|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Document History** | | | | |
| **Version** | **Date** | **Author** | **Section** | **Changes** |
| 0.1 | 6-12-2017 | Salvatore La Fiura | All | V1 features |

EWS microApp Android Integration

1. INTRODUCTION 2

2. LIBRARY INTEGRATION 2

2.1 Library locations 2

2.2 Gradle integration 2

3. Library usage 2

4. Application dependency overview 2

4.1 How to create AppInfra dependency? 2

4.2 How to create the CommCentral dependency? 3

4.3 How to create the ProductKeyMap dependency? 3

4.4 How to create the ContentConfiguration dependency? 3

4.5 Creating UappDependencies 3

5. How to launch the EWS 4

# INTRODUCTION

This document provides an overview of Easy WIFI Setup (EWS) micro app integration procedure for all android applications. EWS is a common component which allows mobile applications to setup their WIFI devices. This common component accept content configuration (optional ) object aiming on customizing the content of the component and a theme configuration object (optional) to specify a theme for the common component.

# LIBRARY INTEGRATION

## Library locations

The source code for the library can be found here:

http://tfsemea1.ta.philips.com:8080/tfs/TPC\_Region24/CDP2/\_git/ews-android-easywifisetupuapp

**This includes the EWS demoApp that can be used as starting point for implementation.**

## Gradle integration

For successful integration you will need to:

* Add the correct maven repository to your root build.gradle file. The url can be found in the uApp root build.gradle file.
* Add the ews dependency to your module build.gradle (usually the ‘app’ module, but you can choose to put it in other modules of course).

# Library usage

For detailed info on how to initialize and use the library, please consult the uApp that can be found in the git archive. Start by taking a look at the *EWSDependencies* class. This class creates everything to initialize EWS.

# Application dependency overview

EWS is having dependency on:

* AppInfra
* CommCentral (DICOMM)
* ProductKeyMap
* Content Configuration

## How to create AppInfra dependency?

**appInfra** = **new** AppInfra.Builder().build(getActivity());

## How to create the CommCentral dependency?

You can have only one instance of CommCentral for the whole project.   
Be sure that you are not recreating such instance but that you keep always the same reference.

@NonNull  
**private** CommCentral createCommCentral(Context context, AppInfraInterface appInfraInterface) {  
LanTransportContext lanTransportContext = **new** LanTransportContext(  
 **new** RuntimeConfiguration(context, appInfraInterface));  
 ApplianceFactory factory = **new** ApplianceFactory (lanTransportContext);  
 *commCentral* = **new** CommCentral(factory, lanTransportContext);  
 **return** *commCentral*;  
}

## How to create the ProductKeyMap dependency?

**private** Map<String, String> createProductMap() {  
 Map<String, String> productKeyMap = **new** HashMap<>();  
 productKeyMap.put(EWSInterface.***PRODUCT\_NAME***, getString(R.string. ***device\_name***));  
 **return** productKeyMap;  
}

## How to create the ContentConfiguration dependency?

ContentConfiguration provides all the needed strings that needs to be provided in order to customize all the fragments of our component.  
This Object is composed of:

* BaseContentConfiguration
* HappyFlowConfiguration
* TroubleShootingConfiguration

All these objects are having a builder in order to have an easier understanding.

## Creating UappDependencies

The UappDependencies object contains all the dependencies needed by the microApp to be launched and can be created as follow:

*/\*\*  
 \* create uApp dependency from proposition for EWS microapp.  
 \* commCentral should be created and passed from proposition.  
 \** ***@param appInfra*** *\** ***@param productKeyMap*** *\** ***@return*** *\*/*@NonNull  
**private** UappDependencies createUappDependencies(**final** AppInfraInterface appInfra,  
 Map<String, String> productKeyMap) {  
 **return new** EWSDependencies(appInfra, productKeyMap,  
 **new** ContentConfiguration(createBaseContentConfiguration(),  
 createHappyFlowConfiguration(),  
 createTroubleShootingConfiguration())) {  
 @Override  
 **public** CommCentral getCommCentral() {  
 **return** createCommCentral(getActivity(), appInfra);  
 }  
 };  
}

# How to launch the EWS

The EWS can be launched in two ways:

* ActivityLauncher (Activity is hosted by the )
* FragmentLauncher (Activity is hosted by the proposition)

**private void** launchEwsUApp() {  
 EWSInterface ewsInterface = **new** EWSInterface();  
 ewsInterface.init(createUappDependencies(**appInfra**, productMap), **new** EWSLauncherInput());  
 *//its up to proposition to pass themeConfig or not ,if not passing theme then it will show default theme of library* ewsInterface.launch(**new** ActivityLauncher(***SCREEN\_ORIENTATION\_PORTRAIT***, (***themeConfig****,* -1, **null**), **new** EWSLauncherInput());

}  
  
**private void** launchEWSFragmentUApp() {  
 EWSInterface ewsInterface = **new** EWSInterface();  
 ewsInterface.init(createUappDependencies(**appInfra**, productMap), **new** UappSettings(getActivity()));  
 FragmentLauncher fragmentLauncher = **new** FragmentLauncher  
 (getActivity(), R.id.***mainContainer***, ((ActionBarListener) getActivity()));  
 ewsInterface.launch(fragmentLauncher, **new** EWSLauncherInput());

}