Connected Digital Propositions: DLS UIKit Integration

**DLS UIKit Integration**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **History** | | | | | |
| **Version** | **Date** | | **Author** | **Section** | **Changes** |
| 0.0.1 | 28-02-2017 | | Amit Kumar | All | Draft |
| 0.0.2 | 06-04-2017 | | Amit Kumar | All | Draft |
| 0.0.3 | 12-04-2017 | | Suraj Raj | All | Draft |
| **Author** | | Amit Kumar (amit.kumar\_5@philips.com) | | | |
| **Approver** | | Rahul(rahul@philips.com) | | | |

Content

1 Introduction 3

2 Dependencies 3

2.1 Add artifactory in repository list in source root build.gradle 3

2.2 Add uikit dependency in project build.gradle 3

3 Theme Setup 4

3.1 Set Theme in Application/Activity in AndroidManifest to 4

3.2 Inject new Theme Configuration in Activity, onCreate() of Activity 4

3.3 Inject Fonts in Application class 4

4 Integrating DLS with non-DLS UIKit 5

4.1 Adding DLS style time/date picker 5

4.2 Override with DLS attributes Refer *Theme.DLS* for attributes styled with DLS color palette. 5

5 Integrating Language pack with UIKit 5

# **Introduction**

UIKit provides DLS implementation for Philips visual designs. This document provides guidelines for integrating UIKit lib in proposition apps.

Source code is hosted at [TFS](http://tfsemea1.ta.philips.com:8080/tfs/TPC_Region24/CDP2/_git/uid-android).

# **Dependencies**

UIKit can be integrated via artifactory dependency management, having advantage for automatic resolution of lib dependency modules.

## Add artifactory in repository list in source root build.gradle

|  |
| --- |
| repositories {  ...........  maven { url 'http://maartensmini.ddns.htc.nl.philips.com:8081/artifactory/jcenter' }  .........  } |

## Add uikit dependency in project build.gradle

|  |
| --- |
| compile 'com.philips.platform.uid:uikitdls:x.x.x' |

# **Theme Setup**

Inputs for selecting theme.

1. **TonalRange** definition: public enum ContentColor {ULTRA\_LIGHT, VERY\_LIGHT, LIGHT, BRIGHT, VERY\_DARK }
2. **NavigationColor** definition: public enum NavigationColor {ULTRA\_LIGHT, VERY\_LIGHT, LIGHT, BRIGHT, VERY\_DARK }

Using the above parameters we can create theme.

## Set Theme in Application/Activity in AndroidManifest to

|  |
| --- |
| **android:theme="@style/Theme.DLS.GroupBlue.UltraLight"** |

Possible Colors: GroupBlue, Blue, Aqua, Green, Orange, Pink, Purple Gray

Possible tonal Ranges: UltraLight, VeryLight, Light, Bright, VeryDark

## Inject new Theme Configuration in Activity, onCreate() of Activity

|  |
| --- |
| @Overrideprotected void onCreate(Bundle savedInstanceState) {  ***UITHelper.init(new ThemeConfiguration(ContentColor.ULTRA\_LIGHT, NavigationColor.ULTRA\_LIGHT, this));***super.onCreate(savedInstanceState);  //Your code goes here  } |

## Inject Fonts in Application class

|  |
| --- |
| @Override protected void attachBaseContext(final Context newBase) {  super.attachBaseContext(CalligraphyContextWrapper.wrap(newBase)); } |

# **Integrating DLS with non-DLS UIKit**

It follows the same steps as normal integration of DLS with extra steps to inject non-DLS theme.  
All below calls must be made before super.onCreate.

1. Set base theme as DLS or DLS derived theme.
2. Inject DLS content and navigation dependencies.

UIDHelper.*init*(new ThemeConfiguration(ContentColor.*ULTRA\_LIGHT* ,NavigationColor.*ULTRA\_LIGHT*,this);

1. Inject non-dls (or extened) themein Activity#onCreate before super.onCreate

getTheme().applyStyle(R.style.Theme\_Philips\_LightBlue, true);

(for example for light blue theme)

## Adding DLS style time/date picker

Override below in your style which extends non-DLS theme.

<item name="android:timePickerDialogTheme" tools:targetApi="lollipop">  
 @style/UIDDatePickerDialogTheme  
</item>  
<item name="android:datePickerDialogTheme" tools:targetApi="lollipop">  
 @style/UIDDatePickerDialogTheme  
</item>

## Override with DLS attributes Refer ***Theme.DLS*** for attributes styled with DLS color palette.

# **Integrating Language pack with UIKit**

To integrate Language packs in your application using UIKIT, please follow the below steps.

1. Extend your activity with UIDActivity



1. Set the path of your JSON file location, which contains the key-value string translations, this needs to be set each time the new JSON is downloaded.



1. Translations in all Views (DLS and Android native) implementing the setText(), setHint(), etc would be handled by UIKit library based on the string resourceID used in JSON
2. Translations in all View (DLS only) implementing android:text=”@string/” through the layout would be handled by UIKit library based on the string resourceID used in JSON

Appendix

List of available components with UIKit.

|  |  |
| --- | --- |
| **Control** | **Class** |
| Button | com.philips.platform.uid.view.widget.Button |
| EditText | com.philips.platform.uid.view.widget.EditText |
| Switch | com.philips.platform.uid.view.widget.Switch |
| CheckBox | com.philips.platform.uid.view.widget.CheckBox |
| ProgressBar | com.philips.platform.uid.view.widget.ProgressBar |
| IndeterminateProgressBar | com.philips.platform.uid.view.widget.IndeterminateProgressBar |
| Label | com.philips.platform.uid.view.widget.Label |