TELUS Component Library

Sizing

The size of elements and the space between them is fundamental to visual design. UDS simplifies a lot of the decision making for designers by providing logical opinionated design tokens to help create clear and functional layouts with a consistent rhythm.

- Improve legibility
- Help reinforce visual hierarchy
- Reduce cognitive load
- Provide customers with a more consistent experience, even across platforms

Using size tokens

Every component in the UDS uses sizing tokens from the TELUS brand palette. These tokens are used in the theme to apply `padding`, `margin`, `height` and `width` for example `"paddingBottom": "{palette.size.size24}"` or `"itemIconSize": "{palette.size.size16}”`.

Spacing and layout

Sometimes, the sizes used for spacing a layout needs to be set on a case-by-case basis, but in a way that is consistently on-brand. To ease this, UDS has a **spacing scale** based on the highly practical 8pt grid: that is multiples of 8 pixels, but with a few justifiable exceptions to account for subtleties in some of the components.

Layout

Creating consistent layouts that make use of the spacing scale is easy: layout components take a `space` prop for the level of spacing to be applied.

This can be a number from `0` to `11`: pass `space={1}` and the smallest level of space will be used, while `space={11}` will use the largest. Pass

`space={0}` for no space.

- Box - A layout utility component that is used to create space (padding) around content.
- Spacer - An empty element used to create space between components.
- StackView - Evenly space child elements.
- StackWrap - An alternative to StackView where the spaced items are allowed to wrap.

Spacing scale

The full scale is shown below. Note that larger values are responsive (they will change size as you change your screen size).

0 / 0px
1 / 4px
2 / 8px
3 / 16px
4 / 24px
5 / 32px
6 / 40px
7 / 48px
8 / 64px
9 / 72px
10 / 96px
11 / 120px

Responsive spacing

Sometimes, a specific layout requires different spacing depending on the user's screen width: such as widening spacing on wider viewports, or varying spacing as a responsive grid layout does or doesn't collapse into different rows.

Every `space` prop that takes a spacing scale index number may also take an object of spacing scale index numbers keyed by the names of the viewports above which it should apply (`xs` = extra small, `sm` = small, `md` = medium, `lg` = large, `xl` = extra large).

For example, `space={{ xs: 1, lg: 3 }}` applies the smallest level of space at `xs`, `sm` and `md` viewports, and increases it to the third smallest at `lg` and `xl` viewports.

Advanced spacing options

An object passed to a `space` prop may also take an `options` object with the following properties:

- `subtract`: a number of pixels to subtract from spacing to account for another part of a layout; for example, to offset a variable border width, `space={{ space: 4, options: { subtract: borderWidth }}}` would apply the fourth smallest spacing scale size then subtract the current `borderWidth` value.
- `size`: for very rare cases where an exact pixel size is needed; such as working around an externally controlled asset. In such cases, try to use `subtract` rather than `size` wherever possible, for readability: `space={{ space 6, options: { subtract: iframePadding }}}` communicates intent much more clearly than `space={{ options: { size: 21 }}}`
- `variant`: not currently used in the main theme, this allows themes to define variants of the spacing scale, such as compact scales to allow internal tools to use narrower spacing.