TELUS Component Library

Getting started

Welcome to the developer documentation for building accessible digital experiences with the Universal Design System for the TELUS Brand! We'll help you implement the right components for your application.

Before installing, be sure to check the system requirements.

DANGER

Deprecation Notice: `@telus-uds/ds-allium` Package

The `@telus-uds/ds-allium` package will no longer be actively maintained or supported after 31st July 2023. As a result, we encourage all users to start transitioning away from this package. Going forward, we recommend using the new and improved method that works seamlessly for all users. Detailed instructions on how to implement the new approach can be found in our updated documentation.

To ensure a smooth and uninterrupted experience with UDS components, we kindly request that you update your implementation to use the new preferred approach at your earliest convenience. This change will enable you to benefit from improved efficiency, compatibility, and ongoing support. Please refer to the migration guide for more information and detailed steps on how to transition from `@telus-uds/ds-allium` to `@telus-uds/components-web`.

Determine your application requirements

CAUTION

Please note that not all features, properties, and variants are available across all brands and platforms. You must install the appropriate component package to build for your application. For example, a single-brand, web-only application may have a comprehensive collection of variants in order to build a colourful customer experience that best represent the brand. However, a multi-brand application may not need a whole suite of options, when the experience is more straight-forward and must track variants between brands in order to enable a build-once, deploy everywhere model.

The Universal Design System is packaged and distributed as multiple modules. While every application requires one of our palette, theme, and components packages, additional palettes and themes may be necessary based on the specific requirements of each application.

Follow the appropriate instructions based on your application requirements.

1. Build a web application
2. Build a mobile application

Build a web application

CAUTION

A multi-brand, web-only component may have a limited collection of features, properties, and variants to account for a more direct, transactional customer experience that must be available across multiple brands. Please be sure to check each component page to see which features are available.

Install

```
npm install @telus-uds/components-web @telus-uds/palette-allium
@telus-uds/theme-allium
```

Usage

1. Export component as a named export from root
2. Wrap the UDS component in an `BaseProvider`
3. Include the BaseProvider provider once at the root of your application (e.g. in App.jsx)

```
import { BaseProvider, Button } from '@telus-uds/components-web'
import alliumTheme from '@telus-uds/theme-allium'
...
  <BaseProvider defaultTheme={alliumTheme}>
    ...
    <Button onPress={() => {}}>Hello World!</Button>
    ...
  </BaseProvider>
...
```

Assets

Font and icon assets are distributed as part of the TELUS palette via the `@telus-uds/palette-allium` package. This package leverages style-dictionary to build distributions specific to a platform or technology. There are both web (`web`) and React Native (`rn`) distributions, in general you should use the one that matches your target platform/technology.

Fonts

If you're building a react web application we recommend you use the CDN font distribution. `@telus-uds/palette-allium` provides a css snippet to load these fonts for use with UDS, assuming you have a css loader set up for your project you can simply include the following snippet:

```
import '@telus-uds/palette-allium/build/web/fonts/fonts-cdn.css'
```

CAUTION

If you are using the Isomorphic Starter Kit and handle your styling via Styled Components, you may experience some issues with the aforementioned method. In this case, you may need to load the fonts with Styled Components' `createGlobalStyle` helper:

```
import { createGlobalStyle } from 'styled-components'
import * as alliumFonts from '@telus-uds/palette-
allium/build/web/fonts/fonts-cdn.css'

...

createGlobalStyle(`
  ...
  ${alliumFonts}
  ...
`)
```

Icons

Icon SVGs are made available as React components via the palette.

```
import { Award } from '@telus-uds/palette-allium/build/web/icons'
```

Icons usage is documented in more detail in the Icon component documentation.

Build a mobile application

Install

```
npm install @telus-uds/components-base @telus-uds/palette-allium
@telus-uds/theme-allium
```

Usage

1. Export component as a named export from root
2. Wrap the UDS component in an `BaseProvider`
3. Include the BaseProvider provider once at the root of your application (e.g. in App.jsx)

```
import { BaseProvider, Button } from '@telus-uds/components-base'
import alliumTheme from '@telus-uds/theme-allium'
...
  <BaseProvider defaultTheme={alliumTheme}>
    ...
    <Button onPress={() => {}}>Hello World!</Button>
    ...
  </BaseProvider>
...
```

Assets

Font and icon assets are distributed as part of the TELUS palette via the `@telus-uds/palette-allium` package. This package leverages style-dictionary to build distributions specific to a platform or technology. There are both web (`web`) and React Native (`rn`) distributions, in general you should use the one that matches your target platform/technology. An important thing to remember is that in addition to the peer dependencies mentioned above, you'll also have to install `react-native-svg` :

```
npm install react-native-svg
```

Fonts

After the installation step, import the useFonts hook from expo-font package in your project. The hook keeps track of the loading state of the font. When an app is initialized, the hook loads the map of fonts. Let's take a look at a minimal example that uses `useFonts` to load fonts.

```
  ...
  import { useFonts } from 'expo-font'
  ...
  export default function App() {
    const [fontsLoaded] = useFonts({
      HelveticaNow300normal: require('@telus-uds/palette-
allium/build/rn/fonts/HelveticaNow-300.otf'),
      HelveticaNow400normal: require('@telus-uds/palette-
allium/build/rn/fonts/HelveticaNowOTF-400.otf'),
      HelveticaNow500normal: require('@telus-uds/palette-
allium/build/rn/fonts/HelveticaNowOTF-500.otf'),
      HelveticaNow700normal: require('@telus-uds/palette-
allium/build/rn/fonts/HelveticaNowOTF-700.otf')
    })

    if (!fontsLoaded) {
      return null;
    }

    return (
      <BaseProvider defaultTheme={alliumTheme}>
        ...
        <Button onPress={() => {}}>Hello World!</Button>
        ...
      </BaseProvider>
    );
  }
```

DANGER

For bare React Native projects, it's crucial to verify that you've installed the expo-font package.

```
npx install-expo-modules@latest
npx expo install expo-font
```

Usually, incorporating custom fonts involves placing font files within the `assets/fonts` directory and linking the font assets . However, when working with UDS packages it's important to note that due to platform-specific font family naming conventions on iOS and Android, a straightforward linking approach will not load the fonts on iOS. For iOS, the PostScript name of the font serves as the font family. In contrast, within UDS components, we adopt a specific font family naming structure: `${fontName}${fontWeight}${fontStyle}`.

So for the fonts to load, leveraging the capabilities of the `expo-font` package is essential. This package not only streamlines font loading and management but also assists in reconciling the differences in font family naming conventions between iOS and Android. By using the expo-font package, you can ensure that your brand fonts are applied correctly across different platforms, maintaining design consistency and delivering a polished user experience.

Icons

Icons usage is documented in detail in the Icon component documentation.

Where to get help

We have a dedicated Help & Support page, so you should find everything you need over there.