|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Document History** | | | | |
| **Version** | **Date** | **Author** | **Section** | **Changes** |
| 1.0 | 04/08/2016 | Deepthi Shivakumar | All | Release 1 features |

|  |  |
| --- | --- |
| Author | Deepthi Shivakumar, Richa Bajpai |
| Approved by |  |

Contents

[1. INTRODUCTION 1](#_Toc458088766)

[2. INTEGRATION 1](#_Toc458088767)

[2.1 Maven repository (Artifactory based) Integration 2](#_Toc458088768)

[2.2 Library Integration 2](#_Toc458088769)

[2.3 Library versioning 2](#_Toc458088770)

[2.4 .Root gradle changes 2](#_Toc458088771)

[2.5 Proxy dependencies 3](#_Toc458088772)

[3. INITIALIZATION 3](#_Toc458088773)

[4. Android Manifest Changes 3](#_Toc458088774)

[4.1 Other User Permissions 3](#_Toc458088775)

[5. Base Classes 3](#_Toc458088776)

[5.1 BaseClass for Configuration : 3](#_Toc458088777)

[5.2 Classes for Launching Mechanism 4](#_Toc458088778)

[6. Interfaces 4](#_Toc458088779)

# INTRODUCTION

This document provides an overview of integration procedure for uAppframework library in android mobile applications.

# INTEGRATION

There are two ways to integrate “Mobile App Infrastructure” library with any Android app.

* + 1. **Maven repository based**: At compile time, machine has to be connected with Philips network. Do not follow section 2.2
    2. **Library Integration**: If unable to connect with Philips network then include libraries to your root application. Do not follow section 2.4, 2.5

## Maven repository (Artifactory based) Integration

The easiest and preferred way to use these components is using maven.

All dependent libraries should be downloaded from artifactory.

**Artifactory Path:**

If you are inside Philips network then you can directly refer “**2.5 Gradle dependencies**” section. It will automatically download all nested dependencies from artifactory.

## Library Integration

Need to copy all aar files in libs folder; below are the libraries needed, Please make gradle changes

dependencies {  
compile fileTree(**dir**: **'libs'**, **include**: [**'\*.jar'**])compile **'com.android.support:appcompat-v7:23.4.0'**compile(**group**: **'com.philips.cdp'**, **name**: **'AppInfra'**, **version**: **'1.1.0'**, **ext**: **'aar'**)

}

## Library versioning

Library version can be obtained by using below API

version = objcdp.getVersion()

## .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'}**

**}**

**}**

## Proxy dependencies

Gradle dependencies can get some network/proxy related issues. 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 this proxy settings locally. But Eindhoven, does not use above proxy settings.

# INITIALIZATION

There is not applicable for uAppFramework. It contains only base classes and interfaces .

# Android Manifest Changes

No special Permission required:

## Other User Permissions

No special Permission required:

# Base Classes

## BaseClass for Configuration :

Class: LaunchType

This class needs to be extended by micro app frameworks for common configuration.

## Classes for Launching Mechanism

An app can launch micro app either as activity or fragment. Each micro app need to facilitate both kind of launching mechanism. Micro app (component) framework enables this by providing set of base classes. App or any component which launches a micro app is expected to create an object of either Fragment launcher or Activity launcher and inject to micro app launch API

1. Abstract Class: UiLauncher

Contains methods for setting animations for the activity

**public void** setAnimation(**int** enterAnimResId, **int** exitAnimation)

* 1. Class: ActivityLauncher

Needs to be instantiated for launching micro app as activity.

**public** ActivityLauncher(ActivityLauncher.ActivityOrientation screenOrientation,Bundle bunde)

ActivityOrientation is a defined set of int defs which can be used to specify ScreenOrientation.

* 1. Class: FragmentLauncher

Needs to be instantiated for launching micro app as Fragment.

**public** FragmentLauncher(FragmentActivity fragmentActivity,  
 **int** parentContainerResId,  
 ActionBarListener actionbarUpdateListener)

# Interfaces

1.ActionBarListener

For Updating title on Action Bar this interface needs to be implemented providing definition of below method:

**void** updateActionBar(String titleText,**boolean** enableBackKey);

2. BackEventListener

Each micro app is expected to implement this interface. Micro app needs to return true if they need to consume and handle back key event else return false if they do not want to any actions there by app calls super.onBackPressed(). The below method needs to be defined.

**boolean** handleBackEvent();

3. uAppInterface

uAppframework provides below interface which has standard APIs to initialise, launch and set configuration.

Each micro app will expose one single class for vertical to launch UI where in it needs to implement this interface.

Below methods needs to be implemented for this interface

**public void** init(Context context, AppInfra appInfra);

**public void** launch(UiLauncher uiLauncher, uAppListener uAppListener);

**public void** setLaunchInput(LaunchInput launchInput);