudioSessionCategoryOptionInterruptSpokenAudioAndMixWithOthers allows your audio prompts to pause certain apps with spoken audio (such as podcasts or audio books) and mix with other apps. For details, see the documents for AVAudioSessionCategoryOptionInterruptSpokenAudioAndMixWithOthers.
- AVAudioSessionCategoryOptionDucking allows ducking (lowering the volume) for other apps such as music while your audio is played. For details, see the documents for AVAudioSessionCategoryOptionDucking.

Finally, always call setActive YES on your AVAudioSession when an audio prompt is about to play. For a better user experience, you may keep the audio session active if you know that multiple audio prompts are going to be played in succession, in a short amount of time. While your AVAudioSession is active, music apps will remain ducked, and apps with spoken audio will remain paused. Don’t hold on to the active state for more than few seconds if audio prompts are not playing.

When you are done, call setActive NO on your AVAudioSession to allow other audio to resume.
Publishing Your CarPlay Navigation App

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

Ensure that your app follows the CarPlay Navigation App Guidelines. You may be requested to submit a video demonstrating that your app meets the CarPlay Navigation App Guidelines. When submitting a video,