**App Framework Android Integration**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Document History** | | | | |
| **Version** | **Date** | **Author** | **Section** | **Changes** |
| 0.1 | 09-Aug-2016 | Ritesh Jha, Rakesh Krishamurthy | All | Initial draft |
| 0.2 | 22-Sept-2016 | Richa ,Spoorti | All | Initial draft |

|  |  |
| --- | --- |
| **Author** | Richa, Spoorti |
| **Approved By** |  |
| Email Id | richa.bajpai@philips.com , spoorti.hallur@philips.com |

Contents

[1. Introduction 2](#_Toc462321875)

[2. Integration 6](#_Toc462321876)

[2.1 Artifactory 6](#_Toc462321877)

[2.2 Root gradle changes 6](#_Toc462321878)

[2.3 Dependencies 8](#_Toc462321879)

[2.3.1 Gradle dependencies 8](#_Toc462321880)

[2.3.2 Library dependencies 8](#_Toc462321881)

[2.5 Some more dependencies 9](#_Toc462321882)

[2.6 Proxy dependencies 10](#_Toc462321883)

[2.7. Prerequisites 10](#_Toc462321884)

[3. Steps for Integration of Common Components 10](#_Toc462321885)

[4.Initializing & Launching a common component 11](#_Toc462321886)

[5. Handling Back key from coco 12](#_Toc462321887)

[6. Manifest Details 12](#_Toc462321888)

[Kindly find manifest details in source folder to find the various Android Components added and used. 12](#_Toc462321889)

[6. Notes 12](#_Toc462321890)

# Introduction

This document will provide step by step procedure to integrate various horizontal components within **AppFramework** application.

AppFramework has following features integrated:

* Introduction / On boarding screens
* Settings
* User Registration
* Order History
* Digital Care
* Product Registration
* In App Purchase shopping catalogue
* Shopping cart
* About the application

**Architectural Over-View:**

AppFramework follows MVP architecture and UI-State-Pattern.

Details:

1. User interface consists of the Model, view and Presenter.
2. Every View is associated with a Presenter and a State.
3. **UIFlowManager:** Flow-Manager is a singleton class and the whole responsibility of it is to control the navigations across the various modules

**public class** UIFlowManager {  
 UIState **currentState**;  
  
 */\*\* getter for current state  
 \*  
 \** ***@return*** *\*/* **public** UIState getCurrentState() {  
 **return currentState**;  
 }  
  
 */\*\*  
 \* setter for current state  
 \** ***@param currentState*** *\*/* **public void** setCurrentState(UIState currentState) {  
 **this**.**currentState** = currentState;  
 }  
  
 *//* ***TODO: Presenter should be represented as an interface*** */\*\*  
 \* For naviating to next state  
 \** ***@param uiState*** *requires Uistate object  
 \** ***@param context*** *requires context  
 \*/* **public void** navigateToState(UIState uiState, Context context) {  
 uiState.navigate(context);  
 setCurrentState(uiState);  
 }  
}

1. **State Class**: Each screen of the app has a state and a state class is supposed to extend UIstate Class and implement the abstract methods for navigation and back handling

For eg :

**public class** SettingsFragmentState **extends** UIState {  
 */\*\*  
 \* constructor  
 \** ***@param stateID*** *\*/* **public** SettingsFragmentState(@UIStateDef **int** stateID) {  
 **super**(stateID);  
 }  
  
 */\*\*  
 \* to navigate  
 \** ***@param context*** *requires context  
 \*/* @Override  
 **public void** navigate(Context context) {  
 **if**(context **instanceof** HomeActivity) {  
 ((AppFrameworkBaseActivity) context).showFragment(**new** SettingsFragment(), SettingsFragment.***TAG***);  
 }  
 }  
  
 */\*\*  
 \* to handle back  
 \** ***@param context*** *requires context  
 \*/* @Override  
 **public void** back(**final** Context context) {  
 ((AppFrameworkBaseActivity)context).popBackTillHomeFragment();  
 }  
}

**UIState class**

Below are the methods exposed by UiState class:  
  
 **public** UIState(@UIState.UIStateDef **int** stateID){  
 **this**.**stateID** = stateID;  
 }  
  
 @UIState.UIStateDef  
 **public int** getStateID() {  
 **return stateID**;  
 }  
@UIState.UIStateDef  
 **public void** setStateID(**int** stateID) {  
 **this**.**stateID** = stateID;  
 }  
  
 */\*\*  
 \* For navigating from one state to other  
 \** ***@param context*** *requires context  
 \*/* **protected abstract void** navigate(Context context);  
  
 */\*\*  
 \* For going back to last state  
 \** ***@param context*** *requires context  
 \*/* **public abstract void** back(Context context);  
  
 */\*\*  
 \* to set the presenter  
 \** ***@param uiBasePresenter*** *\*/* **public void** setPresenter(UIBasePresenter uiBasePresenter){  
 **this**.**uiBasePresenter** = uiBasePresenter;  
 }  
  
 */\*\*  
 \* to get the presenter object  
 \** ***@return*** *\*/* **public** UIBasePresenter getPresenter(){  
 **return uiBasePresenter**;  
 }

**5. Presenter Class**: Presenter creates a State, sets the presenter to the state and using flowManager navigates to the corresponding state. Each Fragment must have a presenter which extends the presenter base class UiBasePresenter and it mush implement its methods to performing operations on loading the fragment or on click events inside the fragment

**public class** SettingsFragmentPresenter **extends** UIBasePresenter **implements** UserRegistrationState.SetStateCallBack {  
  
 SettingsFragmentPresenter(){  
 setState(UIState.***UI\_SETTINGS\_FRAGMENT\_STATE***);  
 }  
  
 AppFrameworkApplication **appFrameworkApplication**;  
 UIState **uiState**;  
  
 @Override  
 **public void** onClick(**int** componentID, Context context) {  
 **appFrameworkApplication** = (AppFrameworkApplication) context.getApplicationContext();  
 **switch** (componentID){  
 **case** SettingsFragment.***logOutButton***:  
 **uiState** = **new** HomeFragmentState(UIState.***UI\_HOME\_FRAGMENT\_STATE***);  
 **uiState**.setPresenter(**this**);  
 **appFrameworkApplication**.getFlowManager().navigateToState(**uiState**,context);  
 **break**;  
 **case** SettingsAdapter.***iapHistoryLaunch***:  
 **uiState** = **new** InAppPurchaseHistoryFragmentState(UIState.***UI\_IAP\_SHOPPING\_HISTORY\_FRAGMENT\_STATE***);  
 **uiState**.setPresenter(**this**);  
 **appFrameworkApplication**.getFlowManager().navigateToState(**uiState**,context);  
 **break**;  
 }  
 }  
@Override  
 **public void** onLoad(Context context) {  
 **appFrameworkApplication** = (AppFrameworkApplication) context.getApplicationContext();  
 **uiState** = **new** UserRegistrationState(UIState.***UI\_USER\_REGISTRATION\_STATE***);  
 **uiState**.setPresenter(**this**);  
 ((UserRegistrationState)**uiState**).registerForNextState(**this**);  
 **appFrameworkApplication**.getFlowManager().navigateToState(**uiState**,context);  
 }  
  
 */\*\*  
 \* For setting the next state  
 \** ***@param context*** *\*/* @Override  
 **public void** setNextState(Context context) {  
 **appFrameworkApplication** = (AppFrameworkApplication) context.getApplicationContext();  
 **uiState** = **new** HomeActivityState(UIState.***UI\_HOME\_STATE***);  
 **uiState**.setPresenter(**this**);  
 ((HomeActivity)context).finishAffinity();  
 **appFrameworkApplication**.getFlowManager().navigateToState(**uiState**,context);  
 }

**Base class of Presenter is as below** :

**abstract public class** UIBasePresenter {  
 */\*\*  
 \* The onclick of objects in a particular state can be defined here  
 \** ***@param componentID*** *The Id of any button or widget or any other component  
 \** ***@param context*** *the current context  
 \*/* **public abstract void** onClick(**int** componentID, Context context);  
  
 */\*\*  
 \* This method needs to be implemented to do oeprations when te particular state loads  
 \** ***@param context*** *requires context  
 \*/* **public abstract void** onLoad(Context context);  
  
 */\*\*  
 \* For seeting the current state , so that flow manager is updated with current state  
 \** ***@param stateID*** *requires State ID  
 \*/* **public void** setState(**int** stateID){  
  
 }  
}

**Source code Links:**<http://maartens-mini.ddns.htc.nl.philips.com:8081/artifactory/simple/libs-release-local-android/com/philips/cdp/appFramework/0.1.0>

**BigBucket Link:**

https://atlas.natlab.research.philips.com/bitbucket/projects/MAF/repos/app-framework\_android/browse

# Integration

Integration can be done in following ways.

# 2.1 Artifactory

All dependent libraries should be downloaded from artifactory.

**Artifactory path:**

[http://maartens-mini.ddns.htc.nl.philips.com:8081/artifactory/simple/libs-release-local-android/com/philips/cdp/appFramework/](http://maartens-mini.ddns.htc.nl.philips.com:8081/artifactory/simple/libs-stage-local-android/com/philips/cdp/appFramework/)

# 2.2 Root gradle changes

buildscript {  
 repositories {  
  
 maven { url 'http://maartens-mini.ddns.htc.nl.philips.com:8081/artifactory/jcenter' }  
  
 }  
 dependencies {  
 classpath 'com.android.tools.build:gradle:2.1.0'  
  
 // NOTE: Do not place your application dependencies here; they belong  
 // in the individual module build.gradle files  
 }  
}  
  
  
  
  
allprojects {  
 repositories {  
  
  
 maven { url 'http://maartens-mini.ddns.htc.nl.philips.com:8081/artifactory/libs-release-local-android' }  
 maven { url 'http://maartens-mini.ddns.htc.nl.philips.com:8081/artifactory/jcenter' }  
 maven { url 'http://maartens-mini.ddns.htc.nl.philips.com:8081/artifactory/ext-release-local'}  
 maven { url 'http://maartens-mini.ddns.htc.nl.philips.com:8081/artifactory/libs-release-local-android' }  
 maven { url 'http://maartens-mini.ddns.htc.nl.philips.com:8081/artifactory/libs-stage-local-android'}  
 maven { url 'http://maartens-mini.ddns.htc.nl.philips.com:8081/artifactory/libs-snapshot-local-android' }   
 maven { url "https://oss.sonatype.org/content/repositories/snapshots" }  
  
 }  
}  
  
  
task clean(type: Delete) {  
 delete rootProject.buildDir  
}

# 2.3 Dependencies

# 2.3.1 Gradle dependencies

dependencies {  
  
 testCompile('org.mockito:mockito-core:1.10.19')  
 testCompile 'org.robolectric:robolectric:3.0'  
 testCompile 'junit:junit:4.12'  
  
 compile 'com.android.support:support-annotations:23.2.0'  
 compile 'com.android.support:multidex:1.0.1'  
 compile 'com.android.support:design:23.1.1'  
 compile 'com.android.support:appcompat-v7:+'  
  
 compile(group: 'com.philips.cdp', name: 'product-registration-lib', version: '1.2.0-SNAPSHOT', ext: 'aar', changing: true) {  
 transitive = true  
 }  
   
 compile(group: 'com.philips.cdp', name: 'digitalCare', version: '6.2.0-SNAPSHOT', ext: 'aar', changing: true){  
 transitive=true  
 }  
  
 compile(group: 'com.philips.cdp', name: 'iap', version: '4.2.0-SNAPSHOT', ext: 'aar', changing: true){  
 transitive=true  
 }  
   
 compile(group: 'com.philips.cdp', name: 'dicommClientLib', version: '1.2.2', ext: 'aar')  
  
 compile(group: 'com.philips.cdp', name: 'shinelib', version: '2.0.0', ext: 'aar')  
  
  
}

# 2.3.2 Library dependencies

1. User Registration : 1.2.0-SNAPSHOT
2. Uikit Library : 3.3.0-SNAPSHOT
3. Product Registration : 1.2.0 - SNAPSHOT
4. Digital Care : 6.2.0-SNAPSHOT
5. InAppPurchase : 4.2.0-SNAPSHOT
6. AppInfra : 1.2.0 - SNAPSHOT

2.4 Package Options

packagingOptions {  
 exclude 'META-INF/DEPENDENCIES.txt'  
 exclude 'META-INF/LICENSE.txt'  
 exclude 'META-INF/NOTICE.txt'  
 exclude 'META-INF/NOTICE'  
 exclude 'META-INF/LICENSE'  
 exclude 'META-INF/DEPENDENCIES'  
 exclude 'META-INF/notice.txt'  
 exclude 'META-INF/license.txt'  
 exclude 'META-INF/dependencies.txt'  
 exclude 'META-INF/LGPL2.1'  
  
 pickFirst 'lib/mips/librsjni.so'  
 pickFirst 'lib/mips/libblasV8.so'  
 pickFirst 'lib/mips/libRSSupport.so'  
 pickFirst 'lib/x86/librsjni.so'  
 pickFirst 'lib/x86/libblasV8.so'  
 pickFirst 'lib/x86/libRSSupport.so'  
 pickFirst 'lib/armeabi-v7a/librsjni.so'  
 pickFirst 'lib/armeabi-v7a/libblasV8.so'  
 pickFirst 'lib/armeabi-v7a/libRSSupport.so'  
 pickFirst 'lib/arm64-v8a/libRSSupport.so'  
 pickFirst 'lib/arm64-v8a/librsjni.so'  
 exclude 'META-INF/INDEX.LIST'  
}

# 2.5 Some more dependencies

minSdkVersion 19

targetSdkVersion 23

compileSdkVersion 23

buildToolsVersion "23.0.3"

# 2.6 Proxy dependencies

Gradle dependencies can get some network/proxy related issues with Philips. In order to fix this issue, we are using below proxy settings in gradle.properties of root folder.

**systemProp.https.proxyHost**=**42.99.164.34**

**systemProp.https.proxyPort**=**10015**

We are using these proxy settings locally. But Eindhoven, does not use above proxy settings.

## Prerequisites

NA

# Steps for Integration of Common Components

**Any invocation in the application context of the app can be done by creating an application class and adding them there. Following are some sample codes to initialize elements for ActivityLife cycle callbacks and Product Registration**

**Eg: Following is the Application class used to initialize Product Registration component :**

public class AppFrameworkApplication extends Application {  
 public UIFlowManager flowManager;  
 private static Context mContext;  
 public static AppInfraInterface gAppInfra;  
 public static LoggingInterface loggingInterface;  
 private IAPInterface iapInterface;  
  
 @SuppressWarnings("deprecation")  
 @Override  
 public void onCreate() {  
 MultiDex.install(this);  
 super.onCreate();  
 mContext = getApplicationContext();  
 flowManager = new UIFlowManager();  
 gAppInfra = new AppInfra.Builder().build(getApplicationContext());  
 loggingInterface = gAppInfra.getLogging().createInstanceForComponent(BuildConfig.APPLICATION\_ID, BuildConfig.VERSION\_NAME);  
 loggingInterface.enableConsoleLog(true);  
 loggingInterface.enableFileLog(true);  
 setLocale();  
 initializeUserRegistrationLibrary(Configuration.STAGING);  
 initializeProductRegistrationLibrary();  
 initializeIAP();  
 }

# 

# 4.Initializing & Launching a common component

To initialize a coco for eg Product registration :

**private void** initializeProductRegistrationLibrary() {  
 PRDependencies prodRegDependencies = **new** PRDependencies(*gAppInfra*);  
  
 UappSettings uappSettings = **new** UappSettings(getApplicationContext());  
 **new** PRInterface().init(prodRegDependencies, uappSettings);  
}

For Launching a Product registration :

FragmentLauncher fragLauncher = **new** FragmentLauncher(**fa**, **containerID**,**actionBarListener**);  
fragLauncher.setCustomAnimation(0, 0);  
prodRegLaunchInput = **new** PRLaunchInput(products, **false**);  
prodRegLaunchInput.setProdRegUiListener(**this**);  
**new** PRInterface().launch(fragLauncher,prodRegLaunchInput);

1. **Handling ActionBar from Coco**

State must implement ActionBarListener to handle the title and back key events from coco

We have following methods to be overridden for getting title

The string in the below method parameter is the title of actionbar that needs to be set by the app

Boolean b : is the value whether back is handled by coco or we need to handle it.

True : Required to show back button and the respective coco will handle back key

False: App needs to handle the back key

**public void** updateActionBar(@StringRes **int** i, **boolean** b) {  
 setTitle(getResources().getString(i));  
 updateActionBarIcon(b);  
}  
  
*/\*\*  
 \* For Updating the actionbar title as coming from other components  
 \** ***@param s*** *String to be updated on actionbar title  
 \** ***@param b*** *Whether back is handled by them or not  
 \*/*@Override  
**public void** updateActionBar(String s, **boolean** b) {  
 setTitle(s);  
 updateActionBarIcon(b);  
  
}

# Handling Back key from coco

We need to check the value of handleBackEvent() method from BackEventListener from coco .

If true is returned - the respective coco will handle the event ,

If false is7returned – the application needs to handle the back event

**if** (currentFrag != **null** && currentFrag **instanceof** BackEventListener && currentFrag i**nstanceof** RegistrationFragment) {  
 backState = ((BackEventListener) currentFrag).handleBackEvent();  
 **if** (!backState) {  
 fragmentManager.popBackStack(); // Do your stuff here   
 }

# Manifest Detail

Kindly find manifest details in source folder to find the various Android Components added and used.

# 7. Notes

1. Please refer interface Spec Doc or Java documents for more details on APIs for each individual component.
2. Please refer demo app for implementation details of various CoCo