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 | 07-06-2017 | | 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 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 5

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://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)

# **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/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

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);  
       ((SidebarRecyclerViewBindingHolder) holder).getBinding().setVariable(1, dataHolder);  
       ((SidebarRecyclerViewBindingHolder) holder).getBinding().executePendingBindings();  
  
       holder.itemView.post(new Runnable() {  
       @Override  
       public void run() {  
       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();  
}  
  
class SidebarRecyclerViewBindingHolder extends RecyclerView.ViewHolder {  
     private ViewDataBinding binding;  
  
     SidebarRecyclerViewBindingHolder(@NonNull View rowView) {  
          super(rowView);  
          binding = DataBindingUtil.bind(rowView);  
     }  
  
     public ViewDataBinding getBinding() {  
          return binding;  
     }  
   }  
}

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"/>

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 |