



PowerApps Accessibility Standards and Guidelines

White paper

Summary: This is a technical white paper aimed at Microsoft PowerApps makers in the enterprise. This document contains standards and patterns to ensure that your apps comply with [Web Content Accessibility Guidelines](#) and work properly with assistive technologies.

Writers: Aniket J Gaud, Krishna Chaitanya (Kc), Snigdha Manasvi.

Technical Contributors: Emma Cooper, Pat Dunn, Venu Gopal Gaddamedi, Rema Gopinathan, Tah Wei Hoon, Ruth Jacob, Filip Karadzic, Vikas Khanna, Anirudh Kishan, Aza Mathew, Santhosh Sudhakaran.

Contents

1	Introduction	4
1.1	Purpose of this white paper	4
1.2	Scope of this white paper	4
1.2.1	This is a living document	4
2	Getting started	5
2.1	Accessibility overview	5
2.2	Accessibility in PowerApps.....	5
2.3	Further reading	6
3	Building accessible PowerApps canvas apps	6
3.1	General guidelines	6
3.1.1	Keyboard accessibility	7
3.2	Screen.....	7
3.3	Label	9
3.3.1	Live label	10
3.3.2	Interactive label	13
3.4	Button	13
3.5	Text Input.....	15
3.6	HTML Text	17
3.7	Gallery	18
3.8	Image.....	19
3.9	Icons	19
3.10	Add picture.....	20
3.11	Data card.....	20
3.12	Check box.....	21
3.13	Sliders.....	22
3.14	Toggle.....	23
3.15	Timer	23
3.16	Known accessibility issues in PowerApps	24
4	Testing PowerApps for accessibility.....	24
4.1	The Accessibility checker	24
4.2	Accessibility Insights	25
4.3	Color Contrast Analyzer	26

4.4	Configuring your device	27
4.4.1	Android TalkBack	27
4.4.2	iOS VoiceOver	28
4.4.3	Narrator.....	29
4.4.4	Windows high-contrast mode.....	30
5	High-contrast mode	31
6	PowerApps accessibility scenario	33
7	References	36

1 Introduction

PowerApps is a high-productivity application development platform from Microsoft. The platform is used by Microsoft to build first-party applications on Dynamics 365 for Sales, Service, Field Service, Marketing, and Talent. Enterprise customers can build their own custom line-of-business applications using the same technology. Individual users and teams within your organization can also build personal or team productivity applications with no code or low code.

1.1 Purpose of this white paper

This white paper is for the enterprise application developer (maker) responsible for designing, building, testing, deploying, and maintaining PowerApps in a corporate or government environment. This white paper is a collaborative effort of the Microsoft PowerApps team, Microsoft IT, and industry professionals. Adherence to the guidelines and standards in this document will assist developers in making their PowerApps canvas apps accessible to all app users. From here on, this document refers to enterprise application makers and developers as makers.

1.2 Scope of this white paper

Unless specifically noted, all features mentioned in this white paper are available as of May 2019. The following topics are out of scope for this white paper:

- PowerApps fundamentals for building applications. This paper assumes that the reader has a working, but not necessarily expert, knowledge of how to build a PowerApps canvas app and a basic understanding of accessibility. For blogs, tutorials, training resources, and community support, visit “Learn PowerApps” at powerapps.com: <https://docs.microsoft.com/en-us/powerapps/index>.
- Power BI and other parts of the broader Microsoft Power Platform.
- General coding standards for PowerApps canvas apps. For that, we recommend that makers read our companion white paper at <https://aka.ms/powerappscanvasguidelines>.

1.2.1 This is a living document

This white paper is intended to be a living document. As the Microsoft Power Platform capabilities and industry standards change, so will this document.

Microsoft is listening to customer feedback and constantly evolving the Microsoft Power Platform so that makers can build better apps for users. As a result, today’s best practice might be deprecated as new features change the most efficient approaches. Check back periodically to see the latest standards and guidelines.

Thank you to all the professionals who contributed to this document for sharing your collective guidance and experience. Now, on to the guidance.

2 Getting started

In this section, we cover the fundamentals: What is accessibility, and how is it implemented in Microsoft PowerApps Studio? Where can you find online documentation about canvas apps? If you have a good foundation in these topics, you can skip ahead to the next section, where we cover implementation.

2.1 Accessibility overview

The most humbling experience for our engineering team came when we asked a user with visual impairments to test our new internal Careers site. We watched his frustration mount as the focus jumped unpredictably around the screen and cryptic control names were read off, out of order, in a robotic voice. To our horror, he tabbed right past the polished “Search Jobs” image button our UX group had worked so hard to create. To him, it was invisible. His verdict? “I can’t use this site.”

Let that sink in for a moment—he could not apply for a job using our site.

PowerApps embodies the idea of “democratization of **development**”—anyone in your organization can quickly and easily create a powerful app and share it broadly. But the app maker has an ethical, and sometimes legal, obligation to support “democratization of **usage**” as well—any user of your app must be able to use it as it was intended.

Democratization of usage can unintentionally be compromised. For example, a user with profound vision loss might not be able to differentiate light gray text from a white background. Other users might not have the motor capabilities to tap on a mobile device screen or will be unable to use your app if the **TabIndex** property of a control is undefined.

As an app maker, you must understand that users might have vision, hearing, mobility, or cognitive impairments that make it necessary for them to use assistive technologies to use your apps. These technologies vary across mobile and desktop platforms.

The World Wide Web Consortium (W3C) is the main international standards organization for the World Wide Web. W3C has developed the Web Accessibility Initiative (WAI), which is a great place to learn about accessibility [fundamentals](#) and standards, like the [Web Content Accessibility Guidelines \(WCAG\)](#) and [mobile-specific content](#).

2.2 Accessibility in PowerApps

The Microsoft Power Platform has been designed with accessibility in mind. This begins with the maker experience itself, where the app builder is free to create accessible apps with PowerApps while using assistive technologies. For example, [Image 2-1](#) shows a blank app in high-contrast mode.

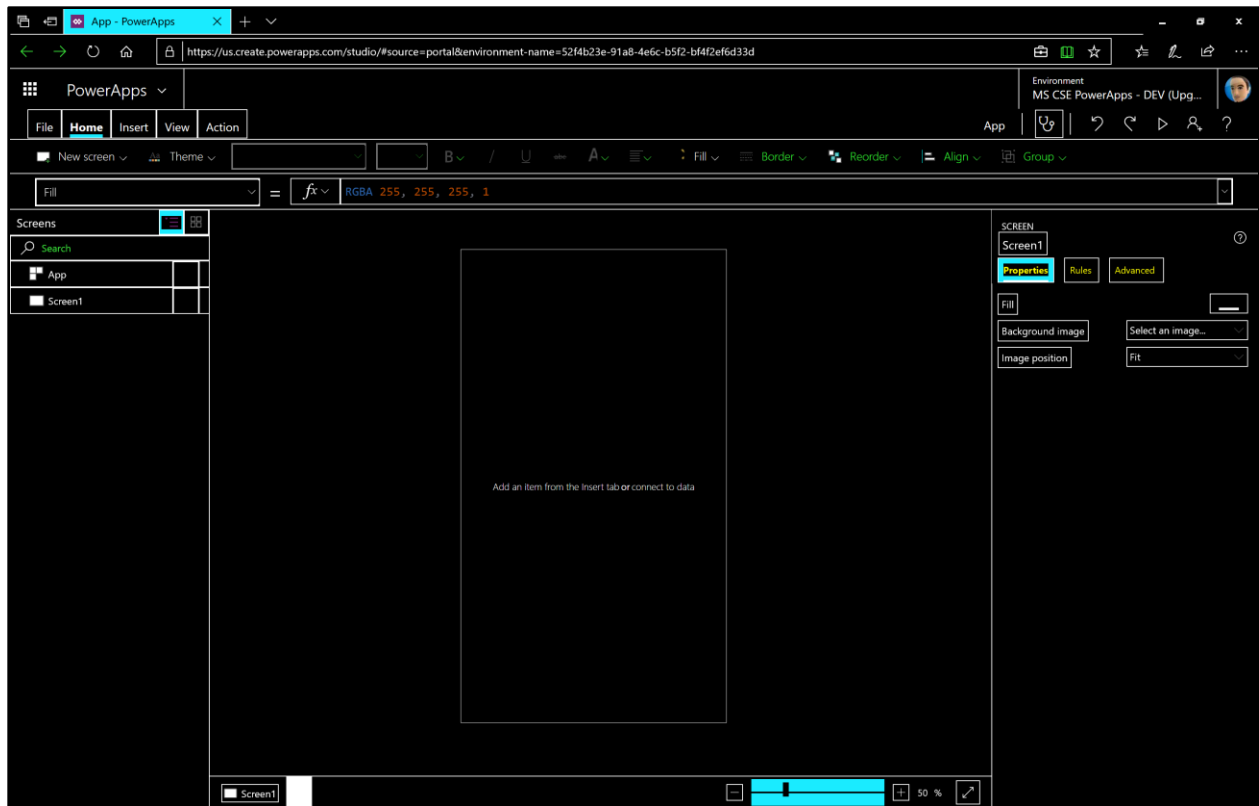


Image 2-1

Dedicated teams of engineers around the world have strived to ensure that PowerApps support accessibility, yet it is possible to build an app out of accessible components that are implemented in an inaccessible way. For example, a maker might choose to make a button out of an **Image** control with no alt text, and a user with visual impairments would not be able to understand its purpose.

2.3 Further reading

Fortunately, great resources are already available within the Microsoft Power Platform to get you started. To begin with, we highly recommend that you read [Create accessible canvas apps in PowerApps](#) to understand how to build and test your apps for accessibility.

3 Building accessible PowerApps canvas apps

In this section, we cover practical accessibility guidelines for the screens and controls in your apps.

3.1 General guidelines

An accessible PowerApps canvas app begins by having an orderly, consistent way of defining the controls on each screen. Some controls are decorative, like a horizontal line between Form sections, or important but non-interactive, like text on a label.

3.1.1 Keyboard accessibility

Some controls are interactive, where a user can select the control to interact with it (like entering text or selecting a drop-down list). Follow these general guidelines for interactive controls:

- Try to have a control flow that matches your language. For example, an English speaker would expect that the Tab key would move the focus on the screen from top to bottom and left to right.
- The **TabIndex** property must be set to **0** or greater so that keyboard users can navigate through interactive controls.
- The visual focus indicator (focus border) should be visible. To do this:
 - The **FocusBorderColor** property must be set to indicate that the control is focused, and the border color must have a minimum 3:1 contrast ratio with the background color. For more information, see [WCAG standards](#).
 - The **FocusBorderThickness** property must be set to a value greater than **0** for the border to be visible.
 - If the control doesn't have a border when it is unfocused, we recommend setting the **BorderStyle** property to **BorderStyle.Dotted**.

3.2 Screen

Many app makers assume that screen names are seen only in the PowerApps editor, but screen names are also read by screen readers. For example, if your application launches to a main menu screen called **scrnMainMenu**, the screen reader attempts to pronounce the screen name before reading the content on the screen.

To make the screen names understandable when announced by a screen reader, follow these guidelines:

- End all the screen names with the word “screen” so it’s clear what is being announced by the reader as shown in [Image 3-1](#).

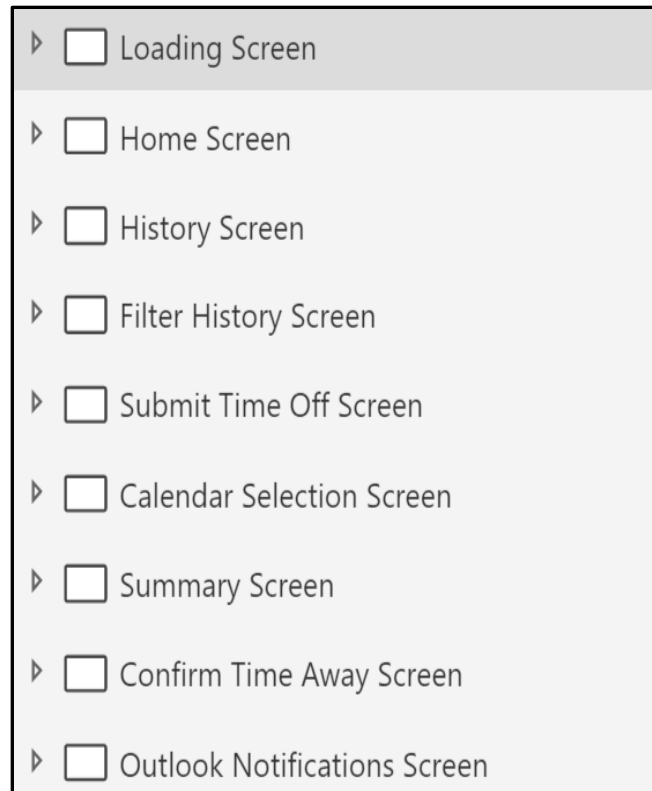


Image 3-1

- The screen name should describe the screen’s contents and function. For example, if your app has a screen that is used to search for colleagues in an organization, you could set the screen name to **Search colleagues screen**.

In cases where a confirmation dialog box or a pop-up window is required, we recommend creating a separate screen ([Image 3-2](#)) instead of having it overlay on an existing screen ([Image 3-3](#)). Otherwise, screen readers might detect and announce controls in background screens.

For example, when reading the screen on the right ([Image 3-3](#)), the screen reader might read the background control labels (like **Available Days** and **1 Deducted Day**), which is not intended.

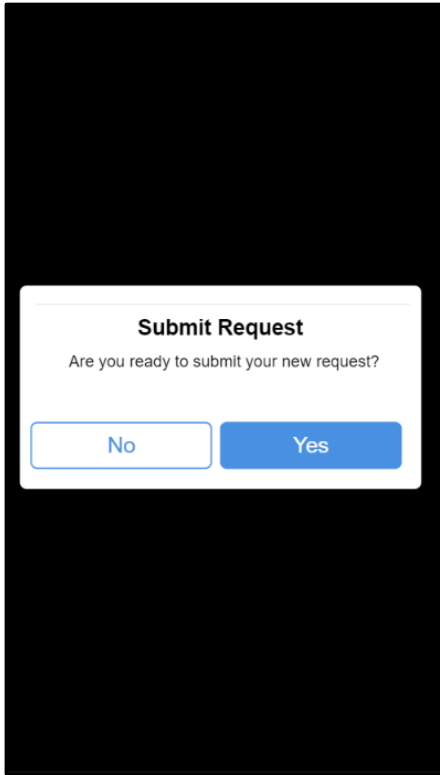


Image 3-2

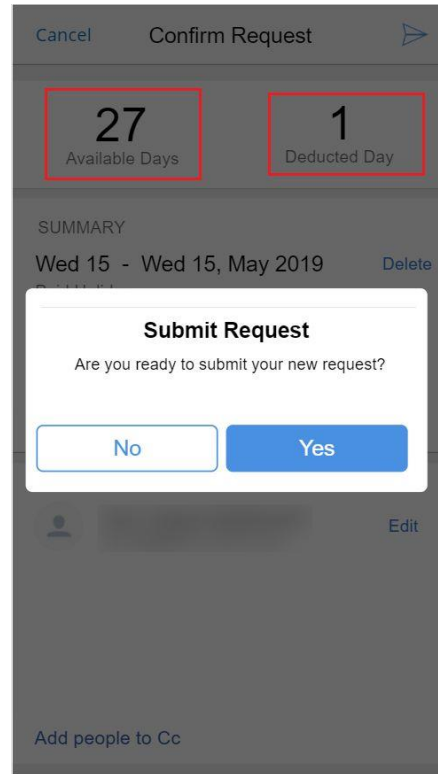


Image 3-3

3.3 Label

Labels can convey critical content and context to app users. To ensure that labels are accessible, follow these guidelines:

- Make sure screen readers announce labels. Screen readers announce the **Label** control's **Text** property if the label's **Visible** property is set to **true**.
- Make sure text can be easily seen. Per [WCAG standards](#), visible text must have a minimum luminosity contrast ratio of 4.5:1 against the background to provide enough contrast between text and its background so that it can be read by people with moderately low vision.
- Make sure headings have labels. Labels can be used for heading levels 1 to 4 by setting the **Role** property to the correct level as shown in [Image 3-5](#). Headings are necessary for a user to understand the structure of information on the screen. When it comes to headings in a screen, we follow this convention:
 - Screen headings can have a **Heading1** role as in [Image 3-4](#)
 - **Heading2** and **Heading3** roles can be used for subheadings and gallery names within a screen.
 - **Heading4** roles can be used for gallery item names.

Note: The **Tooltip** property for a label is read inconsistently by different screen readers and browsers. For example, Narrator will read a tooltip after a long pause, but other screen readers might not announce it at all. Some screen readers allow the user to toggle reading on and off. For these reasons, don't rely on the **Tooltip** property for accessibility.

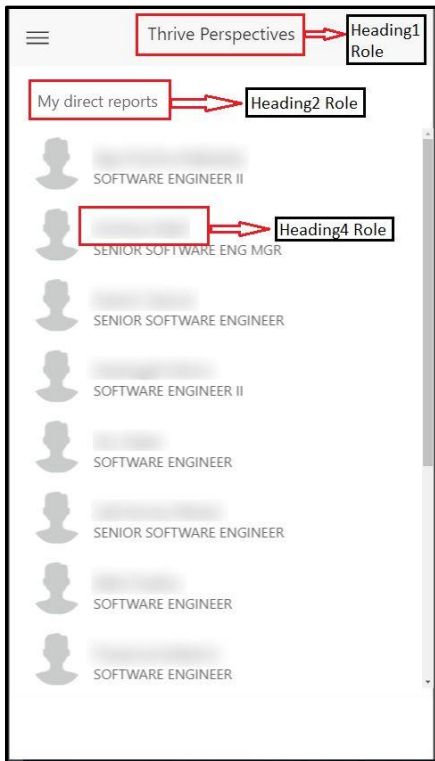


Image 3-4

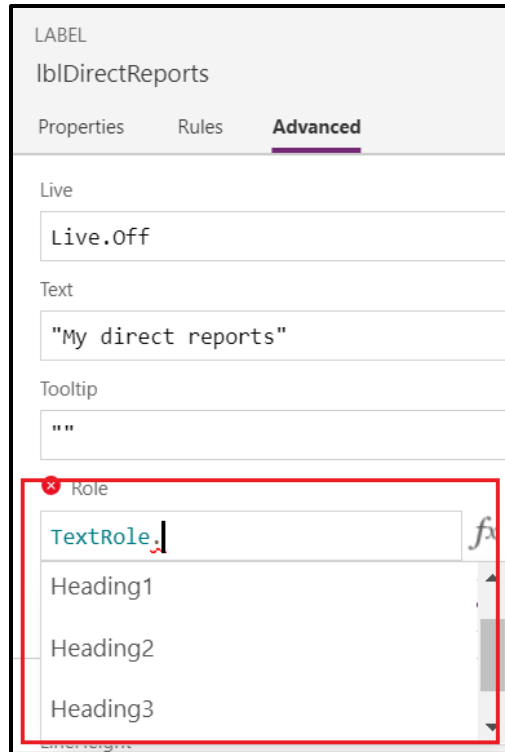


Image 3-5

Here are other ways to make labels more accessible in PowerApps:

- Use live labels
- Use interactive labels

3.3.1 Live label

Labels have a **Live** property that can enable screen readers to announce changes to the label's **Text** property. This property is useful for announcing dynamic changes in the app's UI in an accessible way.

The **Live** property has three settings, and each behaves differently.

Off	Screen readers don't announce dynamic changes.
Polite	Screen readers finish speaking before announcing any changes that occurred while speaking.
Assertive	Screen readers interrupt themselves to announce any changes that occur while speaking.

When you set the **Live** property to **Polite** or **Assertive**, the screen reader announces dynamic changes in the app— that is, it announces the **Text** property of the label as it changes.

Here are some recommendations for using live labels:

- Use live labels to announce events such as the selection of an item. For example, in [Image 3-6](#), the **Live** property of the label at the top of the screen is set to **Assertive**. As the user selects people in the people-picker control, the screen reader immediately announces how many people have been selected.

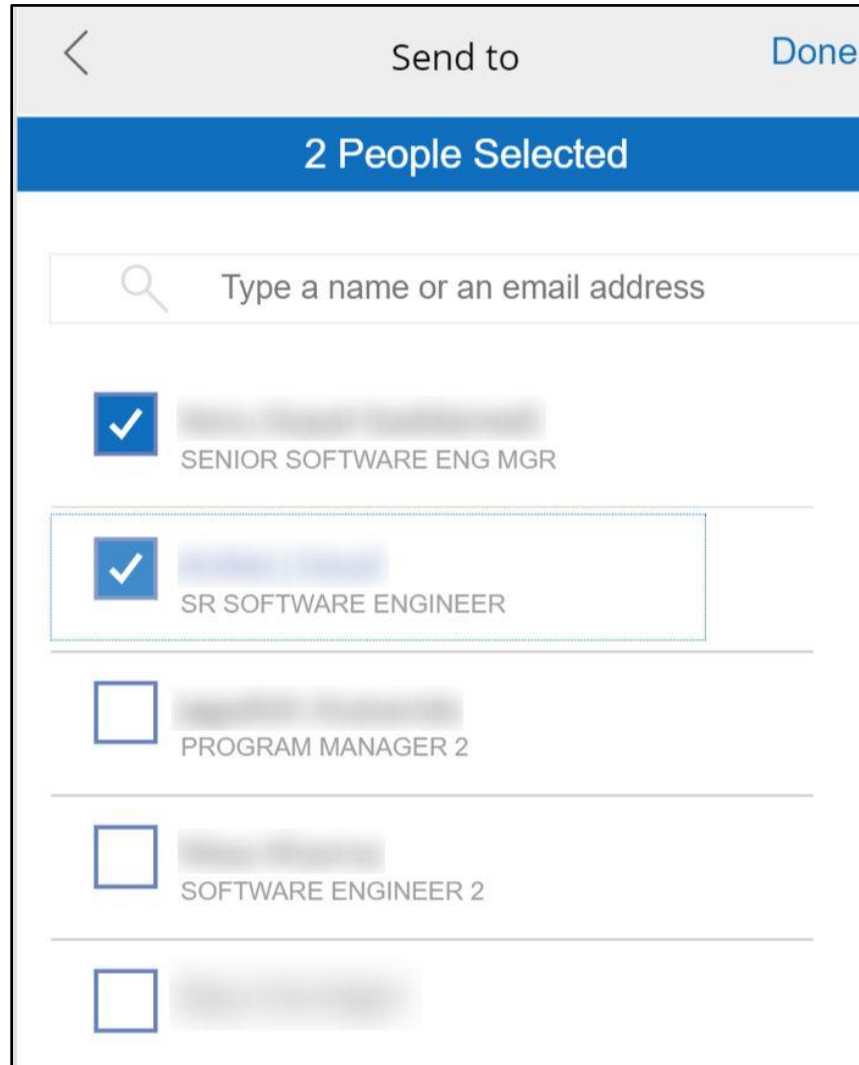


Image 3-6

- Use live labels to announce when an item is inserted or deleted from the UI. For example, if the user selects a **Delete** button, a live label can change its text to read “The record was successfully deleted.”

In a scenario such as the one shown in [Image 3-7](#), screen readers should announce when a user removes a person. This can be done by using a timer to reset the **Text** property of the live label after the announcement is made. In this example, set the **Duration** property of the **Timer** control to the time span taken by the screen reader to announce the text (typically, not more than 1 second). Set the **OnTimerEnd** property to make the live label’s text blank after the allotted time has completed.

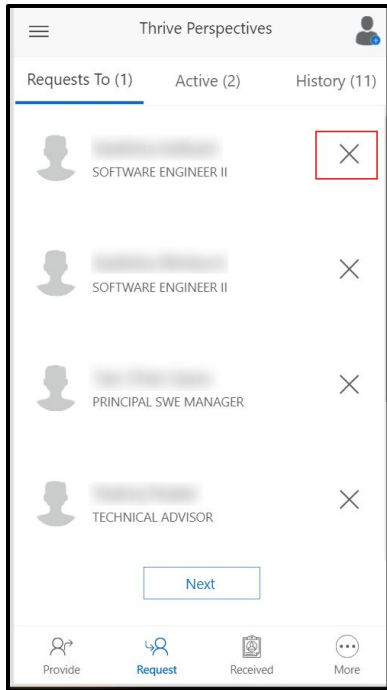


Image 3-7

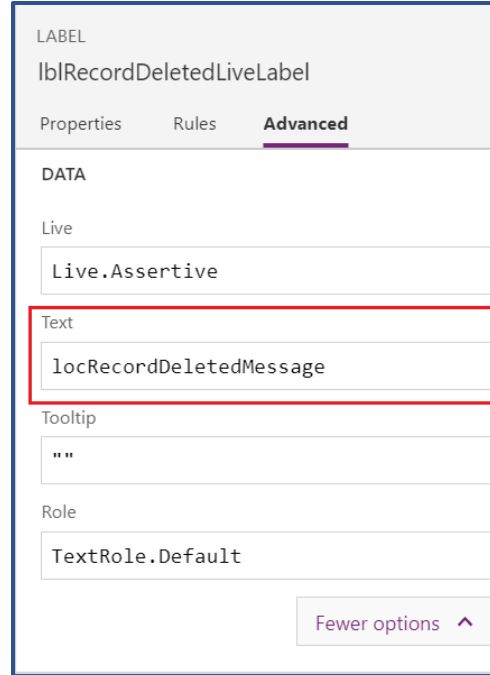


Image 3-8

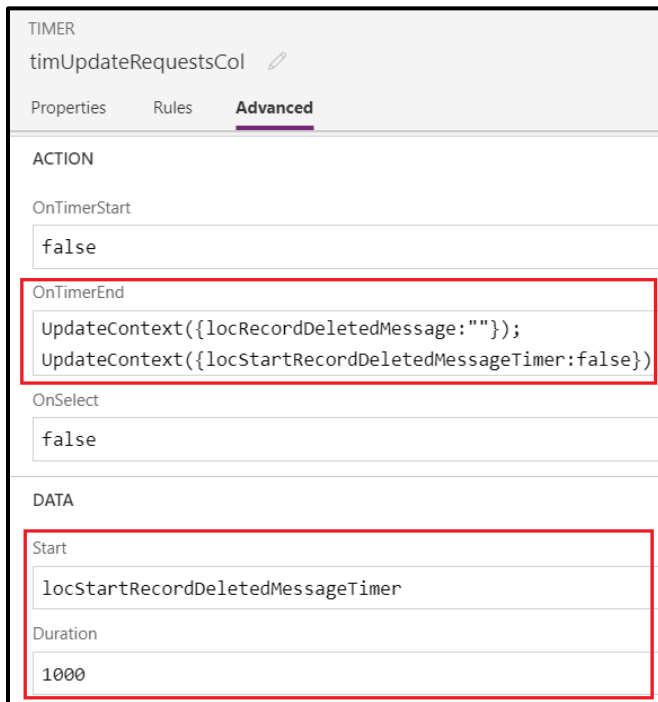


Image 3-9

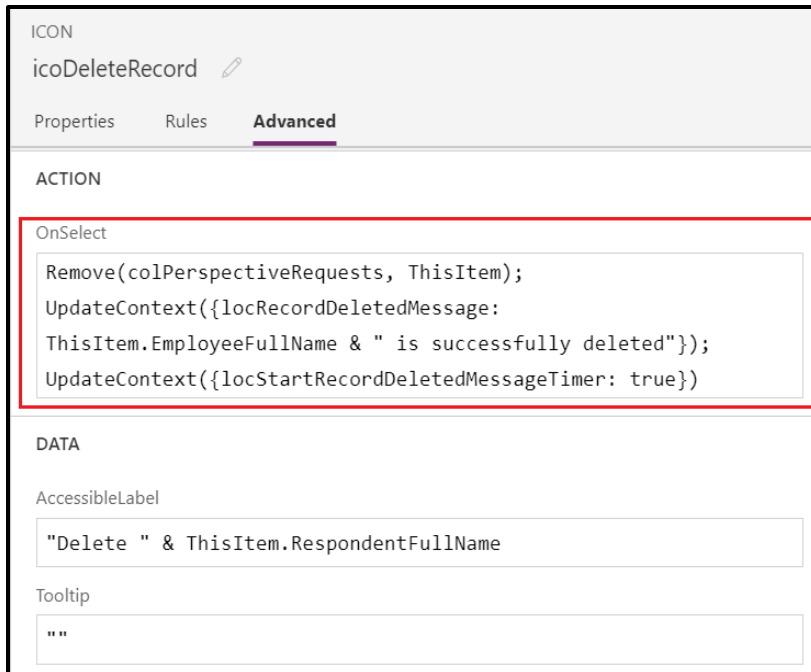


Image 3-10

Image	Control	Description
Image 3-8	Live Label	Set the Live Label's Text property to the RecordDeleted variable and initialize the variable to blank.
Image 3-9	Timer	1. Set the timer duration to 1 second and set the Start property to variable StartPeopleDeletedTimer . 2. Use the OnTimerEnd property. Update the RecordDeleted variable to blank and the StartPeopleDeletedTimer to false .
Image 3-10	Icon	Update the RecordDeleted variable to the appropriate message and the StartPeopleDeletedTimer to true when the icon is selected.

3.3.2 Interactive label

If the **On Select** property of a label is set to perform an action, then it is considered an interactive label.

To make the label keyboard-accessible, please read the [Keyboard accessibility](#) section in this document.

In general web practices, because buttons are used for interactive controls such as hyperlinks, we recommend using [Button controls](#) instead of interactive labels .

3.4 Button

The **Button** control gives the user a simple way to trigger an event or interact with the application. To make a button accessible, use the following guidelines:

- Because screen readers announce the **Text** property of a button, we recommend that the **Text** property be indicative of the button’s action. For example, in [Image 3-11](#), screen readers will announce this button as “**Policies Button**”.
- We recommend using tooltips to provide additional information to the button’s **Text** property, such as its selected state when the user moves the mouse pointer over the button.

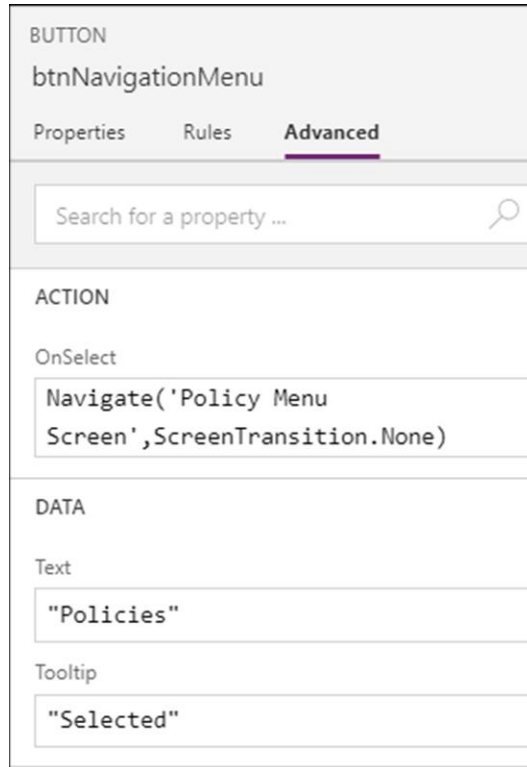


Image 3-11

- The **DisplayMode** property must be set to **Disabled** for all disabled buttons so that the screen reader explicitly announces that the button is disabled. As seen in [Image 3-12](#), when the focus moves onto the **NewPolicyBtn** button, the screen reader announces that the button is disabled or skips the button entirely.

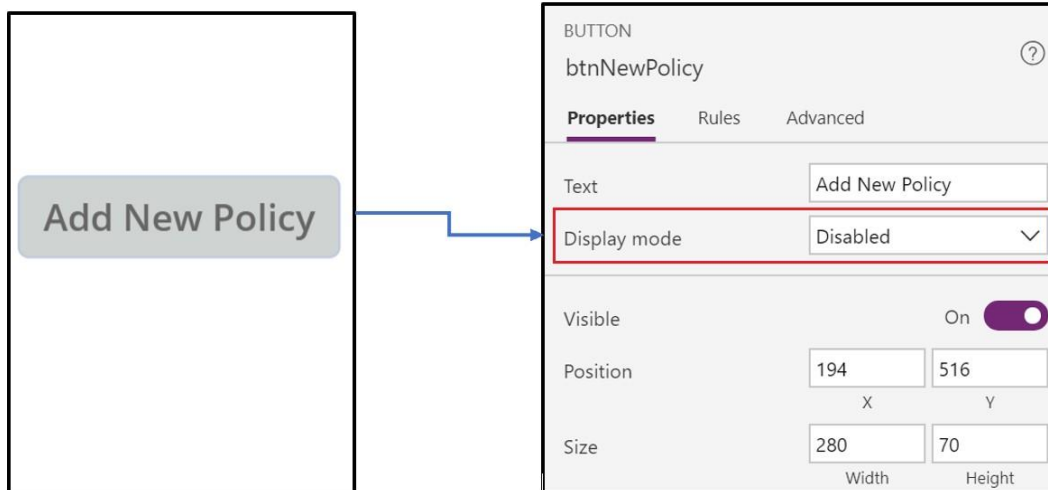


Image 3-12

- Per [WCAG standards](#), visible text must have a minimum luminosity contrast ratio of 4.5:1 against the background.
- To make the button keyboard-accessible, please read the [Keyboard accessibility](#) section in this document.

3.5 Text Input

The **Text Input** control allows a user to enter text, numbers, and other data. To make the **Text Input** control accessible, follow these guidelines:

- Use the **AccessibleLabel** property instead of the **HintText** property. The **HintText** property is rendered as a placeholder attribute in HTML. Because different browsers and screen readers handle this differently, we recommend not using it for accessibility. Instead, use the **AccessibleLabel** property.
- Per the [WCAG standards](#), visible text must have a minimum luminosity contrast ratio of 4.5:1 against the background.
- To make the **Text Input** control keyboard accessible, please read the [Keyboard accessibility](#) section in this document.
- Make the **AccessibleLabel** property descriptive so that the user can understand the purpose of the **Text Input** control.
- Many apps impose a character limit for text input fields. When this is the case, be sure to announce the maximum number of characters to your users. This can be done by setting the **AccessibleLabel** property appropriately.
- The user should be notified as the content of the **Text Input** control approaches the maximum character limit. As shown in [Image 3-13](#), when the maximum character limit is 100 for the control, the screen reader should announce “50 out of 100 characters remaining” when the user reaches 50% of the maximum character limit. This can be done using a live label and a timer. The timer should be triggered when the length of the text input is 50%. Change the **Text** property of the live label using **OnTimerStart** and reset the **Text** property to blank using

OnTimerEnd.Duration can be set according to the time taken by screen readers to announce the dynamic change, which is typically 1 second.

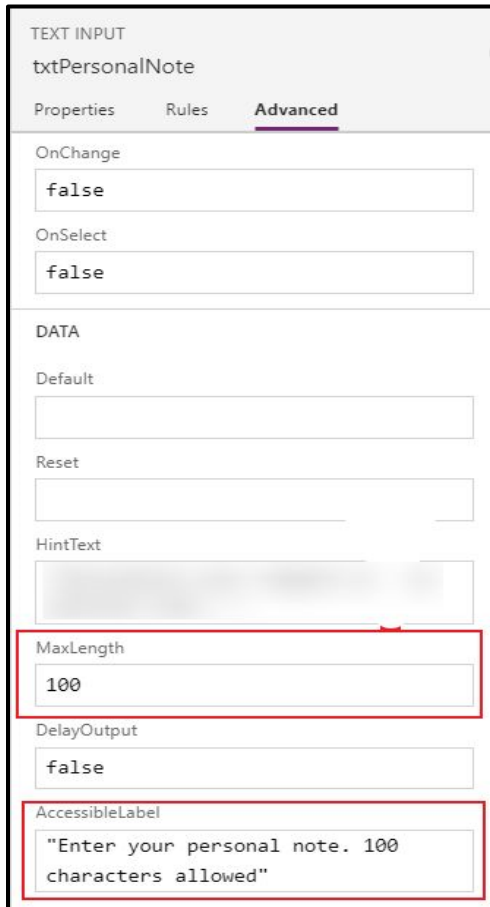


Image 3-13

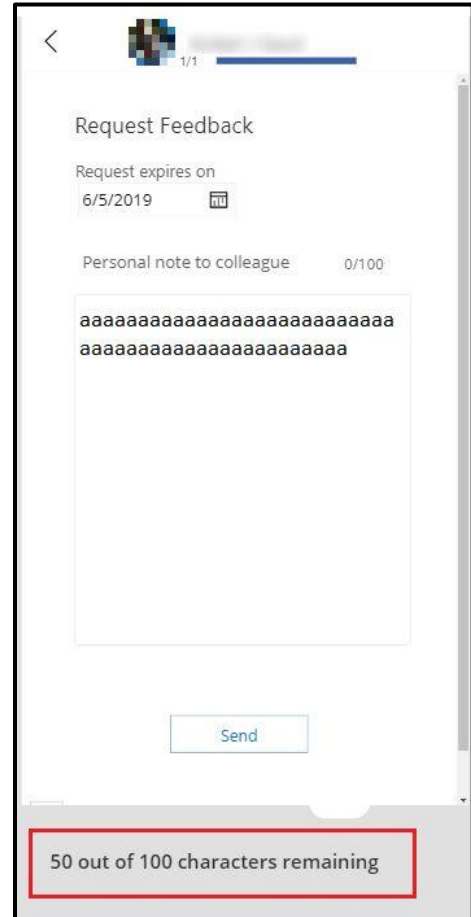


Image 3-14

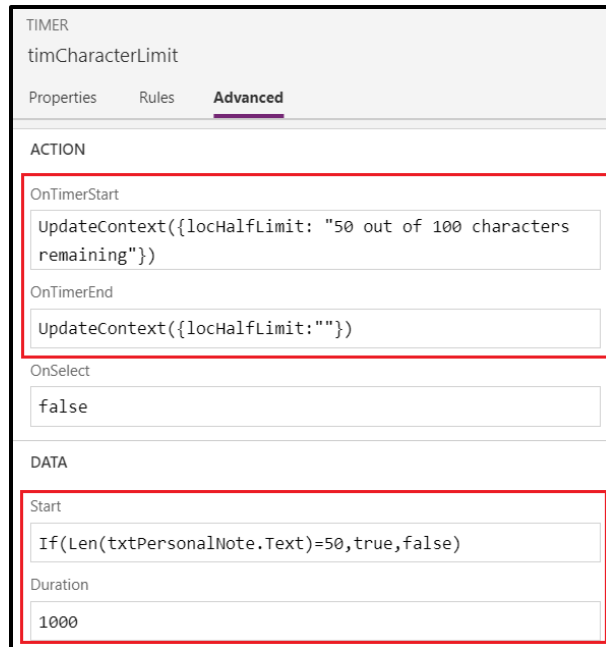


Image 3-15

Image	Control	Description
Image 3-13	Text Input MaxLength and AccessibleLabel	The MaxLength of the Text Input is set to 100 and AccessibleLabel is set appropriately.
Image 3-14	Live Label	Set the Live Label text property to HalfLimitReached and initialize the value to blank.
Image 3-15	Timer	<ol style="list-style-type: none"> 1. Set the timer duration to 1 second and trigger the timer (Start property) when the content length in the Text Input reaches the threshold. 2. Update the variable 'HalfLimitReached' variable to "50 out of 100 characters remained" in the OnTimerStart property. 3. Reset the Live Label's text by setting the variable 'HalfLimitReached' to blank in the OnTimerEnd property.

3.6 HTML Text

The **HTML Text** control is used to convert HTML tags to formatting. It can also be used to identify the headings in the screen. Follow these guidelines to make **HTML Text** controls accessible:

- Per [WCAG standards](#), visible text must have a minimum luminosity contrast ratio of 4.5:1 against the background.

- **HTML Text** controls should not contain interactive elements like **<button>**, **<a>**, or **<input>**. The TabIndex system in PowerApps does not consider elements inside HTML text.

3.7 Gallery

A **Gallery** control can show multiple records from a data source, and each record can contain multiple types of data. It can also contain other controls. To make a gallery accessible, follow these steps:

- The text in the **AccessibleLabel** property of the **Gallery** control is announced by screen readers, so be sure to make the **AccessibleLabel** property descriptive of the gallery's contents. For example, the **AccessibleLabel** property in [Image 3-16](#) is "Vacation Types".

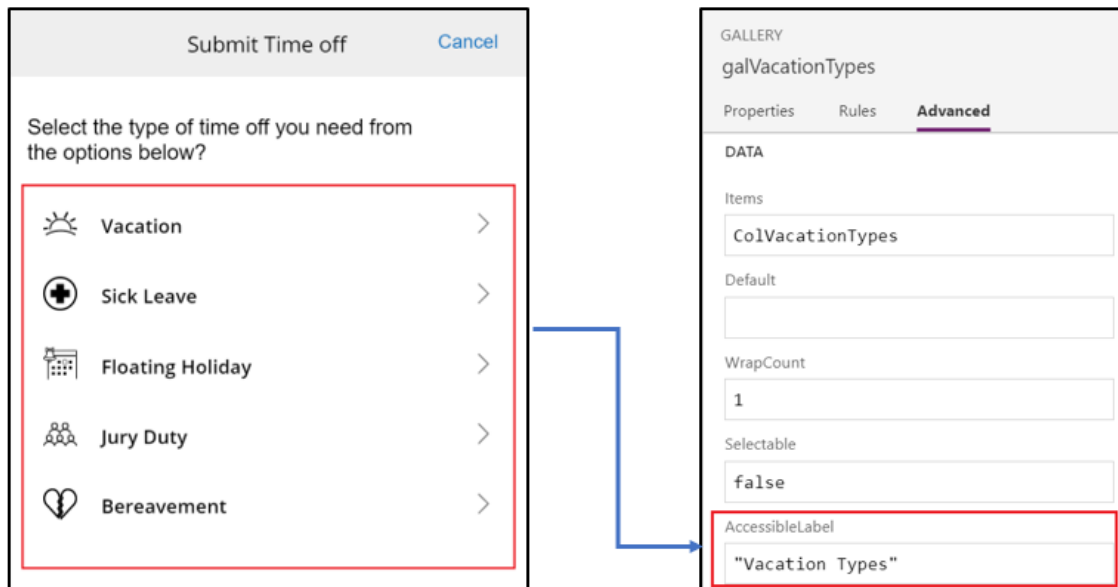


Image 3-16

- For example, if the gallery is used as a navigation menu as shown in [Image 3-17](#). Screen readers should announce the state of each menu item as "selected" or "not selected." This can be done with the **ItemAccessibleLabel** property, as in the following steps.
 - Set the **Gallery** control's **Selectable** property to **true**.
 - Set the **ItemAccessibleLabel** property to something similar to:
`If(ThisItem.IsSelected=true, "Selected", "Not selected")`

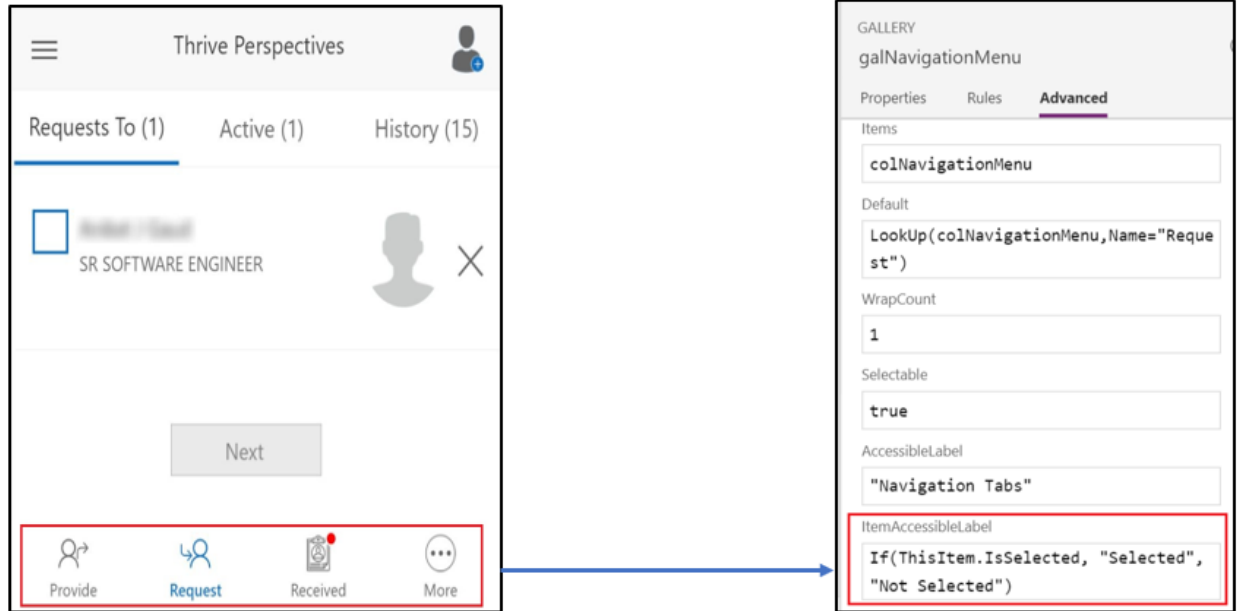


Image 3-17

3.8 Image

The **Image** control is used to display images either from a local file or from a data source.

- To make an interactive **Image** control accessible, please read the [Keyboard accessibility](#) section in this document.
- A thorough reading of the [WCAG standards](#) section on Graphical Objects will help the maker understand how to make images accessible. In general, interactive images must have a minimum of a 3:1 contrast ratio against adjacent colors. Images that are purely decorative have no minimum contrast requirement.
- An interactive **Image** control should have an **AccessibleLabel** property that describes the purpose of the image.

3.9 Icons

Icons are graphic controls with appearance and behavior properties that are configurable. The difference between icons and images is that icons are built in and implicitly provided by PowerApps.

- To make an interactive **Icon** control keyboard accessible, please read the [Keyboard accessibility](#) section in this document.
- Per [WCAG standards](#), interactive Icons should have a minimum of a 3:1 contrast ratio with the adjacent colors.
- Icons should have tooltips for understanding and operating content.
- An interactive **Icon** control should have an **AccessibleLabel** property that describes the purpose of the image.
- Use icons instead of images wherever possible because the PowerApps platform ensures that icons are visible in high-contrast mode ([Image 3-18](#)).

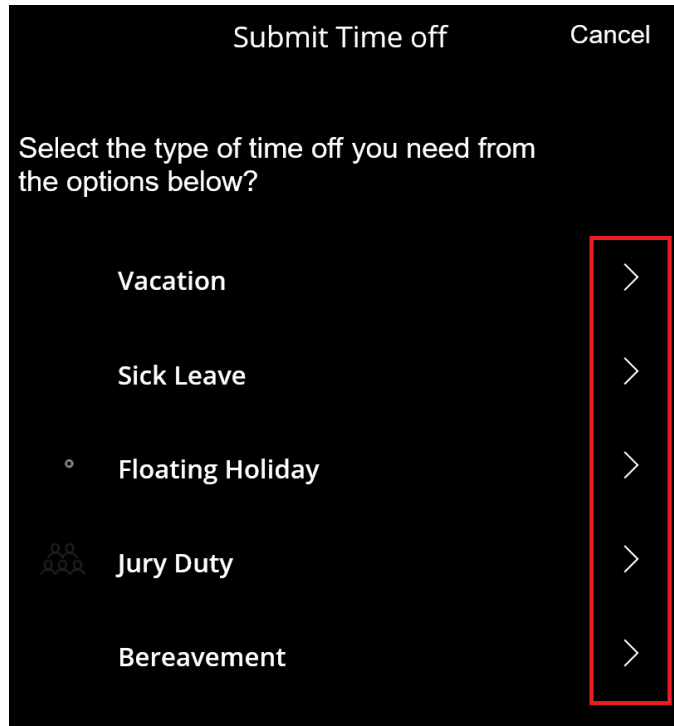


Image 3-18

3.10 Add picture

The **Add picture** control enables users to take photos or upload image files from a device and update the data source with the content. To make the control accessible, follow these guidelines:

- To make an **Add picture** control keyboard accessible, please read the [Keyboard accessibility](#) section in this document.
- Ensure that screen readers announce when a picture is added, changed, or deleted. This can be done using a live label, please read the [Live Label](#) section in this document.
- Use descriptive text in the **Text** property to help users understand the context.

3.11 Data card

Card controls are the building blocks of the **Edit form** and **Display form** controls in canvas apps. The form represents the entire record, and each card represents a single field of that record.

- The focus within a data card moves from top to bottom and left to right. In [Image 3-19](#), the highlighted area is within a data card. As a result, the focus moves from "Time" to "12:00 AM" to "-" instead of moving from "Time" to "Wed, Jan 23".

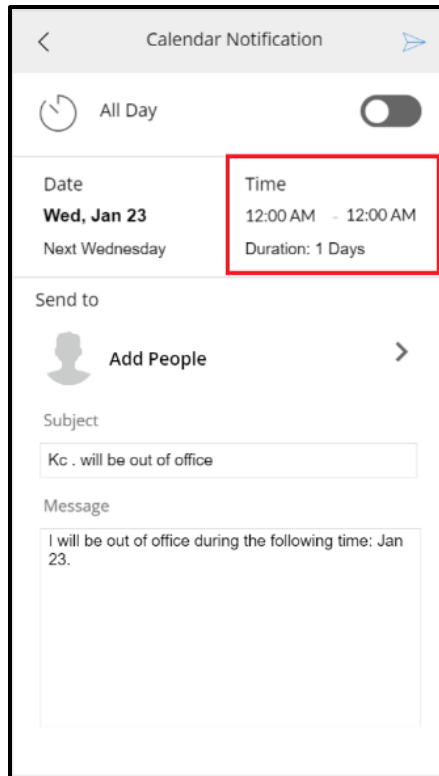


Image 3-19

3.12 Check box

A check box is a control that the user can select or clear to set its value to **true** or **false**.

- The PowerApps platform ensures that screen readers will announce the current state of the check box—whether it is checked or unchecked.
- To make a **Check Box** control keyboard accessible, please read the [Keyboard accessibility](#) section in this document.
- Per [WCAG standards](#), there must be a minimum of 3:1 color contrast ratio between the values for:
 - **CheckmarkFill** and **CheckboxBackgroundFill**
 - **CheckboxBackgroundFill** and **Fill**
 - **CheckboxBackgroundFill** and **PressedFill**
 - **CheckboxBackgroundFill** and **HoverFill**
- The PowerApps platform ensures that check boxes are visible in high-contrast mode. Please see [Image 3-20](#) for an example.
- As changes are made to the **Check Box** control value, the PowerApps platform ensures that the new state is announced by the screen reader.

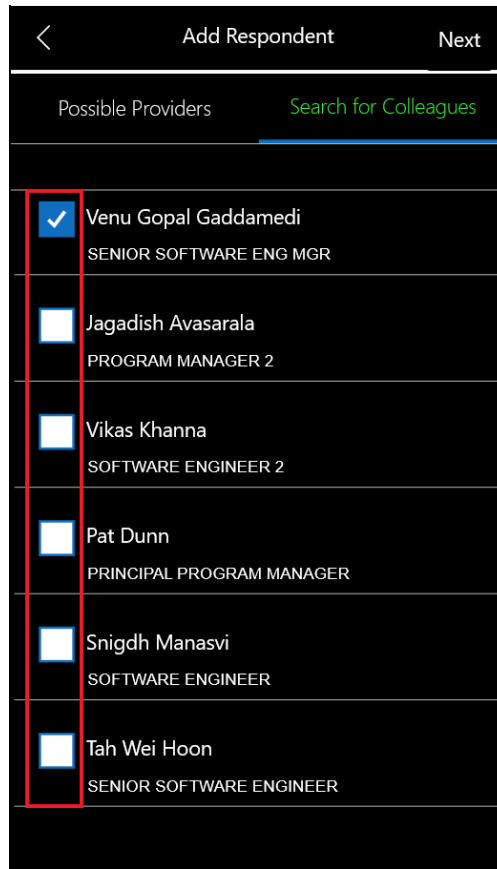


Image 3-20

3.13 Sliders

A **Slider** control (shown in [Image 3-21](#)) can be used to select between a minimum and a maximum value. The user can change the value by dragging the handle of a slider left to right or up and down.

- The PowerApps platform will ensure that the screen reader announces the value of the slider and any changes to it.
- To make a **Slider** control keyboard accessible, please read the [Keyboard accessibility](#) section in this document.
- The PowerApps platform ensures that sliders are visible in high-contrast mode as shown in [Image 3-21](#).

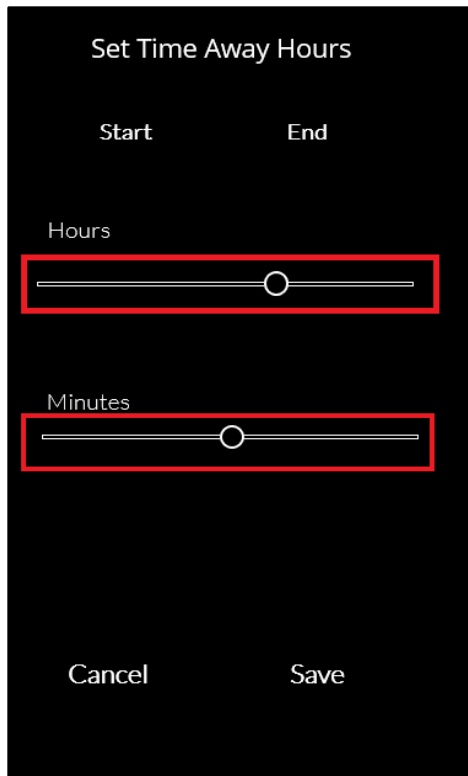


Image 3-21

3.14 Toggle

The **Toggle** control can be turned on or off by dragging the handle. It behaves the same way as a check box.

- Screen readers will announce the toggle value.
- To make a **Toggle** control keyboard accessible, please read the [Keyboard accessibility](#) section in this document.
- Screen readers must also announce any change made to the toggle value. This can be done by using a live label, please read the [Live Label](#) section in this document for more information.
- The PowerApps platform ensures that toggles are visible in high-contrast mode as shown in [Image 3-22](#).



Image 3-22

3.15 Timer

A **Timer** control can be used to determine how an app responds after a certain amount of time passes.

- Timer controls are typically hidden and used to manage some kind of event. For example, let's say that a user submits a vacation request in a time away app. The timer control can be

triggered to refresh the user's request history when the call is complete. In this scenario, set the **TabIndex** property of the timer to **-1** so that you cannot tab to the control. Also, set the **Visible** property to **false** so that screen reader does not announce them.

- If the timer is a visible interactive control on the screen, you can make it keyboard accessible. Please see the [Keyboard accessibility](#) section for details.

3.16 Known accessibility issues in PowerApps

- Focus moves randomly when navigating to a new screen in iOS while using VoiceOver.
- When text size is increased in the device's settings, the text size in PowerApps is not increased. Instead, the user must pinch to zoom in.
- SVG images are not visible in high-contrast mode.

4 Testing PowerApps for accessibility

Once you've created your application using these accessibility guidelines, it's time to test it to ensure it will be accessible to all users. This section covers how to set up your device for accessibility testing. It also describes accessibility testing features in the Microsoft Power Platform, and other tools and utilities.

4.1 The Accessibility checker

After you have built your app, be sure to check it using the [Accessibility checker](#), which scans your app for common issues such as uncaptioned images and controls that are missing the **TabIndex** property as shown in [Image 4-1](#).

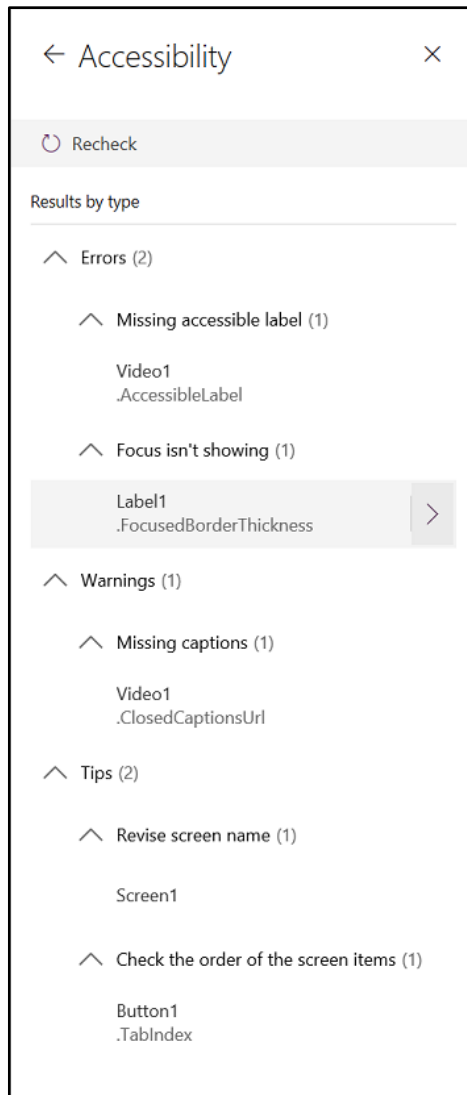


Image 4-1

4.2 Accessibility Insights

Accessibility Insights is an accessibility testing tool for makers. It helps makers to find accessibility issues during development, before they reach your users.

- Download and install [Accessibility Insights](#). The accessibility insights window opens.
- Open the application in studio mode.
 - A highlighter appears that moves over the controls in the app screen as and when the user selects a tab.
 - The callout for each control in the screen will be visible in the accessibility insights window ([Image 4-2](#)).

- This helps the maker know how each control will be called out. It results in fewer bugs in the testing phase. Makers can even record the tests done in accessibility insights.
- **Shift + F7** – Enter/toggle event recording on the selected item.
- **Shift + F8** – Run tests on the selected element anytime.
- **Shift + F9** – Minimize or activate the main window.

Example screenshot:

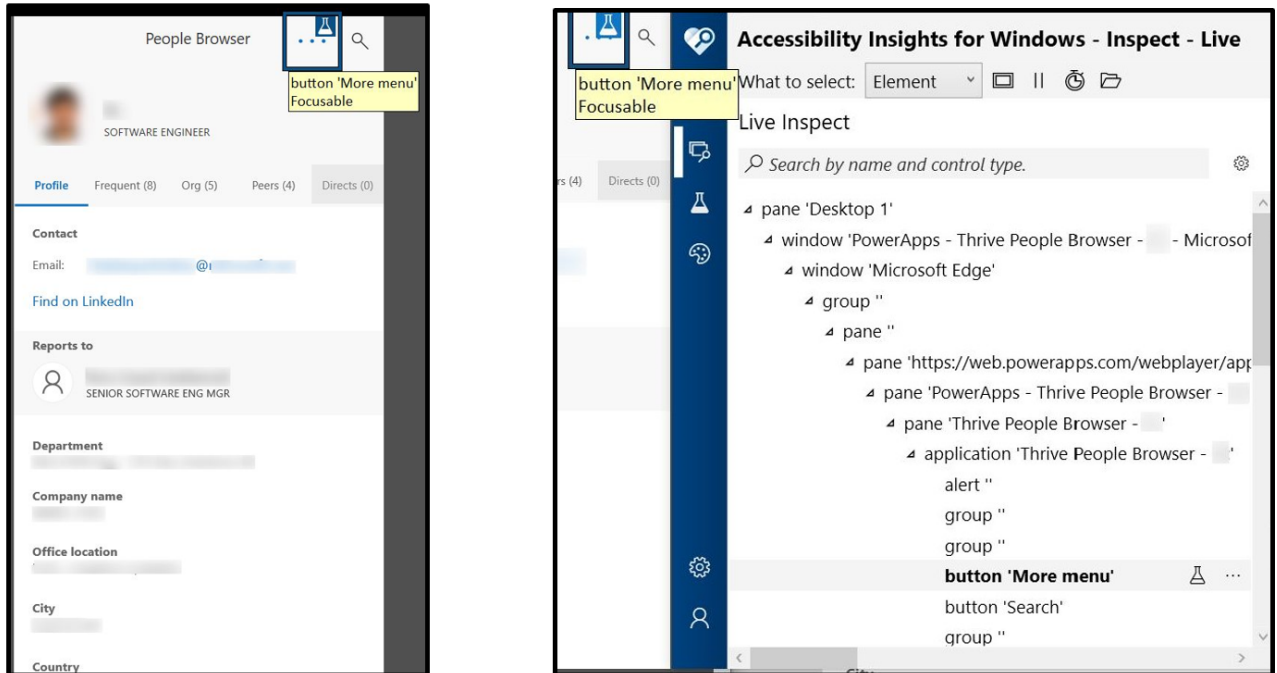


Image 4-2

4.3 Color Contrast Analyzer

To check for contrast ratio for text and icons, we recommend using [Accessibility Insights](#) or [WCAG Luminosity Contrast Ratio Analyzer](#) chrome extension.

Color Contrast Analyzer is a feature in Accessibility Insights that helps investigate contrast ratios.

- Large-scale text and images of large-scale text that have a contrast ratio of at least 4.5:1 pass the color contrast test.
- A color contrast test can be performed in Accessibility Insights.
 - Select the Palette icon in the Accessibility Insights window.
 - Enter the foreground color and background color to check for the contrast ratio.
 - To pass, the color contrast ratio should be greater than or equal to 4.5:1.
- Example: When the foreground color is black and the background color is white, the color contrast ratio is 21:00:1. This is greater than 4.5:1, so it passes the test ([Image 4-3](#)).

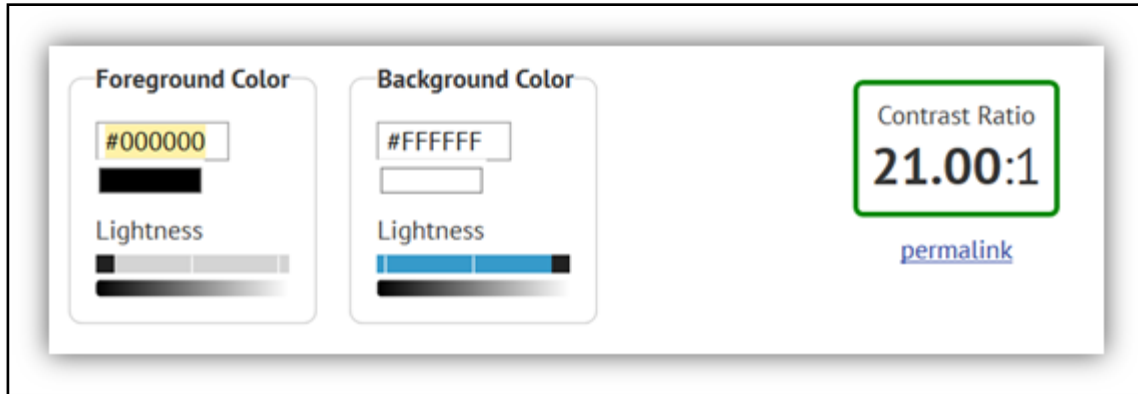


Image 4-3

4.4 Configuring your device

To begin developing and testing your PowerApps applications for accessibility, enable common assistive technologies for your platform. This list is not exhaustive, but it's a good place to start.

4.4.1 Android TalkBack

To enable TalkBack in Android devices, follow these steps:

1. Open your device's Settings app.
2. Open **Accessibility** and then open **TalkBack**.
3. Turn on **TalkBack**.
4. In the confirmation dialog box, tap **OK**.
5. Tap over the toggle button next to **Accessibility**.
 - Swipe two fingers to scroll.
 - When **TalkBack** is on, your device provides spoken feedback to help users who are blind or have low vision. For example, it describes what the user can touch, select, or activate.

Example screenshot: [Image 4-4](#)

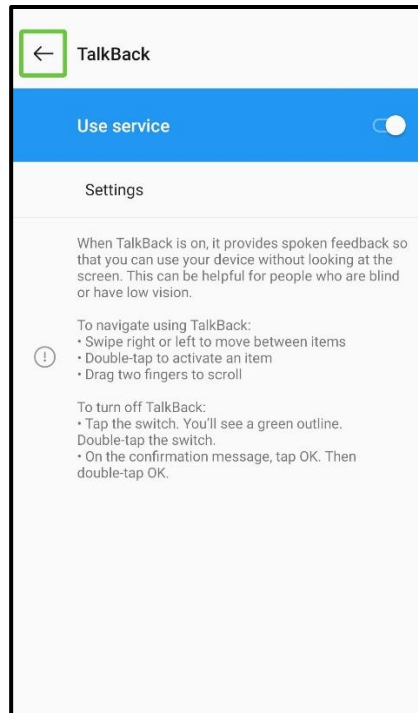


Image 4-4

For additional information, please read the [Get started on Android with TalkBack](#) article.

4.4.2 iOS VoiceOver

To enable VoiceOver in iOS devices, follow these steps:

1. Open the Settings app.
2. Tap **General**.
3. Tap **Accessibility**.
4. Tap **VoiceOver** under the **Vision** category at the top.
5. Tap the **VoiceOver** switch to enable it.
6. VoiceOver announces items on the screen.
 - Tap once to select an item.
 - Double-tap to activate the selected item.
 - Swipe three fingers to scroll.
7. Users can adjust the speaking rate (speed) at which the announcements are made.
8. Navigate to the app that must be tested, and then navigate through the controls in the app by swiping left or right, and listen to how VoiceOver describes them.

Example screenshot: [Image 4-5](#)

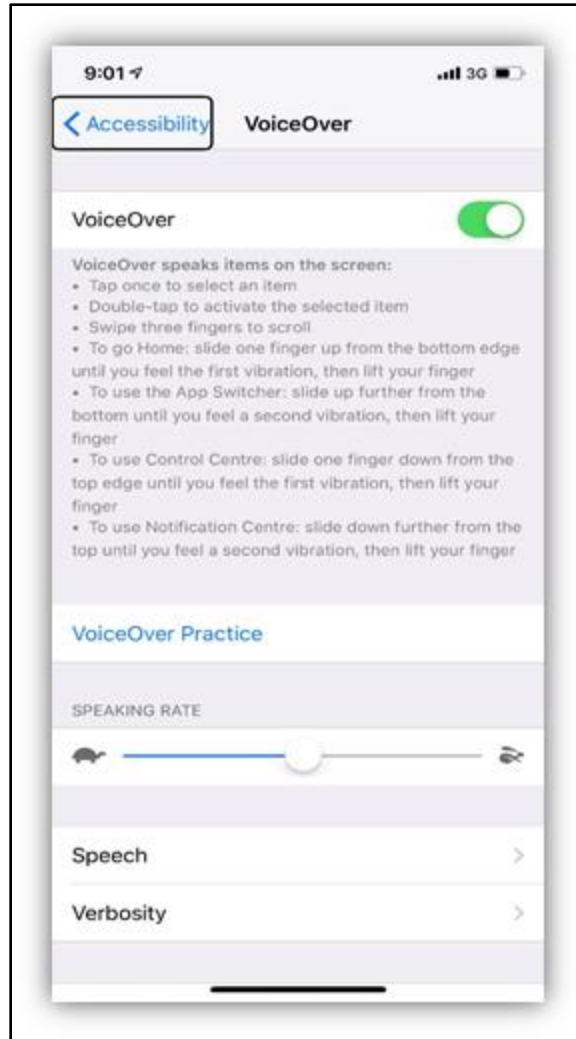



Image 4-5

For additional information, please read the [iOS Vision Accessibility](#) documentation.

4.4.3 Narrator

Narrator is the screen reader for Windows desktop devices.

To enable Narrator in Windows devices, follow these steps:

1. Open **Settings**.
2. Select **Ease of Access**.
3. Tap **Narrator** and turn on the toggle below **Use Narrator to read and interact with your device**.
4. A blue highlighter appears on the screen. As the user selects a tab, the highlighter keeps switching controls. Callouts are heard accordingly. Test how the callouts are made for each control.
5. Press **Ctrl++Enter** to exit Narrator.

4.4.3.1 Scan mode

Scan mode is the default navigation mode in Narrator. These are some frequently used keyboard shortcuts:

- **Caps lock+Spacebar** – Scan mode turned on.
- **Spacebar** – Activate an item.
- **Caps lock+Spacebar** – Scan mode turned off.
- **Ctrl+Alt+arrow key** – Navigate by cells in a row or column.
- Please read [this blog post](#) for more shortcuts.

For additional information, please read the [Complete guide to Narrator](#) article.

We recommend using Microsoft Edge while working with Narrator. Chrome and Firefox do not work well with Narrator, and Internet Explorer uses an older accessibility API.

4.4.4 Windows high-contrast mode

To enable the high-contrast mode in Windows, follow these steps:

1. Open **Settings**.
2. Select **Ease of Access**.
3. Select **High contrast**, and then select **Turn on high contrast** ([Image 4-6](#)).
4. Alternatively, the user can search for high contrast in Windows Settings.

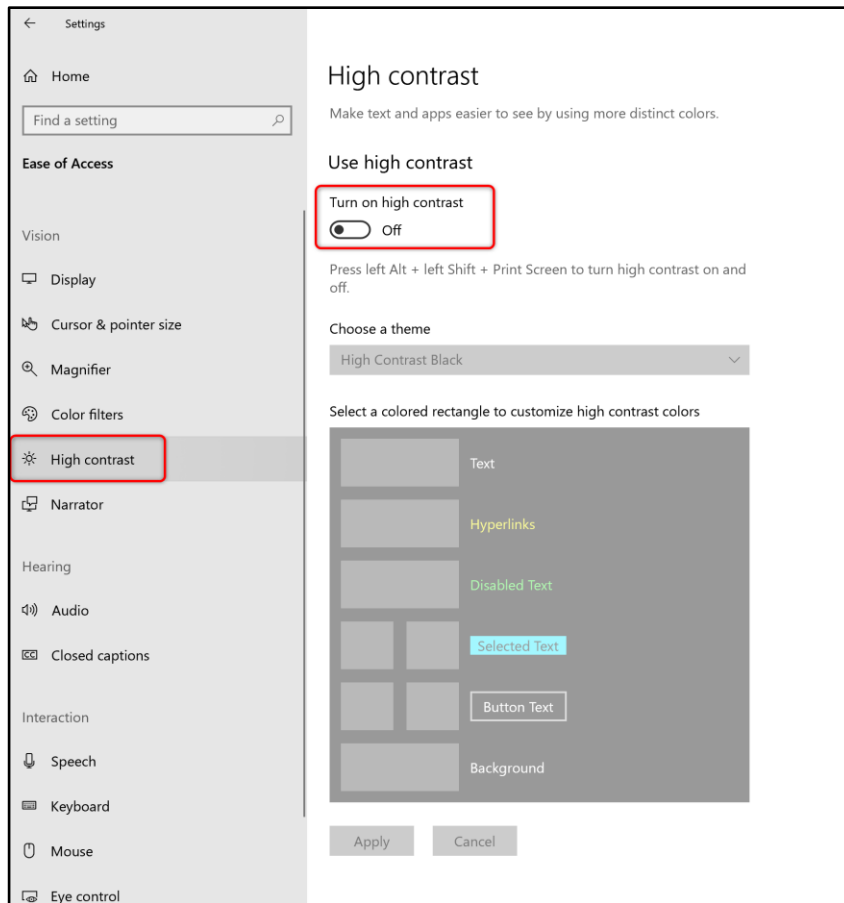


Image 4-6

For more information on how to make an app accessible in high-contrast mode, please see the [High-contrast mode](#) section in this document.

Note: We recommend using Microsoft Edge or Internet Explorer while working in high-contrast mode.

5 High-contrast mode

- Makers need to ensure that all controls are visible in high-contrast mode as in [Image 5-1](#).
- Recommendations:
 - If there are overlapping controls on a screen, ensure that the controls are reordered by using **Bring Forward** or **Send Backward** in such a way that all the controls are visible in high-contrast mode. This can be done by right-clicking the control and selecting the appropriate option under the **Reorder** tab ([Image 5-2](#)).

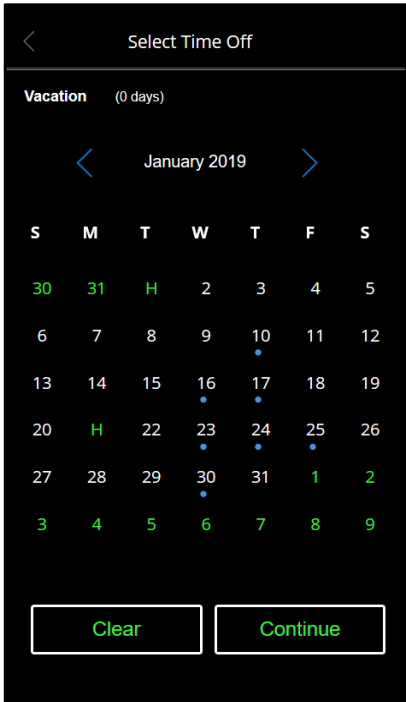


Image 5-1

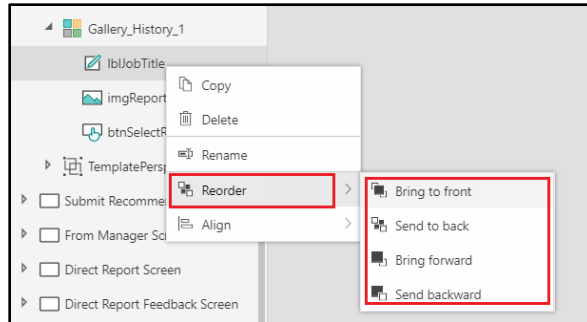


Image 5-2

- Consider the example of a **Gallery** control (Image 5-2). Each item has a **btnSelectReportee** button, a **lblJobTitle** label, and an image (Image 5-3). The entire area covered by the button must be selectable. However, if the **lblJobTitle** label (which is not interactive) is brought to the front, it makes the label area non-interactive. In such situations, set the **OnSelect** property of the **lblJobTitle** label to be the same as the **OnSelect** property of the underlying **btnSelectReportee** button. In this case, ensure that the **TabIndex** property of the **lblJobTitle** label is -1 to avoid focus moving onto the **lblJobTitle** label since the underlying **btnSelectReportee** button performs the same action. To avoid keyboard focus moving onto the **lblJobTitle** label, ensure that the **TabIndex** property is set to -1 .

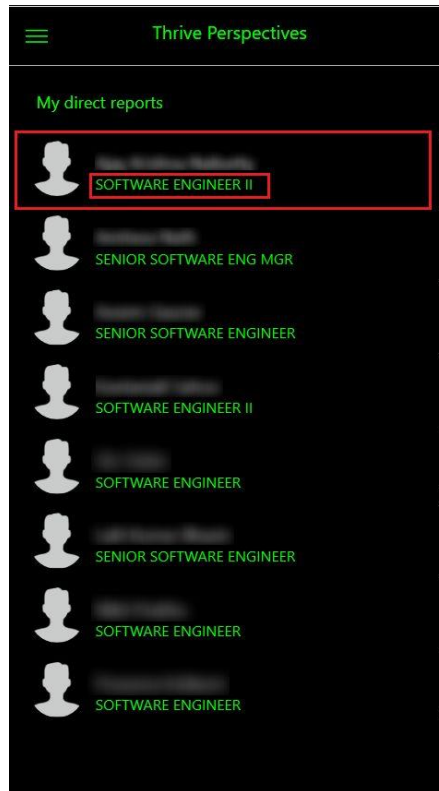


Image 5-3

- For informative or functional images, we recommend using vector formats like SVG images instead of raster formats like PNG images for the accessible high-contrast mode. Decorative images have no accessibility requirements.

6 PowerApps accessibility scenario

Consider a team that has received a request for an app that is able to search and select users. To make sure the app has search and select functionality and is accessible, the maker identifies these controls:

- A **Text Input** control where the name can type in a name to search for.
- A **Button** control to trigger the search.
- A **Timer** control to fetch the data once the user has entered enough characters to begin the search.
- A **Label** control to announce events as they occur.
- A **Gallery** control containing a **Label** control to display search results.
- A **Check box** control so the user can select people from the search results.

Using the PowerApps accessibility guidelines discussed in this white paper, the maker designs an accessible app:

- When the user types in a name and selects Search, the search timer is triggered to fetch the search results. ([Image 6-1](#))

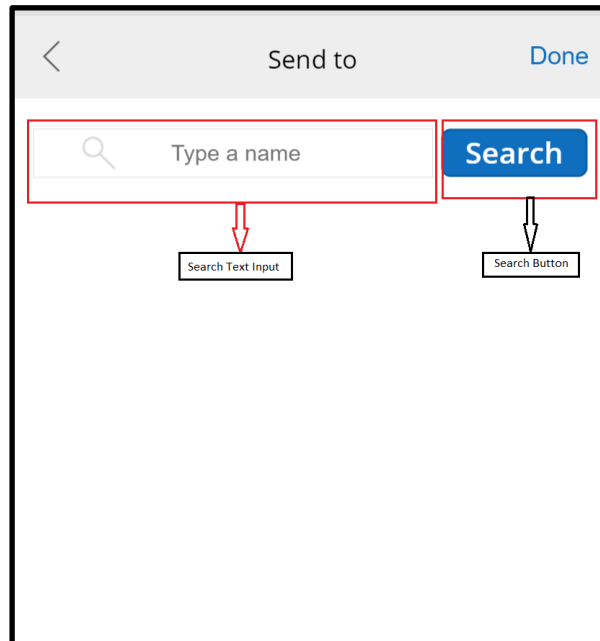


Image 6-1

- As the search results are being fetched, the text "Searching..." is displayed on a live label, and the screen reader announces that same text to the user ([Image 6-2](#)).

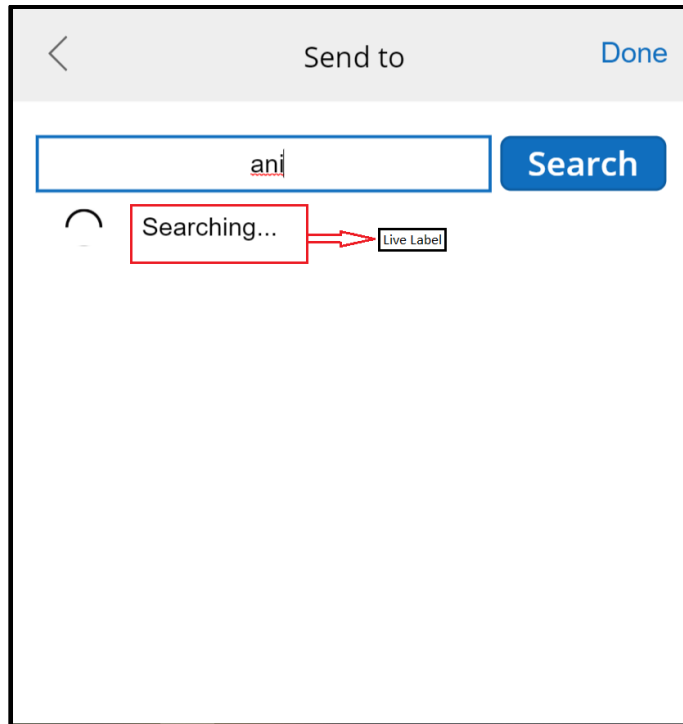


Image 6-2

- Once the search results are fetched and bound with the **Gallery** control, the screen reader reads through the search results, along with the number of records ([Image 6-3](#)).

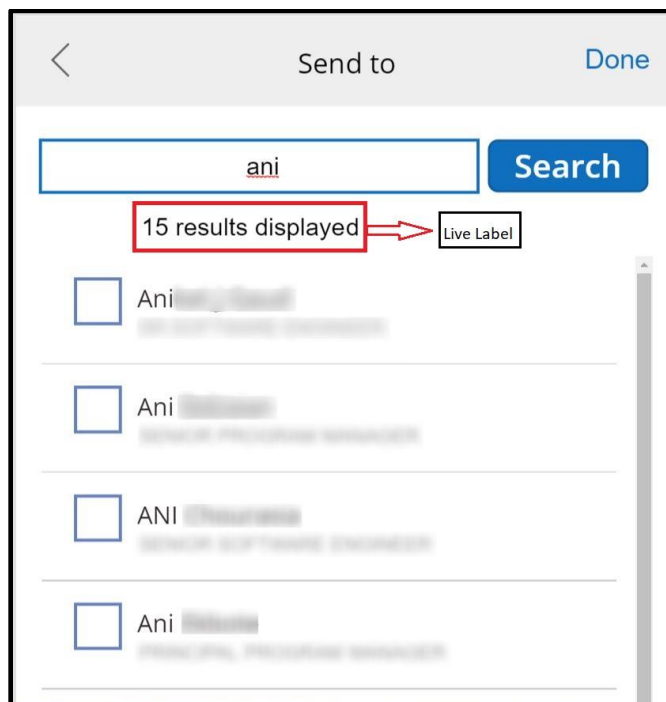


Image 6-3

7 References

- [Making apps accessible](#). Microsoft PowerApps. 7th May 2019.
- [Create accessible canvas apps in PowerApps](#). Microsoft PowerApps. 7th May 2019.
- [Mobile Accessibility at W3C](#). WAI (Web Accessibility Initiative). 7th May 2019.
- [Complete guide to Narrator](#). Microsoft Help. 7th May 2019.
- [HTML attributes - Screen reader compatibility](#). PowerMapper. 7th May 2019.