All about Carbon What's happening	~				
Desidering					
Designing Developing	~				
Contributing	~	Nlatification			
Migrating	~	Notification			
Elements	\sim	Usage Style Code Accessibility			
Guidelines	\checkmark	Usage Style Code Accessibility			
Components	^				
Overview					
Accordion		No accessibility annotations are needed for notifications, but keep			
AI label		these considerations in mind if you are modifying Carbon or			
Breadcrumb		creating a custom component.			
Button					
Checkbox		↓ What Carbon provides			
Code snippet		↓ Design recommendations			
Contained list					
Content switcher		→ Development considerations			
Data table					
Date picker Dropdown					
File uploader		What Carbon provides			
Form					
Inline loading		Carbon bakes keyboard operation into its components, improving the experience of blind users and			
Link		others who operate via the keyboard. Carbon incorporates many other accessibility considerations, some of which are described below.			
List					
Loading					
Menu		Keyboard interactions			
Menu buttons					
Modal		Users can navigate through the interactive elements within the notification using the Tab key.			
Notification		Actions such as closing or activating buttons can be performed using the Space or Enter keys. Additionally, the notification can optionally be closed by pressing the Escape key.			
Number input					
Pagination		Inline and toast			
Popover		Inline and to a not fications do not contain interactive elements. They use the roles of status , alert , or log , and they do not receive focus.			
Progress bar					

Progress indicator

Radio button

Search

Select

Slider

Structured list

Tabs

Tag

Text input

Tile

Toggle

Toggletip

Tooltip

Tree view

UI shell header

UI shell left panel

UI shell right panel

Patterns **Community assets**

Data visualization Help

 \sim

 \sim

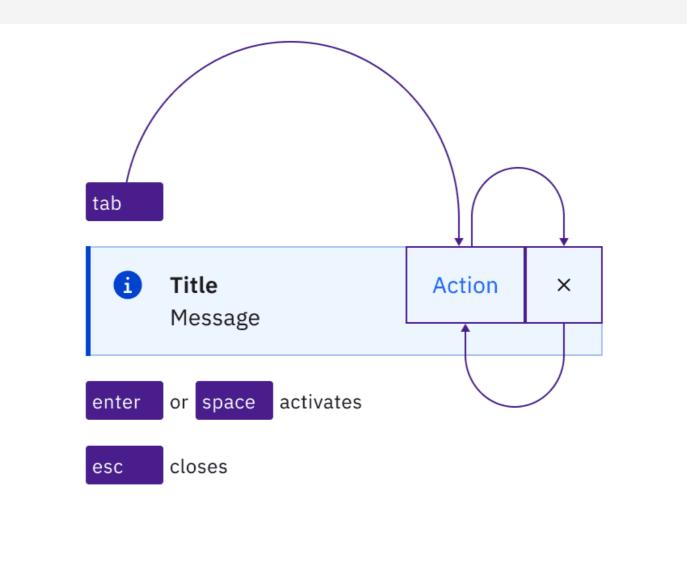
 \sim

 \sim

GitHub

Actionable

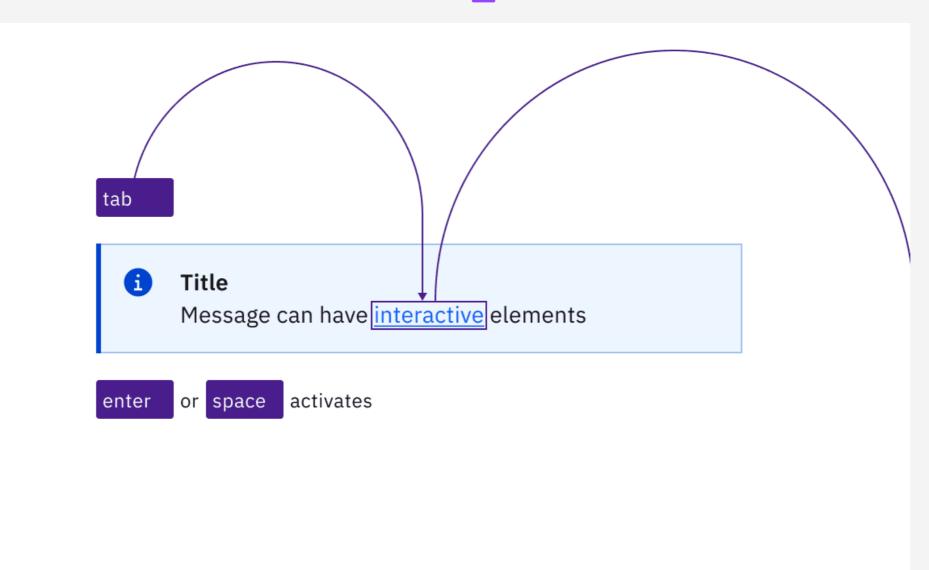
Actionable notifications may contain interactive elements such as links and buttons. This component grabs and traps focus until an action is taken or the notification is dismissed. Users can navigate through the interactive elements using the Tab key, and actions like closing or activating buttons can be performed using the Space or Enter keys. Additionally, the notification can be optionally closed by pressing the Escape key.



Focus trapped until an action is taken or the notification is dismissed.

Callout

Callout is not automatically announced by screen readers. It can include interactive elements such as links, which users can navigate through using the Tab key. Additionally, actions to activate these links can be performed using the Space or Enter keys



Navigating through interactive elements using the Tab key,

Design recommendations

Semantic styling

When using semantic styling on notifications, it's crucial to ensure accessibility by providing clear, concise text. Colours used to indicate status should always be paired with icons to avoid reliance on colour alone.

Text

It is recommended that notification messages use plain text, as semantic styling such as bold or italic and structural elements like <1i> are not conveyed to users by screen readers. If semantic styling must be included, it should not be essential for understanding the notification

Icon and colours

Ensure that screen readers can access the icon within the Callout to convey the type or semantic meaning of the message. The icon should have descriptive alt text that provides context about the Callout's type (e.g., information or warning). This helps users understand the nature and importance of the message.

Development considerations

Use a role of alert, log, or status for notifications that do not require user action. For notifications that require user action, use a role of alertdialog.

Special care should be given to focus management for notifications with interactive elements or actions. Venturing beyond using a role of alert log status or alertdialog for event-driven notifications is not recommended at this time as presents unique challenges. If you choose to do so, there are two known possible approaches to consider and research on your own.

- Collect notifications in a persistent area in your application for users to be able to navigate to and take action on notifications.
- Render notifications in an already-existing region that can be accessed via a hotkey. Focus should jump to the notification region after the hotkey is invoked. Once the user has reached the end of the region, focus should return to the previously focused item in the document before the hotkey was invoked.

Neither approach is perfect, but with either one: ensure notifications are properly announced, respect user timeout preferences, and ultimately provide an easy way to be navigated to by keyboard/screenreader to take action.

Accessibility testing status ?

Latest version: | Framework: React (@carbon/react)

Component	Accessibility test	Status	Link to source code			
Notification	Default state	Tested	GitHub link 🖸			
	Advanced states	Tested				
	Keyboard navigation	Tested				
	Screen reader	Manually tested				
Learn more about tag and test meaning $ ightarrow$						

View all component accessibility status $\,
ightarrow \,$

Edit this page on GitHub

	otification: Code		Next Components: Number input
Priv Terr Acc		Medium Twitter	Have questions? Email us at <u>carbon@us.ibm.com</u> or open an issue on <u>GitHub.</u> React Components version ^1.64.0 Last updated 19 September 2024 Copyright © 2024 IBM

