

TELUS Component Library

Icon

Multi-Platform Component | [Figma UI KIT](#)



```
<Icon icon={Icons.EyeMasked} />
```

Introduction

A wrapper for svg icons. Takes an icon from a [UDS palette](#), and renders it with theming applied.

[Follow the appropriate instructions](#) to add this component in to your app.

Guidance

Design guidance on the usage and choice of Iconography is given in the [icons palette documentation](#) for each category of icons used in the TELUS brand.

Accessibility

- Pass an `accessibilityLabel` for icons that have a meaning that should be communicated to the user. This will be used by assistive technology such as screen readers.
- Don't repeat content in an `accessibilityLabel` that is already present in text. If an icon's `accessibilityLabel` would simply repeat the text next to it, it is better to treat the icon as decorative.
- If the `accessibilityLabel` prop is not passed, the icon will be treated as purely decorative and will be set as ignorable by assistive technology via the attribute `aria-hidden`.
- If an icon is to be included alongside text and has a meaning related to the text, pass the prop `scalesWithText` as true so that if a sighted users

increases their device's text size, the icon also scales to a size similar to the size they find comfortable for reading.

Platform considerations

In web-only projects, use the web palette build to import icons.

```
import Users from '@telus-uds/palette-  
allium/build/web/icons/Users'  
import { Icon } from '@telus-uds/components-web'  
  
const icon = <Icon icon={Users} accessibilityLabel="Works on Web"  
/>
```

In native apps and cross-platform projects, import the palette icons from the `rn` build instead of the `web` build. This will select an svg icon for web, and a native svg icon for native platforms.

! INFO

In order for this to work on native platforms, you will need to have `react-native-svg` installed as a dependency in your project.

```
import { Users } from '@telus-uds/palette-allium/build/rn/icons'  
import { Icon } from '@telus-uds/components-base'  
  
const icon = <Icon icon={Users} accessibilityLabel="Works on Web,  
iOS and Android" />
```

Props

Name	Type	Platform	Default	Description
scalesWithText	bool	standard	false	controls whether size should be proportionate to

Name	Type	Platform	Default	Description
				accessibility-related scaling.
style	object	standard	{}	Custom style object applied to the icon component. This is for overriding the icon style but for where desires style not available as token. Note: This prop is used in web. @example <Icon style={{ color: 'red' }} />
variant	variant	standard		System variant prop. See variants for more information.
tokens	tokens	standard		System tokens prop. See tokens for more information.
icon.*	elementType	standard		A valid UDS icon component imported from a UDS palette. @example `import { MyIcon } from '@telus-uds/palette'; <MyIcon brand="telus" />`

Tokens

In exceptional circumstances, the following tokens can be passed to this component to override its default styles. **Do not do this unless absolutely necessary.** [Read more about overriding styles.](#)

► View Tokens


Variants

`Icon` has a fixed size and colour and does not automatically inherit sizes or colour from surrounding text. Instead, components that use icons alongside text, such as `Link`, set the icon's size and colour tokens explicitly, to complement the styles of each variant.

There are two variants generally available for icons used outside of such components (`Size` and `Color`):

Size

- To change the size of an icon by setting the `size` variant to one of the following values: `micro`, `small`, `large`, `extraLarge` (with "normal" size being the default)



```
<StackView tokens={{ justifyContent: 'space-around' }} space={2}
direction="row">
  <Icon icon={Icons.EyeMasked} variant={{ size: 'micro' }} />
  <Icon icon={Icons.EyeMasked} variant={{ size: 'small' }} />
  <Icon icon={Icons.EyeMasked} />
  <Icon icon={Icons.EyeMasked} variant={{ size: 'large' }} />
  <Icon icon={Icons.EyeMasked} variant={{ size: 'extraLarge' }}
/>
</StackView>
```

Color

- To give an icon a more eye-catching purple colour, set variant `{ rank: 'primary' }`

Color token	Code	Description
<code>\$greyCharcoal</code>	<code>color: 'Default'</code>	Use as the default decorative/interactions icon colour

Color token	Code	Description
\$purpleTelus	<code>color: 'Brand'</code>	Use to add visual priority
\$greyShuttle	<code>color: 'Subtle'</code>	Use to visually minimize or deprioritize
\$greenTelus	<code>color: 'Success'</code>	Use to indicate successful and positive meaning
\$redDark	<code>color: 'Danger'</code>	Use to indicate critical, destructive, and negative meaning
\$amberDark	<code>color: 'Warning'</code>	Use to indicate caution
\$white	<code>color: 'Inverse'</code>	Use to display icons on darker backgrounds



```
<StackView tokens={{ justifyContent: 'space-around' }} space={2}
direction="row">
  <Icon icon={Icons.EyeMasked} />
  <Icon icon={Icons.EyeMasked} variant={{ color: 'brand' }} />
  <Icon icon={Icons.EyeMasked} variant={{ color: 'default' }} />
  <Icon icon={Icons.EyeMasked} variant={{ color: 'subtle' }} />
  <Icon icon={Icons.EyeMasked} variant={{ color: 'success' }} />
  <Icon icon={Icons.EyeMasked} variant={{ color: 'danger' }} />
  <Icon icon={Icons.EyeMasked} variant={{ color: 'warning' }} />
  <div style={{ backgroundColor: 'black', padding: 2 }}>
    <Icon
      icon={Icons.EyeMasked}
      variant={{ color: 'inverse' }}
      tokens={{ BackgroundColor: 'black' }}
    />{ ' ' }
  </div>
</StackView>
```

```
</div>  
</StackView>
```

Rank

! INFO

This is a deprecated variant that achieves the same purpose of adding visual priority as `variant={{color: Brand }}`. Please use the variant color whenever possible.

- To give an icon a more eye-catching purple colour, set variant `{ rank: 'primary' }`



```
<StackView space={2} direction="row">  
  <Icon icon={Icons.EyeMasked} />  
  <Icon icon={Icons.EyeMasked} variant={{ rank: 'primary' }} />  
</StackView>
```

Consult with the TELUS brand team if you wish to use other icon styles for a custom component. It is technically possible via the `tokens` prop but the brand team should be consulted to ensure the desired design is not off-brand or inconsistent.

Feedback

- Spotted a problem with this component? Raise an [issue on GitHub](#)
- See any [existing issues](#) for this component
- Contact the team on slack in [#ds-support](#)