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 |
| 1.0 | 12-04-2017 | | Suraj Raj | All | Add completed features |
| 1.1 | 30-10-2017 | | Suraj Raj | All | Update completed features |
| **Author** | | Suraj Raj (suraj.raj@philips.com) | | | |
| **Approver** | | Amit Kumar (amit.kumar\_5@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 IconFont support 4

4 Theme Setup 4

4.1 Set Theme in Application 4

4.2 Set Theme in Activity 5

4.3 Inject Fonts in Application class 5

5 Integrating DLS with non-DLS UIKit 6

5.1 Adding DLS style time/date picker 6

5.2 Override with DLS attributes Refer *Theme.DLS* for attributes styled with DLS color palette. 6

6 Integrating Language pack with UIKit 6

7 Integrating Sidebar with UIKit 7

8 Integrating Bottom TabBar with UIKit 11

# **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://artifactory-ehv.ta.philips.com:8082/artifactory/platform-pkgs-android-snapshot](http://artifactory-ehv.ta.philips.com:8082/artifactory/platform-pkgs-android-snapshot/com/philips/cdp/uid/0.1.1-SNAPSHOT.20170706172130/uid-0.1.1-SNAPSHOT.20170706172130.aar)’ }  .........  } |

## Add uikit dependency in project build.gradle

Please refer artifactory for the latest releases. It’s recommended to use scm plugin instead of hardcoded version to get the latest snapshot or stable versions.

compile(group: 'com.philips.cdp', name: 'uid', version: '1.0.0', ext: 'aar', changing: true)

# **IconFont support**

UIKit integrates dls icons in font. Please refer attached document for using ttf.



Apps don’t need to add icons dependency and its provided with UIKit.

# **Theme Setup**

Inputs for selecting theme.

1. **TonalRange** definition: public enum ContentColor {ULTRA\_LIGHT, VERY\_LIGHT, BRIGHT, VERY\_DARK }
2. **NavigationColor** definition: public enum NavigationColor {ULTRA\_LIGHT, VERY\_LIGHT, BRIGHT, VERY\_DARK }
3. **AccentRange:** public enum AccentRange {GROUP\_BLUE, AQUA, GREEN, ORANGE, PINK, PURPLE, GRAY }

Using the above parameters, we can create theme.

## Set Theme in Application

**Target activity (expecting DLS theme from Application) must not set any theme in AndroidManifest and must extend UIDActivity.**

Set DLS (or DLS derived) theme in manifest under application tag.

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

Extend Application class and inject other UID dependencies in onCreate of custom Application class.

|  |
| --- |
| ***UITHelper.init(new ThemeConfiguration(this, ContentColor.ULTRA\_LIGHT, NavigationColor.ULTRA\_LIGHT, AccentRange.GROUP\_BLUE));*** |

## Set Theme in Activity

1. Set theme in AndroidManifest.

|  |
| --- |
| **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

1. Inject new Theme Configuration in Activity, onCreate() of Activity. Provide different configurations for navigation colors, content colors and accent colors.  
   This api doesn’t set theme but only manipulates different colors.

|  |
| --- |
| @Overrideprotected void onCreate(Bundle savedInstanceState) {  ***UITHelper.init(new ThemeConfiguration(this, ContentColor.ULTRA\_LIGHT, NavigationColor.ULTRA\_LIGHT, AccentRange.GROUP\_BLUE));***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 calling super.onCreate.

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

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

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 the 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

# **Integrating SideBar with UIKit**

To integrate SideBar in the application using UIKIT, please follow the below steps.

1. **UID Sidebar is custom DrawerLayout as per DLS design.  
   Use Sidebar in your xml layout file similar to DrawerLayout as shown below in sample code.**<com.philips.platform.uid.view.widget.SideBar  
    android:id="@+id/sidebar\_layout"  
    android:layout\_width="match\_parent"  
    android:layout\_height="match\_parent"  
    android:fitsSystemWindows="true">

         <  ------  Your main content layout with toolbar etc..  -----  />

        <android.support.design.widget.NavigationView  
          android:id="@+id/sidebar\_left\_root"  
          android:layout\_width="match\_parent"  
          android:layout\_height="match\_parent"  
          android:layout\_gravity="start"  
          **android:theme="@style/Theme.DLS.GroupBlue.Bright">**

**< ! - - To make your Sidebar Them-able, you must have to use "theme" attribute as shown above  - - > < ! - -  For any other layout other than NavigationView, you also need to use "background" attribute as below - - >**

**android:background="?attr/uidContentPrimaryBackgroundColor"**

          </android.support.design.widget.NavigationView>

</com.philips.platform.uid.view.widget.SideBar>

2.  **If you are using ListView in your NavigationView, then by default it will be supporting theme you have set for Content Area.**

**If you want to make your ListView to support Navigation Area mapped Theme, then you can use below sample code for ListViewAdapter.**

public class SidebarListViewAdapter extends ArrayAdapter<String> {  
     private int resID;  
     private LayoutInflater inflater;  
  
     public SidebarListViewAdapter(@NonNull Context context, @LayoutRes int resource, String[] values, boolean isNavigationContext) {  
         super(context, resource, values);  
         resID = resource;  
         inflater = (LayoutInflater) context.getSystemService(Context.LAYOUT\_INFLATER\_SERVICE);  
         **if(isNavigationContext)**  
                **inflater = inflater.cloneInContext(UIDHelper.getNavigationThemedContext(context));**  
}  
  
@NonNull  
@Override  
public View getView(int position, @Nullable View convertView, @NonNull ViewGroup parent) {  
      Label view;  
      if (convertView == null) {  
          view = (Label) inflater.inflate(resID, parent, false);  
      } else {  
         view = (Label) convertView;  
      }  
     view.setText(getItem(position));  
     return view;  
     }  
}

3. **If you are using RecyclerView in your NavigationView, and want to support Navigation Area mapped theme, then you can use below sample code.**

RecyclerViewSeparatorItemDecoration navigationThemedSeparatorItemDecoration = **new RecyclerViewSeparatorItemDecoration(UIDHelper.getNavigationThemedContext(context));**

DataHolderView navigationThemedDataHolderView = **getIconDataHolderView(UIDHelper.getNavigationThemedContext(context));**

private class SidebarRecyclerViewAdapter extends RecyclerView.Adapter {  
  
private ObservableArrayList<DataHolder> dataHolders;  
private LayoutInflater inflater;  
private boolean isNavigationContext;  
  
private SidebarRecyclerViewAdapter(@NonNull final ObservableArrayList<DataHolder> dataHolders, boolean isNavigationContext) {  
     this.dataHolders = dataHolders;  
     this.isNavigationContext = isNavigationContext;  
}  
  
@Override  
public RecyclerView.ViewHolder onCreateViewHolder(@NonNull final ViewGroup parent, final int viewType) {  
     inflater = (LayoutInflater) parent.getContext().getSystemService(Context.LAYOUT\_INFLATER\_SERVICE);  
     **if(isNavigationContext)**  
**inflater = inflater.cloneInContext(UIDHelper.getNavigationThemedContext(parent.getContext()));**  
     View v = inflater.inflate(R.layout.sidebar\_recyclerview\_item, parent, false);  
  
return new SidebarRecyclerViewBindingHolder(v);  
}  
  
@Override  
public void onBindViewHolder(@NonNull final RecyclerView.ViewHolder holder, final int position) {  
       final DataHolder dataHolder = dataHolders.get(position);

       holder.itemView.setSelected(leftRecyclerViewSelectedPosition == position);

((SidebarRecyclerViewBindingHolder) holder).itemView.setOnClickListener(new View.OnClickListener() {  
@Override  
public void onClick(final View v) {

}  
});  
}  
  
@Override  
public int getItemCount() {  
    return dataHolders.size();  
}

4. **If you are just using TextView as your list item, you can use below sample code for your list item layout.**

<com.philips.platform.uid.view.widget.Label  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 style="@style/UIDLabel.ListItemText"  
 android:layout\_width="match\_parent"  
 android:layout\_height="@dimen/uid\_recyclerview\_item\_one\_line\_height"  
 android:paddingEnd="@dimen/uid\_recyclerview\_margin\_left\_right"  
 android:paddingStart="@dimen/uid\_recyclerview\_margin\_left\_right"  
 android:gravity="start|center\_vertical"  
 android:background="@drawable/uid\_list\_item\_selector"/>

# **Integrating Bottom TabBar with UIKit**

To integrate Bottom TabBar in the application using UIKIT, you can use below sample code:

**UID Bottom TabBar is custom TabLayout as per DLS design.  
Use Bottom TabBar in your xml layout file similar to TabLayout as shown below in sample code.**

There are two variants of Bottom TabBar:

1. **Icon Only Layout:**

<**com.philips.platform.uid.view.widget.TabLayout  
 android:id="@+id/tab\_layout\_icon\_only"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_alignParentBottom="true"** >  
  
 <**com.philips.platform.uid.view.widget.UIDTabItem  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:src="@drawable/search\_icon"**/>

</**com.philips.platform.uid.view.widget.TabLayout**>

1. **Icon With Title Layout**: Here you have to use style **"@style/UIDBottomTabLayout.Title"** as shown below, to make it work with title.

<**com.philips.platform.uid.view.widget.TabLayout  
 android:id="@+id/tab\_layout\_with\_title"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_alignParentBottom="true"  
 style="@style/UIDBottomTabLayout.Title"** >  
  
 <**com.philips.platform.uid.view.widget.UIDTabItem  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:src="@drawable/search\_icon"  
 android:text="@string/search"** />

</**com.philips.platform.uid.view.widget.TabLayout**>

1. **To make your TabLayout theme-able you need to use,**

**eg. android:theme="@style/Theme.DLS.GroupBlue.VeryDark"**

1. **There are list of custom attributes as shown below, which you can use as per your requirement:**

|  |  |
| --- | --- |
| TabLayout Height | app:uidTabItemPreferredHeight |
| UIDTabItem Text | android:text |
| UIDTabItem Text Color | android:textColor |
| UIDTabItem Icon Source | android:src |
| UIDTabItem Icon Tint Color | android:tint |
| UIDTabItem Icon Tint Mode | android:tintMode |
| UIDTabItem Notification Badge | app:uidTabItemNotificationBadgeCount |

1. **Till now we have seen, how to create TabLayout & UIDTabItem through xml layout file.**

**You can also create it from code as shown below:**

**com.philips.platform.uid.view.widget.TabLayout tabLayout = new TabLayout();**

You can create UIDTabItem as shown below:

If you want icon only layout, pass false as second parameter in constructor as below:

***UIDTabItem uidTabItem = new UIDTabItem(getContext(), false);***

If you want icon with title layout, pass true as second parameter in constructor as below:

***UIDTabItem uidTabItem = new UIDTabItem(getContext(), true);***

After creating UIDTabItem, you can add it in TabLayout as below:

***tabLayout.addView(uidTabItem);***

# **Integrating Quite Expander**

To integrate Expander in the application using UIKIT, you can use below sample code:

**UID Expander is a custom Layout as per DLS design.  
Use Bottom Expander in your xml layout file similar to other control as shown below in sample code.**

**Note: Theme of Quite Expander is not be set, it takes the theme of parent view.**

*<?***xml version="1.0" encoding="utf-8"***?>*<**LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:orientation="vertical" android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"**>  
  
  
 <**com.philips.platform.uid.view.widget.Expander  
 android:id="@+id/layout\_expander\_id\_test"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"**>  
 </**com.philips.platform.uid.view.widget.Expander**>  
  
</**LinearLayout**>

**It has two view internally:**

1. **Top Title View (Required & Clickable)**
2. **Bottom Content View (Optional)**

**When Initialized Title view is set by a default xml view ‘uid\_expander\_title\_layout\_default.xml’ , however user can override it with** setExpanderCustomPanelView() API

**This content view needs to always set by user using** setExpanderContentView()

**API**

**Title left icon can be set by API** setExpanderPanelIcon(Drawable icon)

**Title text can be set by API** setExpanderTitle(String title)

**Expander can be expanded/collapsed by code using API** expand(**boolean** aBoolean)

**And current state can be fetched using API boolean** isExpanded()

**To customize default title text API** Label getTitleLabel() can be used which returns Label.

**There is a Listener interface** UIDExpanderListener **which when set gives the expand and collapse completion callbacks.**

API for setting listener: setExpanderListener(UIDExpanderListener uidExpanderListener)

**Note: once user sets customize title panel using API** setExpanderCustomPanelView() then these APIs will be void:

* setExpanderPanelIcon(Drawable icon)
* setExpanderTitle(String title)
* getTitleLabel()
* setExpanderListener(UIDExpanderListener uidExpanderListener)

**Note: Please refer sample app** ExpanderFragment **class for reference**

**Confluence Reference:** [**https://confluence.atlas.philips.com/display/UIT/Expander+%7C+1.0.0**](https://confluence.atlas.philips.com/display/UIT/Expander+%7C+1.0.0)

Appendix

List of available components with UIKit.

|  |  |
| --- | --- |
| **Control** | **Class** |
| AlertDialogFragment | com.philips.platform.uid.view.widget.AlertDialogFragment |
| Button | com.philips.platform.uid.view.widget.Button |
| CheckBox | com.philips.platform.uid.view.widget.CheckBox |
| DotNavigation | com.philips.platform.uid.view.widget.DotNavigationIndicator |
| EditText | com.philips.platform.uid.view.widget.EditText |
| Hyperlink | Use Label for Hyperlinks |
| IndeterminateProgressBar | com.philips.platform.uid.view.widget.IndeterminateProgressBar |
| Label | com.philips.platform.uid.view.widget.Label |
| NotificationBadge | com.philips.platform.uid.view.widget.NotificationBadge |
| ProgressBar | com.philips.platform.uid.view.widget.ProgressBar |
| ProgressBarButton | com.philips.platform.uid.view.widget. ProgressBarButton |
| ProgressBarWithLabel | com.philips.platform.uid.view.widget.ProgressBarWithLabel |
| RadioButton | com.philips.platform.uid.view.widget.RadioButton |
| RadioGroup | com.philips.platform.uid.view.widget.RadioGroup |
| RatingBar | com.philips.platform.uid.view.widget.RatingBar |
| SearchBox | com.philips.platform.uid.view.widget.SearchBox |
| Switch | com.philips.platform.uid.view.widget.Switch |
| SideBar | com.philips.platform.uid.view.widget.SideBar |
| UI Picker | com.philips.platform.uid.view.widget.UIPicker |
| About Screen | com.philips.platform.uid.view.widget.AboutScreen |
| SplashScreen | com.philips.platform.uid.view.widget.SplashScreen |
| Slider | com.philips.platform.uid.view.widget.Slider |
| DiscreteSlider | com.philips.platform.uid.view.widget.DiscreteSlider |
| Bottom TabBar | com.philips.platform.uid.view.widget.TabLayout |