Android for Cars App Library design guidelines

Design navigation and point-of-interest (POI) apps for Android Auto and Android Automotive OS (AAOS)
Contents

01 Introduction
02 Designing your app
03 Customizing app styling
04 UX guidelines
05 Templates
06 Sample flows
07 Components

What’s new
In 1.3-beta update

More list text when parked
Timed buttons
Floating navigation bar
New sample flows
Introduction

The Android for Cars App Library helps you build navigation and point-of-interest (POI) apps that work with both of the Android for Cars systems: Android Auto (projected from phones into cars) and Android Automotive OS (built into cars). Using this library, you can design once to reach both sets of users.

The Android for Cars App Library:

- Provides a set of vehicle-optimized templates for designing your app’s user interface
- Takes care of important functions such as responsive screen sizing and input handling, so your app can work effectively in all compatible vehicles (for details, visit Who handles what)

This deck provides design guidance for creating apps with the templates. It also describes how users experience your app on each of the Android for Cars systems.

Note: While the design process with the templates is the same for both Android Auto and Android Automotive OS (AAOS), the development process involves producing two different APKs. For technical guidance on building an app with the templates, visit Using the Android for Cars App Library.
How users experience your app

Users can experience apps built with the Android for Cars App Library in 2 ways:

1) Projected from phones (Android Auto): Users who install an Android Auto-compatible app on their phones can project the app experience onto the screens of all compatible vehicles. The connection can occur either wirelessly or over a USB cable.

When projected from the phone, your app will use the same colors and styling across all compatible vehicles.

2) Downloaded into cars (AAOS): In vehicles that have Android Automotive OS (AAOS) built in, users can download your app from Google Play directly into the vehicle. They can then run your app in the vehicle without needing to connect a phone.

When your app is downloaded into cars, vehicle OEMs can adjust colors and customize styling to fit specific vehicle models.
Who handles what

Android for Cars App Library takes care of the following, in general:

- **Tactile input:** Handling user input to templates via mechanisms available in specific cars, such as touchscreen or rotary

- **Screen sizing:** Adapting content to screen sizes

- **Screen transitions:** *Motion transition* between screens

- **Consistent, driving-optimized UI:** Ensuring that the UI and interaction patterns are familiar and consistent across apps

- **Light & dark mode** (except as noted*): Adjusting template features to appropriate mode for ambient light conditions

- **UX restrictions based on driving state:** Limiting text or disabling certain features, such as the keyboard, while the user is driving

- **Maps for non-navigation apps:** Drawing maps for POI apps

*Note:* Navigation apps must take care of drawing and updating their maps, including drawing a light-themed or dark-themed map when instructed to do so

App developers take care of the following:

- **Voice input:** Processing recorded audio

- **User flows:** Creating customized sequences of templates that address critical user journeys

- **Metadata:** Providing metadata such as list items and locations to be pinned on maps

- **Branding elements:** Providing app iconography, imagery, and custom *accent colors* (with light and dark variants)

For AAOS, vehicle OEMs take care of:

- **Vehicle-specific styling:** *Customizing styles* to integrate with OEM branding and vehicle interiors
Designing your app
App design steps

Designing with the Android for Cars App Library involves the following steps, in general:

1. **Define user tasks**: Figure out which tasks are important for users to perform with your app. Focus on driving-related tasks presented in ways that limit driver distraction.

2. **Plan task flows**: Choose a sequence of templates to take users through the process of performing each task, planning the template order carefully. Strive to keep flows short (for examples, visit Sample flows), and consider how to incorporate interactive maps.

3. **Plan communications**: Consider where you might communicate with users by means of toasts, notifications, and navigation alerts, as well as by using message-oriented templates or voice input.

4. **Customize your app**: Customize the content of each template to reflect your users’ needs, and customize styling to reflect your app’s brand. For AAOS versions of your app, be aware that vehicle OEMs can adjust the styling to fit their vehicles.

**Note**: Throughout the design process, refer to the UX guidelines to make sure you are following UX requirements and recommendations.
Limiting driver distraction

To minimize distraction for drivers, keep these strategies in mind when designing your car app.

Limit information on each screen: To keep driver attention on the road as much as possible, the templates in the app library limit the amount of information that can appear on each screen while driving. Allowable numbers of actions, images, and other elements (such as amount of text, in some cases) are described for each template. For templates with list and grid items, the maximum allowed number of items varies by vehicle and can be retrieved via the ConstraintManager API.

Present only essential app content: For the car version of your app, focus on essential, driving-related content such as frequently-used locations, rather than the full content of your app.

Minimize attention needed for tasks: Simplify processes for drivers by keeping task flows short.

Provide shortcuts: Present saved user content early in task flows for quick access (for example, favorites or recently visited locations).

Minimize user input: When possible, present pre-selected options and defaults, so users can easily make choices while driving. Also, consider enabling voice input while driving.

Refresh content only when necessary: To minimize the need for users to look away from the road to see screen updates, the types of refreshes allowed on each template are carefully limited. Allowable refreshes (which do not affect the step count for your task flow) include updating routing information during a drive, providing dynamic search results, filtering or sorting lists, and other minimally distracting updates. To learn more about what is considered a step or refresh, see Step counts and refreshes. For an example of a template update too extensive to be considered a refresh, see Browse locations and start navigation. For technical information, visit Template refreshes.

For general information about limiting distraction in car UI design, visit Design foundations.
Keeping task flows short

A new task starts when a user does any of these:

- Lands on (or returns to) the app's landing template
- Chooses an action that opens another app
- Lands on the Navigation template (in navigation apps)

To minimize driver attention needed, keep task flows to 3 steps or fewer when possible. The limit is 5 steps, including the starting and ending steps. When flows are longer than 3 steps, consider providing shortcuts back to earlier steps.

**Flows involving purchases** should be 4 steps or fewer if possible.

**Refreshes** and certain **subflows** aren’t included in step totals.

When building your template sequence, also keep in mind the template-specific considerations described in [Template order in task flows](#).
Step counts and refreshes

To design task flows that don’t exceed the 5-step limit, it’s important to understand how steps are counted for each flow.

The step count is incremented whenever one of the following occurs:

- A new template is shown (unless it is a template usable only in a parked state, such as Sign-in or Long Message)
- The same template is shown with new content – unless the new content is simply a refresh of existing content

Refreshes do not add to the step count. For details about what counts as a refresh, see What is a refresh?

For more examples of how step counts would be incremented in specific task flows, see Sample flows.

**Note:** A task flow can have one added subflow without adding to the overall step count, as described in Adding subflows.
What is a refresh?

Refreshes don’t increment the step count. Types of refreshes include:

- **User-initiated refreshes**, as shown in the sample flow [Refresh a list with the refresh button](Note: in AAOS, vehicle OEMs can opt to hide the refresh button)

- **Other template-specific refreshes**, discussed below

In the Navigation template, all updates are considered template refreshes.

To find out what is considered a refresh for other templates, consult the Template Restrictions for those templates. For example, in the List Template Restrictions, you can see that changes to a toggle state are considered refreshes.

For more examples of refreshes, see [Sample flows](#)

In the example above, a toast is displayed after the user selects “Grant permissions,” directing the user to the phone for permission details. The toast display counts as a refresh and doesn’t add to the step count.
Adding subflows

A subflow involves stepping aside to take an action before returning to the same step. Subflows are useful to support actions such as filtering or sorting a list.

Even a task flow with the maximum number of steps (5) can have a subflow added, as shown at right, as long as the subflow is not the 5th step. While the subflow is included in the step count when a user launches it, it’s removed from the step count when the user returns to the main flow.

To minimize distraction, subflows should be limited to a single step, and there should be no more than 1 subflow per task flow.

Note: Upon return to a step from a subflow, the content in the template to which you are returning can be refreshed (for example, to apply a filter).
Template order in task flows

When creating task flows, keep the following template ordering considerations in mind:

- **Avoid List and Grid templates for step 5.** If the flow involves 5 steps, the 5th step must not be a list- or grid-based template (examples of acceptable templates include Navigation, Message, and Pane).

- **Don’t put 5 Lists or Grids in a row.** Apps should not create flows with 5 list-based or grid-based templates in a row, even if one of those templates is in a subflow.
Interactive maps

The ability for users to interact with maps via features such as zooming and panning is now provided for templates specific to navigation apps: Navigation, Route Preview, Map, and Place List (navigation). The Place List (map) template does not have an interactive map but does allow refresh of the list next to the map, as noted below.

Users can interact with maps using:

- **Touchscreen gestures** (such as swiping to pan)
- **Taps** on specific areas of the map, such as points of interest
- **Buttons on the map action strip**
- **A refresh button** that refreshes the information adjacent to the map (currently available only on the Place List templates)

Refreshing the content next to the map does not add to the step count for a task flow.
Communicating with users

When your app needs to communicate with users, choose the most appropriate of the following methods:

- **Message template**: Use the Message template when you want to show an icon or image with a brief message and up to 4 optional relevant actions (2 in template body and 2 in action strip).

- **Long Message template**: Use the Long Message template for long, scrollable messages to be read while the car is parked, with up to 4 relevant actions (2 in body, 2 in action strip). This template is useful for providing details about a destination or presenting legal text.

- **Toasts**: To help keep flows short, use toasts to inform users of an action when you don’t need to show a relevant template – for example, to provide a custom message when you tell a user to continue a flow on the phone when parked. (For technical details, visit Show toasts and Handle user input).

- **Voice**: To give users a way to communicate with your app using their voice, use the voice input feature.

- **Navigation alerts**: On the Navigation template, use navigation alerts to provide a brief message and optional actions without blocking the navigation route.

- **Notifications**: Use heads-up notifications (HUNs) to communicate highly important updates outside of the Navigation template. HUNs can include up to 2 actions, and they can deep-link to relevant parts of the app. However, be aware that vehicle OEMs can decide whether to display navigation HUNs in the AAOS version of your app.

For details about types of notifications used by navigation apps, refer to Navigation notifications: TBT and regular. For technical details about displaying notifications in Android for Cars, refer to Display notifications. And for additional details relevant to AAOS, visit Notifications on Android Automotive OS.
Voice input

Apps can access the car’s microphone to gather audio input for purposes such as creating their own in-app assistant.

Best practices:

- **Get permission first.** Make sure the user has granted permission for your app to access the car’s microphone (ideally before the drive starts).

- **Provide an entry point.** Give the user a way to start voice input, such as a microphone icon in the action strip. Then, wait for them to initiate the process.

- **Brand the experience.** When creating an in-app assistant, make clear that it’s an assistant specific to your app.

- **Stop when the user does.** When the user is done talking, stop recording.

**Note:** Processing the recording is up to the app. The recording is not saved in the car.

How it works: The user requests voice input (via the microphone icon in the action strip, in this case). An overlay appears, signalling that recording is in process. Users can stop the recording by dismissing the overlay. Or, they can stop talking, at which point the app should stop recording.

For a sample flow showing this process, see [Communicate with the app by voice](#).
Customizing app styling
Customizing app styling

This section provides an overview of how app developers and vehicle OEMs contribute to the visual design of apps created with the Android for Cars App Library.

Included are the following topics:

- Visual design customization overview
- Color customization details
- Choosing colors for your app
- App customization examples
- Vehicle OEM customization examples
# Visual design customization overview

While the app library determines the template layouts and default styling, app designers and vehicle OEMs both contribute to custom aspects of the visual design.

<table>
<thead>
<tr>
<th>Aspect of UI</th>
<th>What the library determines</th>
<th>What apps determine or customize</th>
<th>What vehicle OEMs can customize</th>
</tr>
</thead>
<tbody>
<tr>
<td>Images &amp; iconography</td>
<td>Iconography for standard elements, such as back button and loading spinner</td>
<td>Apps provide all images and iconography (see Material icons and Google Play icon specifications), except as noted at left</td>
<td></td>
</tr>
<tr>
<td>Layout, sizing &amp; shapes</td>
<td>Default layout, plus size and shapes of all elements (defaults are standard in Android Auto version of app)</td>
<td>Whether images and icons are “large” or “small”</td>
<td>Adjustments to size, shape, button locations and proportions of template elements in AAOS versions of app (or example, the exact sizing for “large” and “small” images and icons in their vehicles)</td>
</tr>
<tr>
<td>Typography &amp; text length</td>
<td>Font family and size in Android Auto version of app (see Typography)</td>
<td>Longer and shorter variants of text strings, in some cases, to accommodate differing amounts of space on different car screens</td>
<td>Font family and size in AAOS versions of app</td>
</tr>
<tr>
<td>Color</td>
<td>Default colors in Android Auto version of app (except those supplied by apps, noted at right)</td>
<td>Colors of place-list markers, some text elements, and some background colors (see Color customization details)</td>
<td>Adjustments to default &amp; app-supplied colors to as needed to blend with vehicle UIs in AAOS versions of app</td>
</tr>
</tbody>
</table>

*Material icons* and *Google Play icon specifications* indicate the usage of these libraries for icons and images.

*Font family and size in Android Auto version of app* and *variants of text strings* refer to the specific typography guidelines provided by Android for Car Apps.

*Color customization details* provide further information on color customization guidelines for apps.

*Font family and size in AAOS versions of app* indicate the usage of these guidelines for font in AAOS versions of the app.
Color customization details

Apps can provide colors for elements of certain templates, as noted below. For AAOS versions of your app, vehicle OEMs can make some adjustments.

What apps can customize:

- **Text color*** in secondary line of list rows (car maker controls color of primary line for AAOS)
- **Button text color**
- **Button background colors** (except on action strip and map action strip)
- **Place-list marker colors**
- **Routing-card elements**: background color, images, and color of duration value in trip estimate (within Navigation template guidelines)
- **Turn-by-turn notifications** (background color)

* For items marked with an asterisk at left, apps choose from 4 standard colors or up to 2 custom accents. For other customizable items, apps can use any color.

Examples of customized template components are shown in App customization examples and Vehicle OEM customization examples.
Choosing colors for your app

For most custom styling (other than the exceptions noted in Color customization details), apps have the following color options:

- **Provide up to 2 custom accent colors** (with light and dark variants, to be applied as appropriate by Android Auto, or by vehicle OEMs for AAOS versions of apps)
- **Choose from 4 standard Android for Cars colors** (current versions shown at right; these may change in the future)

Judicious use of color helps to focus the intent of a design. Be cautious about using colors when they don’t serve a function.

**Note:** For legibility while driving, make sure contrast between foreground and background colors meets contrast requirements of 4.5:1.
App customization examples

- **Gas stations**
  - ARCO: 0.3 miles, 4.2★
  - Shell: 0.5 miles, 4.4★

- **Navigation**
  - Stop Nav
  - Cancel

- **App Name**
  - Drive Home
  - Drive to Work
  - Recents

- **Distance**
  - 0.3 mi
  - Roy St (520)

- **Route Info**
  - 1120 112th Ave NE #100
  - 23 mins

- **Location**
  - 601 N 34th St, Seattle WA, 0.2 miles, 5 mins
  - 700 N 34th St, Seattle WA, 0.3 miles
  - 0.5 miles

android for cars

1.3-beta update
Vehicle OEM customization examples

These examples show additional style customizations a vehicle OEM might apply to the AAOS version of an app. While the color of the routing card comes from the app, vehicle OEMs customize the fonts, theming, and shapes for the routing card, buttons, and ETA card. They can also adjust button width, as shown in OEM customization of buttons.
OEM customization of buttons

These examples highlight how OEMs can customize button width, as well as color and shape, in the AAOS version of an app.

For the button the app designates as the primary button, OEMs can decide whether to use an app accent color or their own accent color. They can also choose whether to put the primary button on the left or on the right, to accommodate situations such as vehicles with right-hand drive.
UX guidelines

Global, app-specific, and template-specific usability guidelines
UX guidelines

Use the guidelines in this section to make sure your app meets UX design requirements and recommendations:

- Global guidelines
- Navigation app guidelines
- Template-specific guidelines

To fully understand the guidelines, make sure to read the definitions of MUST, SHOULD, and MAY at right.

Note: Be aware that Android app quality for cars specifies additional requirements beyond the design-related requirements discussed here. App developers are responsible for complying with all requirements specified there, as well as the ones discussed here.

MUST, SHOULD, and MAY

UX guidelines are expressed as instructions that you MUST, SHOULD, or MAY follow. At a high level, you can understand these labels as follows:

MUST = Required (enforced either in the API or in Android app quality for cars)

SHOULD = Recommended

MAY = Optional

For details, including a discussion of MUST NOT and SHOULD NOT, visit Meaning of Must, Should & May.
Global guidelines

Guidelines on this slide are global to your entire app. Navigation apps should also meet the Navigation app guidelines. To learn about other requirements that apply to your app, visit UX guidelines.

App developers:

- **MUST** keep task flows to 5 steps or fewer in length
- **MUST** get user permission to access the car microphone before recording audio for voice input
- **MUST** use the appropriate method to direct Android Auto users to the phone for actions that are not allowed while driving, instructing them to look at their phone screens only when it’s safe to do so (see relevant sample flow)
- **MUST NOT** end 5-step task flows with a list-based template (5th step must be one of these: Navigation, Message, or Pane)
- **SHOULD** keep task flows short (2 to 3 steps in length)
- **SHOULD** show content (or action strip buttons) for at least 8 seconds before removing that content in an auto-transition between steps
- **SHOULD** ask users to grant any necessary permissions when they first open the app
- **SHOULD** provide 2 accent colors, if possible, to better accommodate dark and light backgrounds
- **SHOULD** provide a back button or other exit mechanism in places where no other actions are available, such as loading screens and actionless Message and Pane templates
- **SHOULD** show useful content when opening a template, rather than an empty state with no options for users
- **SHOULD** provide shortcuts to earlier steps (for example, when task flows exceed 3 screens)
- **SHOULD** provide a user entry point, such as a microphone icon, if your app allows voice input (audio recording)
- **SHOULD** stop recording audio when the user stops providing the input

For additional global guidelines, see the next slide.
Global guidelines, continued

Guidelines on this slide are global to your entire app. Navigation apps should also meet the Navigation app guidelines. To learn about other requirements that apply to your app, visit UX guidelines.

App developers:

- **SHOULD** refresh content only for the purposes noted in limiting driver distraction or where explicitly permitted in guidance for specific templates or app types
- **SHOULD NOT** use auto-transitions to complete tasks without user action (that is, do not use them back-to-back)
- **SHOULD NOT** create buttons with states, such as toggles, in places where actions are supported (toggles are supported only in list rows)
- **MAY** update a list row or grid item’s image, icon, or secondary text to reflect changes

**Purchase flows** only:

- **SHOULD** provide shortcuts wherever possible, such as allowing users to repeat previous transactions (“book again”)
- **MUST NOT** allow users to set up payment methods
- **MUST NOT** ask users to commit to recurring payments
- **MUST NOT** allow users to select multiple items for purchase in a single flow

**POI apps** only:

- **SHOULD** provide a way to launch a navigation app in order to navigate to the point of interest

For additional global guidelines, see the preceding slide.
Navigation app guidelines

Guidelines on this slide pertain to navigation apps. To learn about other requirements that apply to your app, visit [UX guidelines](#).

Nav app developers:

- **MUST** make sure all visual information drawn on maps (such as speed information and route labeling) meets [contrast requirements](#).
- **MUST** draw only map content and drive-related content on the surface of the template.
- **MUST** draw a light-themed or dark-themed map when instructed to do so.
- **SHOULD** make sure text drawn on maps uses a font size of 24dp or larger unless it is paired with a visual element (such as a route or road) or is relatively static on the display.
- **SHOULD** clearly indicate if a task will update the route.

- **SHOULD** meet or exceed minimum size of 36 x 36dp for images, icons, and map markers.
- **SHOULD** use turn-by-turn (TBT) notifications to surface directions when a user is completing a task outside of the Navigation template during active navigation (as shown in Navigation notifications: TBT and regular).
- **SHOULD NOT** create multi-stop journeys, since templates are not optimized for this type of interactivity.
- **SHOULD** refresh duration and distance values during the drive.
- **MAY** use navigation alerts or HUNs to alert users about general navigation-related updates (in addition to turn-by-turn directions), such as traffic ahead.
- **MAY** customize background color of TBT notifications.
- **MAY** use animations when they aid in driving.
Template-specific guidelines

Use the template-specific guidelines to make sure each template in your task flows meets UX requirements and recommendations:

- List template guidelines
- Grid template guidelines
- Sign-in template guidelines
- Message template guidelines
- Long Message template guidelines
- Search template guidelines
- Pane template guidelines
- Map template guidelines
- Place List (map) template guidelines

- Place List (navigation) template guidelines
- Route Preview template guidelines
- Navigation template guidelines

To learn about other requirements that apply to your app, visit UX guidelines.
Templates
### Template overview

The Android for Cars App Library includes:

- 7 general-purpose templates that all apps can use
- 1 template designed specifically for point-of-interest (POI) apps
- 4 templates designed for navigation apps that draw their own maps*

* In all map-based templates for navigation apps (see table at right), the app draws the map.

<table>
<thead>
<tr>
<th>Template</th>
<th>What it shows</th>
<th>For navigation apps</th>
<th>For POI apps</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>List or Grid</strong></td>
<td>Information items in a list or grid layout</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td><strong>Sign-in</strong></td>
<td>Options for signing in to the app</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td><strong>Message or Long Message</strong></td>
<td>Brief or longer full-screen message and relevant actions</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td><strong>Search</strong></td>
<td>Search bar and results list</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td><strong>Pane</strong></td>
<td>Detailed information and prominent actions, with optional image</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td><strong>Map</strong></td>
<td>Compact version of list or pane view (information &amp; actions) next to a map*</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td><strong>Place List (map)</strong></td>
<td>List of locations or other items next to a map drawn by the app*</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td><strong>Place List (navigation)</strong></td>
<td>List of locations or other items next to a map drawn by app library</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td><strong>Place List (navigation)</strong></td>
<td>List of locations or other items next to a map drawn by the app*</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td><strong>Route Preview</strong></td>
<td>List of available routes for a selected destination, next to a map*</td>
<td>□</td>
<td>□</td>
</tr>
<tr>
<td><strong>Navigation</strong></td>
<td>Full-screen map* with action strip and optional routing card</td>
<td>□</td>
<td>□</td>
</tr>
</tbody>
</table>
List template

Purpose: Presents information items in a list layout. Lists may be separated into sections via sublists. For a version of this template with a map, navigation apps can use the Map template.

Includes:

- Header with optional action strip
- List items (where the number of items and the amount of text in a row can vary as described below)

Number of list items. The number of items allowed to be shown depends on the vehicle. To retrieve the list row limit for a given vehicle, use the ConstraintManager API.

Amount of text per item. Secondary text in list rows can be longer than 2 rows when parked.

For more about what rows can include in this template see Row options for List template only and the examples in Sample flows.
List template examples

Uniform list of locations (Android Auto example)

List with sections and toggle rows (Android Auto example)

Selectable list with radio buttons – no sections or non-selectable rows allowed (AAOS example)
More list text when parked

The amount of secondary text allowed in each list row varies depending on whether the car is parked or driving. To minimize distraction, text is truncated to 2 lines while driving. Any content intended to be read while driving should be displayed at the beginning of the secondary text.

For a sample flow illustrating the transition from parked to driving, see View more list text while parked.
List template guidelines

To learn about other requirements that apply to your app, visit UX guidelines.

App developers:

- **MUST** include a section header when sections are present
- **MUST NOT** mix selectable list rows (rows with radio buttons) with other types of rows or separate them with sections
- **SHOULD** present default selections on selectable lists
- **SHOULD** have an action associated with each list item (information-only rows are not recommended)
- **SHOULD** place content in secondary text that is intended to be read while driving near the beginning, to avoid truncation

- **MAY** divide list content into sections
- **MAY** mix toggle rows with other rows as needed
- **MAY** update row text and image or icon asset when user changes toggle state
Grid template

Purpose: Presents items in a grid layout. This template is useful when users rely primarily on images to make their selections.

Includes:

- Header with optional action strip
- Grid items,* each containing an icon or a large-size image
- Primary text for each grid item (mandatory)
- Secondary text for each grid item (optional)

* There is a limit on the number of grid items allowed to be shown, but the limit will not be less than 6 and may be higher in some vehicles. To retrieve the item limit for a given vehicle, use the [ConstraintManager](#) API.
Grid template examples

Grid in which some items have secondary text (Android Auto example)

Grid with one item’s icon temporarily replaced by a loading spinner and another item’s primary text truncated because its length exceeds the available space (AAOS example)
Grid template guidelines

To learn about other requirements that apply to your app, visit UX guidelines.

App developers:

- **SHOULD** limit length of primary and secondary text strings to avoid truncation

- **SHOULD** have an action associated with each grid item (information-only items are not recommended)

- **SHOULD** clearly indicate item state (for grid items that have multiple states, such as selected and unselected) by variations in image, icon, or text

- **MAY** show a loading spinner in place of the icon or image for a grid item when an action associated with the item is in progress
Message template

**Purpose:** Presents a brief message and optional relevant actions. This template is useful for communicating error messages, permission prompts, and other information about UI states.

**Includes:**
- **Header** with optional **action strip**
- Up to 2 lines of wrapping text
- Image, icon, or loading spinner (optional)
- Up to 2 **buttons** in template body (optional), where one can be designated as **primary**

**Note:** This template is intended for quick messages, with related actions as secondary. To display a longer message, use the **Long Message template**. To display more detailed information with prominent actions, use the **Pane template**.
Message template examples

Error message with icon (Android Auto example)

Message with two actions (AAOS example)
Message template guidelines

To learn about other requirements that apply to your app, visit UX guidelines.

App developers:

- **MUST** include message text
- **SHOULD** designate a primary action when providing 2 actions
- **SHOULD** place the primary action closest to the driver (on the left for left-hand-drive vehicles) when there are 2 actions
- **MAY** include an image or icon asset
- **MAY** include up to 2 actions
- **MAY** use this template to prompt users about app permissions and open related flows on phone when parked (as shown in Grant permissions on phone)
Long Message template

Purpose: Presents a long message to be read while the car is parked, with optional relevant actions. This template is useful for providing details about a destination or for presenting legal text, such as terms of service or a privacy policy, during a sign-in process.

Includes:

- **Header** with optional action strip
- Unlimited lines of wrapping text (scrollable)
- Up to 2 buttons in template body (optional), where one can be designated as primary

The header remains in place when text scrolls up, keeping the exit affordance and action strip available.

Note: This template displays its contents only when parked (see examples) and does not increase the step count.
Long Message template examples – Parked and driving states

When the car is parked, this template can show a detailed message, such as a privacy policy, or terms of service for the user to accept when signing in to the app (Android Auto example).

When the user is driving, the long message is not shown, to prevent driver distraction. For these situations, it’s helpful to provide a button with an alternative option, such as skipping sign-in and using the app in guest mode. (Android Auto example)
Long Message template guidelines

To learn about other requirements that apply to your app, visit UX guidelines.

App developers:

- **MUST** include message text
- **SHOULD** designate a primary action when providing 2 actions
- **SHOULD** place the primary action closest to the driver (on the left for left-hand-drive vehicles) when there are 2 actions
- **MAY** include up to 2 actions
Sign-in template

**Purpose:** Presents options for signing in to the app while parked. Supports 4 possible sign-in methods.

**Includes:**
- Header with optional action strip
- Up to 2 lines of primary text (optional)
- Primary sign-in method
- Additional text, such as disclaimers and links to terms of service (optional)
- Up to 2 buttons (optional)

**Note:** This template displays its contents only when parked (see examples) and does not increase the step count.
Sign-in methods

The Sign-in template supports 4 sign-in methods: Provider sign-in, Username/password, PIN code, and QR code (see next slide)

Provider sign-in method:
This method lets users sign in easily via a provider, with no input required. In this example, Google is the provider for the primary sign-in option, with PIN code sign-in offered as a secondary option. (Android Auto example)

Username/password method:
This method lets users enter authentication information in a single, mandatory form field. Currently, this field can be used for entering a username or password. In this example, the other two sign-in methods are offered as secondary options. (Android Auto example)
**Sign-in methods, continued**

**PIN code method:**
This method displays a mandatory PIN code (up to 12 characters in length) provided by the app, as well as instructions for where the user should enter it. The code can be refreshed as needed if it times out. (Android Auto example)

**QR code method:**
This method displays a mandatory QR code provided by the app. The code can be refreshed as needed if it times out. (Android Auto example)
When the car is parked, the user can access the keyboard to type a username or password. (Android Auto example)

When the user is driving, the sign-in content is not shown, to prevent driver distraction. For these situations, it's helpful to provide a button with an alternative option, such as skipping sign-in and using the app in guest mode. (Android Auto example)
Sign-in template guidelines

To learn about other requirements that apply to your app, visit UX guidelines.

App developers:

- **MUST** include a sign-in method when using this template
- **SHOULD** use input fields only for user sign-in, not for collecting other types of user input
- **SHOULD** prioritize shortest flow (fewest clicks)
- **SHOULD** prioritize most popular method
Search template

**Purpose:** Presents a search bar, keyboard, and results list for users to perform searches, such as searching for destinations. During drives, users can’t access the keyboard, but they can use the search template to see past or suggested search results.

**Includes:**
- Search bar [header](#) with optional [action strip](#)
- List [rows](#) for search results (within limits*)
- Keyboard (when parked), which apps can collapse or expand

* The number of list rows allowed to be shown depends on the vehicle. To retrieve the list row limit for a given vehicle, use the [ConstraintManager](#) API.
Search template— parked and driving examples

In a parked state, the keyboard is available for typing search terms
(Android Auto example)

In a driving state, the keyboard is unavailable and the search bar is
disabled, but users can choose from past or suggested results
shown on the list (Android Auto example)
Search template guidelines

To learn about other requirements that apply to your app, visit UX guidelines.

App developers:

● **MUST** update the list when a user enters keywords

● **SHOULD** provide dynamic content (screen refresh) only to show search results during user input

● **SHOULD** either show content or launch a keyboard (if there is no content to show) when opening the template

● **MAY** display the keyboard as either expanded or collapsed when a user opens the template in a parked state (the keyboard is unavailable during the driving state)

● **MAY** provide hint text on the search bar

● **MAY** display a default list of past results or other relevant content
**Pane template**

**Purpose:** Presents detailed information and prominent actions. For easy scanning, actions and information rows are limited to 4 each. This template is useful for presenting non-editable metadata, such as location and reservation details, and for taking action based on data. For a version of this template with a map and no image, navigation apps can use the [Map template](#).

**Includes:**
- **Header** with optional [action strip](#)
- Up to 2 [buttons](#), where one can be designated as [primary](#) (optional; apps can use action strip buttons instead)
- Up to 4 non-actionable [rows](#) (1 row is mandatory)
- Optional large image (see example on next slide)

**Note:** This template highlights actions and information. For quick messages with less detail, use the [Message template](#). For long messages, use the [Long Message](#) template.
Pane template examples

Location details for a parking app, with 2 related actions (Android Auto example)

Location details for a charging app, with a large image that helps drivers identify the location, and with no actions except in the action strip (Android Auto example)
Pane template guidelines

To learn about other requirements that apply to your app, visit UX guidelines.

App developers:

- **MUST** include at least one row of information
- **SHOULD** designate a primary action when providing 2 actions
- **SHOULD** make navigation the primary action, when it’s included as one of the actions
- **MAY** include up to 4 rows of information and 2 actions
Map template

**Purpose:** Presents a compact version of a list (as in the List Template) or of a pane (detailed information with prominent actions, as in the Pane Template) next to a map.

**Includes:**
- **Header** with optional **action strip**
- List **rows** with optional small images
- OR list **rows** without images, augmented by up to 2 optional **buttons**, where one can be designated as **primary** (as as in the Pane Template)
- Base map (full-screen) drawn by the app
- Optional **map action strip** with up to 4 buttons for map interactivity

**Note:** This template is for navigation apps only. To present similar information with no map, POI apps can use the List Template or the Pane template.
Map template examples

Map template with a list of items, next to a map showing the item locations (Android Auto example)

Map template with a pane view showing the details of a parking location with related actions, next to a map showing the parking lot location (Android Auto example)
Map template guidelines

To learn about other requirements that apply to your app, visit UX guidelines.

App developers:

● MUST include at least one row of information if using a pane view

● SHOULD designate a primary action when providing 2 actions

● SHOULD make navigation the primary action, when it’s included as one of the actions

● MAY include up to 4 rows of information and 2 actions (in a pane view) OR up to the number of list rows allowed for the vehicle* (in a list view)

* To retrieve the list row limit for a given vehicle, use the ConstraintManager API.
Place List (map) template

Purpose: Presents an ordered list of locations (or containers for sublists), overlaid on a map provided by app library.

Includes:
- Header (in card) with optional refresh button for users to request a list update (doesn’t add to step count)
- Action strip (optional)
- Base map (full-screen, not drawn by apps)
- List rows (within limits*)
- Markers

* The number of list rows allowed to be shown depends on the vehicle. To retrieve the list row limit for a given vehicle, use the ConstraintManager API.

Note: This template is for POI apps. An alternate version, Place List (navigation), is provided for navigation apps.
Place List (map) template examples

Location list with corresponding map markers and anchor marker (Android Auto example)

List with carets pointing to sublists (AAOS example)
Markers

Markers are used to link list items with locations on the map or to identify an anchor location. Markers can be designated as tappable (as can any area on a map), so that users can tap a marker to trigger an action such as displaying information about that marker.

Types of markers:

1. **Map marker**: On map, labeled with one of the following: text (up to 3 letters), an icon, or an image

2. **List marker** (not shown): On list, marker that corresponds to map marker, with matching metadata and image or icon asset

3. **Anchor marker** (optional): On map, used to show center of search area

Apps can customize the background color of markers with any color. The color used for the map marker is applied to the list marker.
Place List (map) template guidelines

To learn about other requirements that apply to your app, visit UX guidelines.

App developers:

- **MUST** show duration or distance for each list item (except for container items)
- **MUST** associate an action with each list row (information-only rows are not allowed)
- **MUST** display only locations that are appropriate to the app type (parking spots for parking apps, charging stations for charging apps)
- **SHOULD** include at least one location or browsable list item (container) on the list

- **SHOULD** show a corresponding marker on the map for each location on the list
- **SHOULD** limit locations to those that are closest or most relevant
- **SHOULD** consider supporting content refresh for the list, so users can update it after driving out of range of the original list
Place List (navigation) template

Purpose: Presents an ordered list of locations (or containers for sublists), overlaid on a map drawn by the app. The app should draw markers to link list items with map locations.

Includes:

- **Header** (in card) with optional refresh button for users to request a list update (doesn’t add to step count)

- **Action strip** (optional)

- Base map (full-screen) and markers drawn by app

- Optional **map action strip** with up to 4 buttons for map interactivity

- List **rows** (within limits*)

* The number of list rows allowed to be shown depends on the vehicle. To retrieve the list row limit for a given vehicle, use the **ConstraintManager** API.

**Note:** This template is for navigation apps only. An alternate version, Place List (map), is provided for POI apps.
Place List (navigation) template examples

Location list (Android Auto example)

For a sample flow showing how the refresh button might be used, see Refresh a list for a new map location.
Place List (navigation) template guidelines

To learn about other requirements that apply to your app, visit UX guidelines.

App developers:

- **MUST** show duration or distance for each list item (except for container items)
- **MUST** associate an action with each list row (information-only rows are not allowed)
- **SHOULD** include at least one location or browsable (container)
- **SHOULD** include only information relevant to app capabilities (not “favorite friends”)
- **SHOULD** limit locations to those that are closest or most relevant

- **SHOULD** show a corresponding marker on map for each location on list
- **SHOULD** use a font size of at least 24dp Roboto or equivalent for map markers
- **SHOULD** consider supporting content refresh for the list when supporting map interactions
Route Preview template

**Purpose:** Presents up to 3 available routes for a selected destination, next to a map. Route information must include either duration or distance and may also include estimated travel time. As a user selects a route to preview, the app updates the map with a visual representation of the route.

**Includes:**
- Header (in card) and separate action strip (optional)
- List rows with route information
- Base map (full-screen) drawn by the app
- Optional map action strip with up to 4 buttons for map interactivity

**Note:** This template is for navigation apps only.
Route Preview template examples

Two routes, linked to map with numbered markers
(Android Auto example)

Three routes, with estimated trip duration shown on map
(AAOS example)
Route Preview template guidelines

To learn about other requirements that apply to your app, visit UX guidelines.

App developers:

- **MUST** present at least 1 route on this template
- **MUST** have 1 route selected by default
- **MUST** show either duration or distance for routes
- **MUST NOT** present more than 3 routes
- **SHOULD** highlight the relevant route on the map when a user makes a selection
- **MAY** customize “Continue to route” text for highlighted route
Navigation template

Purpose: Presents a base map and optional routing information. When a user is driving without text-based turn-by-turn directions, apps can show a full-screen map updated in real time. During active navigation, apps can surface optional cards with maneuvers and surface details, as well as navigation alerts.

Includes:
- Full-screen base map drawn by app
- Routing card (optional) with upcoming maneuvers
- Travel estimate card (optional) with estimated time of arrival (ETA), time to destination, and remaining distance (or an alternate information display with custom text and icon options)
- Action strip with up to 4 app actions, visible only as described in Visibility of action strips
- Optional map action strip with up to 4 buttons for map interactivity

Note: This template is for navigation apps only.
Navigation template examples

Full-screen map when navigation and map interactivity are not occurring (Android Auto example)

Routing card with travel estimate, action strip (at top), and map action strip (at bottom right) during active navigation – learn more in Routing card details (AAOS example)
Routing card details

When the routing card is in routing state (as opposed to message state), it shows the following information:

1. **Current step**: includes icon (typically a direction arrow), distance, and cue text (which may include image spans such as route markers)

2. **Lane guidance** (optional): shown either as simple lane-assist images or as a larger junction image (flexible size with maximum height of 200dp)

3. **Next step** (optional): includes icon and cue, and can appear only at the bottom of a routing card that doesn’t include a junction image

Another option in routing state is for the routing card to display a spinner animation (not shown here) to indicate transient states such as loading, calculating, or rerouting.

In some circumstances, routing information can instead be displayed in a floating navigation bar, as shown in Add a stop while driving.
**Message state of routing card**

When the routing card is in message state, it displays a message instead of routing directions. The message can be used to convey situations such as route failures and arrival at a destination.

In the message state, the routing card can include:

- A routing-related message of at least one line in length, with an optional second line
- An image or icon (optional)
Navigation notifications: TBT and regular

**TBT notifications:** When an app provides text-based turn-by-turn directions, it must also trigger TBT notifications. These notifications are used to surface TBT directions outside of the Navigation template. Apps can customize TBT notification background color for greater visibility.

**Regular notifications:** To communicate other navigation-related messages, such as changes in route settings, navigation apps can also send regular (non-TBT) notifications (as shown here) or use navigation alerts. These can appear even when the routing card is displayed.

**Note:** For other important considerations regarding notifications and alerts, visit Communicating with users.
Navigation alerts

Navigation alerts provide a brief, temporary message and optional actions in a format that doesn’t block the navigation route. The content should be simple and relevant to the navigation task. For example, the alert might describe a change in traffic conditions or ask if the driver can pick up a customer.

Each alert includes:
- Title and optional subtitle
- Icon (optional)
- Progress indicator – either a bar or (optionally) built into a timed button
- Up to 2 buttons, where a button can be designated as primary or as a timed button (with a progress indicator, as shown at right)

Can be dismissed by any of the following:
- User selection of any action
- Time-out after X seconds (configurable)
- App dismissal without user action

Note: Do not use navigation alerts to show primary navigation information, such as upcoming turns or arrival time. For those types of information, use the routing card or trip estimate card.

For sample flow using a navigation alert, see Respond to a navigation alert.
Navigation template guidelines (MUST & SHOULD)

To learn about other requirements that apply to your app, visit UX guidelines and the MAY guidelines for this template.

App developers:

- **MUST** show at least 1 maneuver on a routing card
- **MUST** include at least one action button on the action strip to enable user flows
- **MUST** include a pan button in the map action strip if the app supports panning gestures
- **SHOULD** include only buttons related to map interactivity on the map action strip (for example, compass, recentering, or 3-D mode)
- **SHOULD** include a button to end navigation when providing turn-by-turn directions
- **SHOULD** use symbols that are standardized or consistent with international or country-specific symbols
- **SHOULD** use the junction image only to show content relevant to navigation, spanning card width with image
- **SHOULD** provide lane images with transparent backgrounds to blend with routing-card background
- **SHOULD** use alerts only for non-distracting information relevant to the current navigation task
Navigation template guidelines (MAY)

To learn about other requirements that apply to your app, visit UX guidelines and the MUST & SHOULD guidelines for this template.

App developers:

- MAY display short supporting text under a lane (Roboto 24 recommended) and lower contrast ratio for non-highlighted lanes
- MAY show 2 maneuvers on a routing card when they occur in rapid succession
- MAY include images such as route markers in routing-card text (current step and next step)
- MAY show a full map when user is driving without text-based turn-by-turn directions or is in free-drive mode
- MAY choose to show or hide routing card and trip estimate components as needed
- MAY draw driving-related details and alerts on the map, such as current speed, speed limit, and camera ahead
- MAY customize routing-card background color and change it during the navigation session to reflect road type or other relevant conditions
Sample flows
Overview of sample flows

The following sample flows provide examples of common app scenarios. Most template sequences are the same on Android Auto and AAOS. However, permission template sequences differ between systems: Android Auto permissions are on the phone, while AAOS permissions are in the car.

Many of these sample flows end with the Navigation template, which is only available for navigation apps. Other apps can use versions of these flows by switching to a navigation app as the last step in the flow. See the Access details and start navigation and Purchase using existing payment method flows.

Permissions and sign-in

- Grant permissions on phone (Android Auto)
- Grant permissions over car screen (AAOS)
- Sign into the app while parked

Navigation

- Navigate to a saved location
- Browse locations and start navigation
- Access location details and start navigation
- Search using past results while driving
- Respond to a navigation alert or to a [NEW] timed alert
- [NEW] Add a stop while driving
- Arrive at destination

Other common scenarios

- Purchase using existing payment method
- Communicate with the app by voice
- [NEW] View more list text while parked
- Refresh a list with the refresh button

To learn more about refreshes, review What is a refresh?

Each flow describes when the step counter is incremented. For context, see Step counts and refreshes.
Permissions and sign-in flows
Grant permissions on phone (Android Auto)

When users try to open your app and lack necessary permissions, use the Message template to tell them they need to grant permissions. In Android Auto, if you have used the method described in Request Permissions, the permissions dialog will open on the phone, as long as the user is not driving (for technical details, visit Handle user input). In this case, provide a toast directing the user to the phone. Then, after permissions are granted, refresh the car screen so the user doesn’t return to the Message template.

Step 1 – Landing template (not shown)

Step 2a – Message template: The user selects the option to grant app permissions

Step 2b – Message template with toast (refresh – doesn’t add to step count): A toast message appears, directing the user to the phone (except if the user is driving – in that case, the toast says the capability is not available while driving)

On phone: The user grants permissions on the phone

Step 1 of new task – Place List template: The app returns to the landing template
Grant permissions over car screen (AAOS)

In AAOS, the permissions flow is the same as on Android Auto, except the user sees permission details on the car screen instead of the phone.

Step 1 – Landing template (not shown)

1. The user selects the option to grant app permissions

Step 2a – Message template:

1. The user sees a system permissions dialog on the car screen and grants permissions

Step 2b – System dialog on Message template

(refresh – doesn’t add to step count):

1. The user sees a system permissions dialog on the car screen and grants permissions

Step 1 of new task – Place List template:

1. The app returns to the landing template
Sign in to the app while parked

Use the Sign-in template to prompt the user to sign in with a PIN code or one of the other 3 sign-in methods. Add the Long Message template to provide details such as text of the Privacy Policy or Terms of Service. These templates are both parked-only templates, so they don’t increment the step count.

Note: This example shows how the templates would look in Android Auto.

- **Step 1a – Landing template (not shown)**
  - On phone: The user sees a code that needs to be entered on the phone or at a specified URL

- **Step 1b – Sign-in template (parked-only template)**
  - On phone: The user enters the code

- **Step 1c – Long message template (parked-only template)**
  - On phone: The user reads the app’s privacy policy (if required)

- **Step 1 of new task – Place List template**
  - On phone: The app returns to the landing template
Navigation flows
Navigate to a saved location

Presenting location shortcuts in the landing screen of an app is a good way to keep task flows short. In the landing screen, if you include shortcuts such as “Home” and “Work,” users can start navigation just by tapping a shortcut.

Note: This example gives an idea of how the templates might look on AAOS.

Step 1 – Place List template (landing template):
The user selects a shortcut to the desired location

Step 1 of new task – Navigation template:
Navigation begins
Browse locations and start navigation

Organizing multiple saved locations under submenus, such as “Recents,” can help users quickly browse and find what they want. In this example, even with the additional step of selecting a route, the task flow from start to finish remains well within the maximum of 5 steps.

**Note:** This example shows how the templates would look in Android Auto.

**Step 1 – Place List template** (landing template):
The user selects a submenu

**Step 2 – Place List template** (location list – new content, not a refresh):
The user selects a location

**Step 3 – Route Preview template:**
The user selects a route

**Step 1 of new task – Navigation template:**
Navigation begins
Access location details and start navigation

When users need more detailed information about a location, the Pane template lets you present that information with buttons for the users to take action. In this example, because the app is not a navigation app, the Navigate button opens a separate navigation app, ending the task flow.

Note: This example shows how the templates would look in Android Auto.

Step 1 – Place List template (landing template): The user selects a submenu

Step 2 – Place List template (location list – new content, not a refresh): The user selects a location from the submenu

Step 3 – Map Pane template: The user checks out details about the location and decides to navigate there

Switch to navigation app (new task): A separate navigation app opens and routing begins
Search using past results while driving

When users are driving, they’re restricted from using the keyboard to search. However, you can still present past search results (locations) or keywords (such as “coffee”) on the Search template. Users can then simply select the result to proceed toward navigation.

Note: This example shows how the templates would look in Android Auto.

Step 1 – Place List template (landing template): The user selects the Search button from the action strip.

Step 2 – Search template (disabled state): The user selects a past search result.

Step 3 – Route Preview template: The user selects a route.

Step 1 of new task – Navigation template: Navigation begins.
Respond to a navigation alert

On the Navigation template, you can use a navigation alert to let users know about incidents such as traffic events and take actions in response. Notification alerts appear in the spot normally used for the ETA, so they don’t block the navigation route. They allow users to make a simple choice about how to respond to the alert.

Step 1a – Navigation template:
While the user is navigating, the app learns about a traffic event ahead on the route

Step 1b – Navigation template (refresh – doesn’t add to step count):
A navigation alert appears in place of the ETA to warn the user and offer an option to reroute, while a progress indicator tracks the time until automatic dismissal of the alert

Step 1c – Navigation template (another refresh):
The alert gets dismissed after it times out
[NEW] Respond to a timed alert

You can use a timed alert to let users know how much time they have to respond before a default action is chosen. To create a timed alert message, use the Pane view within the Map Template, with a timed button for the default action,

**Step 1 – Navigation template:**
The user makes a stop during active navigation.

**Step 2 – Map template**
When the user restarts the car, the shading on the timed "Resume" button tracks the time until the button action will automatically occur.
[NEW] Add a stop while driving (and retain turn info in the floating navigation bar)

You can now transition from the Navigation template to the Map template to show a compact list of places without losing the turn information. The turn information on the routing card is condensed into a floating navigation bar that moves to the side and remains visible.

Step 1 – Navigation template:
The user taps an action to add a stop during an active route

Step 2a – Map template
A compact list appears, showing the available options to add a stop, while the turn information shrinks into a floating navigation bar and moves to the side

Step 2b – Map template (refresh – doesn’t add to step count):
The user taps the desired stop

Step 3 – Navigation template:
The stop is added to the route and the app informs the user with a navigation alert
Arrive at a destination

On the Navigation template, you can indicate arrival by switching the routing card from routing state to message state and displaying an arrival message (this is a refresh, as are all updates to the navigation template). Then, you can use an auto-transition to again refresh the screen and show the map-only version of the template.

Note: This example gives an idea of how the templates might look on AAOS.

Step 1a – Navigation template (routing state):
The routing card directs the user toward the destination

Step 1b – Navigation template (message state – refresh):
A message confirms the user’s arrival at the destination

Step 1c – Navigation template (map only – refresh):
When the app decides it’s time (after at least 8 seconds), the message disappears and the user sees just the map
Other common scenarios
Purchase using existing payment method

Flows that involve purchases should be as simple and short as possible, to minimize attention needed from drivers. Be sure to follow the guidelines for purchase flows.

Note: This example shows how the templates would look in Android Auto.
Communicate with the app by voice

To support building features such as an in-app digital assistant, the Android for Cars App Library now lets you use the car’s microphone to record voice input from the driver. When recording is in progress, an indicator appears onscreen. The recording is sent directly to the app (not saved within the library) for processing and follow-up actions.

Step 1a – Navigation template:
The user taps the app’s mic button (on the action strip) to start voice input.

Step 1b – Navigation template
(refresh – doesn’t add to step count):
While the user is talking, a visual indicator signals that recording is in progress.

Step 1c – Navigation template
(another refresh):
A toast message tells the user that the app has understood and responded to the user’s spoken instructions.
**View more list text while parked**

On the List template, you can now show more content in the secondary text while the vehicle is in a parked state. When the user starts driving, the template will truncate the secondary text to 2 lines to minimize distraction.

**Step 1a – List template (parked state):** While parked, the user reads the full content in the list. Secondary text can be more than 2 lines.

**Note:** The definition of “parked state” is controlled by the vehicle OEM.

**Step 1b – List template (driving state) (refresh – doesn’t add to step count):** When the user starts driving, the template is refreshed and the secondary text for each list item is truncated to 2 lines. A toast provides context during the first transition from parked to driving in this template.

**Step 1c – List template (driving state) (refresh – doesn’t add to step count):** The toast goes away after a few seconds. If the user stops driving at this step, the template goes back to the expanded version (Step 1a).
Refresh a list with the refresh button

The optional refresh button on the Place List templates lets users refresh the list content to be relevant to a new area of the map. (Note: vehicle OEMs can opt to hide this button on AAOS.) The new area might appear when a user pans to it, as shown below, or when the user drives out of the area referenced by the initial list content. The refresh button lets users update the list to apply to the new map area.

Note: For information on app-initiated content refresh, see Step counts and refreshes.

Step 1a – Place List (nav) template: The user pans on the map to look for places in another area

Step 1b – Place List (nav) template (refresh – doesn’t add to step count): After panning to a new area on the map, the user presses the refresh button

Step 1c – Place List (nav) template (another refresh): The app displays a loading state while sending new content that is based on the new coordinates showing on the map

Step 1d – Place List (nav) template (another refresh): The app displays new list content relevant to the new map coordinates
Components
Component overview

This section describes the following components, which are used in multiple templates:

- **Action strip**
- **Map action strip**
- **Button**
- **Header**
- **Row**
**Action strip**

**Used in:** All templates (sometimes in the header)

**Includes:**
- Action buttons (up to 2 – except on templates with maps, which allow up to 4)

Only 1 label button (with label and optional icon) is allowed per template. Button order is specified by the app.

**Guidance:**
Use action strips for secondary or tertiary actions, rather than primary actions – except on the Navigation template. For details about when the action strip is displayed on the Navigation template and when it’s hidden, refer to Visibility of action strips.

**Note:** Templates for navigation apps can also have a map action strip with map interactivity buttons.
Action strip on templates with maps

On templates with maps, the action strip can include up to 4 buttons, as shown in these examples.

Android Auto examples

AAOS example
Map action strip

**Used in:** All templates that include a map, except the Place List (map) template, which is not intended for navigation apps

**Can include up to 4 buttons, such as** (in any order):
- [Pan mode](#) *(required for apps that support user panning)*
- Recenter
- Zoom-in/zoom-out

**Purpose:** Gives users access to map interactivity features. While users can access these features via gestures on touchscreens, they need buttons to access interactivity features on screens with rotary and touchpad inputs. Buttons also add helpful visual clues.

**Behavior:** Like the action strip, the map action strip disappears if 10 seconds go by without user interaction, as described in [Visibility of action strips](#).

**Guidance:** When using the map action strip, be sure not to draw other visual elements in the lower-right corner of the map. On short screens, these elements might end up covering the map action strip.
Pan mode

In pan mode, only the full-screen map and map action strip buttons are visible. Pan mode is used for rotary and touchpad inputs.

For the pan button, apps can provide 2 icons: one for entering pan mode, and one for exiting. The icon for exiting pan mode should clearly indicate that it provides a way to exit. For example, this button can show an “X.”

Note: Apps must include a pan button in the map action strip in order to enable user panning on all screen types, even though the pan button does not display on touch screens. Including the pan button causes touch gestures for map interactivity to be forwarded to the app.
Visibility of action strips

The app library generally takes care of showing and hiding the action strip and map action strip in the map-based templates based on user interactions, as shown below. **Exceptions:** 1) Apps can flag actions in either action strip as persistent to keep them from disappearing. 2) The app library may hide the action strip on small screens after 10 seconds even when there is rotary focus, to minimize clutter.

When a map-based template opens, the action strip and map action strip are visible. If 10 seconds go by with no user interaction, and if the action strip and map action strip do not have user focus for rotary input, they disappear. (Exceptions are noted above.)
**Button**

*Used in:* Pane, Message, Long Message, and Sign-in templates, or on the action strip in any template.

*Can include:*  
- Icon only  
- Label only  
- Icon + label

*Guidance:*  
You can supply foreground and background colors to replace the default colors. However, note that vehicle OEMs can choose whether or not to use the colors you supply in AAOS versions of your app. You can also specify which button is the primary button or create a timed button with a built-in timer.

Keep labels short to avoid truncation – especially on the Navigation template, where there is less space than on other templates.
Primary button

In certain templates that feature up to 2 buttons (Message, Long Message, and Pane), you can optionally tag one button as primary, to represent the primary action. The primary button then gets special treatment in the UI, such as highlighting with the app accent color, to enhance its prominence and usability.

**Note:** On AAOS, vehicle OEMs can apply their own color to the primary button and determine its horizontal order with respect to the other button.

In this Android Auto example, the primary button on the Message template is colored with an accent color supplied by the app.

In this AAOS example, the vehicle OEM has applied its own accent color and chosen to put the primary button on the right.
Timed buttons

Apps can create buttons associated with default actions that are taken automatically if the user doesn’t act within a specified amount of time (customizable by the app). For a sample flow using this strategy, see Respond to a timed alert.

To convey the countdown to the user, the button itself becomes a timer, with a built-in progress indicator. The timer countdown is indicated by the shading that moves horizontally across the button.

The app library determines the timer color based on the app’s suggested background color for the button, modifying it as needed to ensure sufficient contrast.

To create a timed button, assign a default flag to it.

In these examples, the “Resume” or “Accept” action will automatically be taken if the user doesn’t choose the other action before the shaded progress indicator moves all the way across the button.
Row

**Used in:** List, Place list, Search, Route Preview, and Pane templates (with no line dividers in the Pane template because its rows are not actionable)

**Can include:**

- Primary text (mandatory), up to 2 lines, where the second line either wraps or comes after a line break
- Secondary text (optional), up to 2 lines with customizable text color
- Optional inline icon or image in either primary or secondary text
- Caret (optional), indicating presence of a submenu

There are also 2 additional row options for List template only.

**Guidance:**
Rows are highly customizable and can be used to create a variety of list types: lists with data, lists with images, and so on.

---

**Sample row variants**

**Image + text**
- Primary text
- Secondary text

**Wrapping text**
- Primary text that wraps and continues on a second line
- Secondary text that wraps and continues on a second line

**Text + caret**
- Primary text
Header

Used in: All templates except Navigation. In map-based templates, the header appears in the card. Otherwise, it incorporates the optional action strip.

Can include:

- Icon or image (optional), such as app icon
- Text (one line only – longer text is truncated), typically the title of the screen
- Refresh button (only on Place List templates)
- Back button (optional)

Guidance:
A header must include at least one of the following: text, app icon, or back button.
# Row options for List template only

In addition to the typical features available for the Row component (primary and secondary text, image or icon, and carat), a row on the List template can also include either of the following:

- Toggle switch (optional)
- Radio button (optional, used only in selectable lists, which must have radio buttons on all rows)
- Longer secondary text (visible only when parked), to be truncated to 2 lines while driving

**Guidance:**

A row with a toggle switch can't contain a radio button, and vice versa. Also, carats are not used on lists with either toggle switches or radio buttons. However, a row with either of these options can also contain an image or icon, as well as wrapping text.

<table>
<thead>
<tr>
<th>Wrapping text + toggle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primary text that wraps and continues on a second line</td>
</tr>
<tr>
<td>Secondary text that wraps and continues on a second line</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Radio button + text</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Primary text" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Longer secondary text (only while parked)</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Primary text" /></td>
</tr>
<tr>
<td>Secondary text that wraps and continues on a second line</td>
</tr>
</tbody>
</table>

Secondary text that wraps and continues to multiple lines while parked. Parked can be defined by the OEMs and can change from car to car. Apps do not need to worry about this and just send us the content they’d like to show here.
android for cars