# Building for multi-brand

Use UDS component libraries and theme packages to create a multi-brand application. In UDS, `@telus-uds/components-base` and `@telus-uds/components-web` offers brand-agnostic components, while the theme packages provide specific brand themes. We'll cover the usage for both React and React Native applications.

## Building a React application

### 1. Installation

Install the component library and the required theme packages for the desired brands:

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

### 2. Theme configuration

In your React application, create a `themes.js` file to handle theme configuration:

```
import { alliumTheme } from '@telus-uds/theme-allium'
import { koodoTheme } from '@telus-uds/theme-koodo'

// Determine the theme based on the environment variable
const themeName = process.env.REACT_APP_THEME // Use your
specific environment variable

// Configure the theme object based on the selected theme
let theme
if (themeName === 'allium') {
  theme = alliumTheme
```

```javascript
} else if (themeName === 'koodo') {
  theme = koodoTheme
} else {
  // Provide a default theme if needed
  theme = alliumTheme
}

export default theme
```

### 3. BaseProvider configuration

In your application's entry point file (e.g., index.js), import the necessary dependencies and configure the `BaseProvider`: javascript

```javascript
import React from 'react'
import { render } from 'react-dom'
import { BaseProvider } from '@telus-uds/components-web'
import theme from './themes' // Import the configured theme from the 'themes.js' file
import App from './App'

render(
  <BaseProvider defaultTheme={theme}>
    <App />
  </BaseProvider>,
  document.getElementById('root')
)
```

### 4. Component usage

Now you can use the brand-agnostic components provided by `@telus-uds/components-web` in your application. The components will automatically use the selected theme based on the configuration in the `BaseProvider`.

## Building a React Native application

### 1. Installation

Install the component library and the required theme packages for the desired brands:

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

## 2. Theme configuration

In your React application, create a `themes.js` file to handle theme configuration:

```
import { alliumTheme } from '@telus-uds/theme-allium'
import { koodoTheme } from '@telus-uds/theme-koodo'

// Determine the theme based on the environment variable
const themeName = process.env.THEME // Use your specific environment variable

// Configure the theme object based on the selected theme
let theme
if (themeName === 'allium') {
  theme = alliumTheme
} else if (themeName === 'koodo') {
  theme = koodoTheme
} else {
  // Provide a default theme if needed
  theme = alliumTheme
}

export default theme
```

## 3. BaseProvider configuration

In your application's entry point file (e.g., index.js), import the necessary dependencies and configure the `BaseProvider`: javascript

```
import React from 'react'
import { BaseProvider } from '@telus-uds/components-base'
import theme from './themes' // Import the configured theme from the 'themes.js' file
import HomeScreen from './screens/HomeScreen'

export default function App() {
  return (
    <BaseProvider theme={theme}>
      <HomeScreen />
    </BaseProvider>
  )
}
```

### 4. Component usage

Now you can use the brand-agnostic components provided by `@telus-uds/components-base` in your application. The components will automatically use the selected theme based on the configuration in the `BaseProvider`.

## Where to get help

- Join the #ds-support Slack channel to connect with the UDS team and community to troubleshoot issues or suggest improvements
- Please feel free to report a bug
- Schedule office hours to discuss a specific issue or concern

Edit this page