**Product Registration Android Integration**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Document History** | | | | |
| **Version** | **Date** | **Author** | **Section** | **Changes** |
| 0.1 | 02-05-2016 | Yogesh HS | All | Initial draft |
| 0.2 | 16-05-2016 | Yogesh HS | All | Incorporated review comments |

|  |  |
| --- | --- |
| **Author** | Yogesh HS |
| **Approved By** | Shivakumar Deepthi |
| **Email Id** | yogesh.hs@philips.com |

Contents

[1. Introduction 3](#_Toc451266879)

[2. Integration 3](#_Toc451266880)

[2.1 Artifactory 3](#_Toc451266881)

[2.2 Root gradle changes 3](#_Toc451266882)

[2.3 Dependencies 4](#_Toc451266883)

[2.3.1 Gradle dependencies 4](#_Toc451266884)

[2.3.2 Library dependencies 4](#_Toc451266885)

[2.4 Proxy dependencies 4](#_Toc451266886)

[2.5 Prerequisites 5](#_Toc451266887)

[3. Steps for API integration 5](#_Toc451266888)

[4. Notes 7](#_Toc451266889)

# Introduction

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

Source Path: [https://atlas.natlab.research.philips.com:7999/pr/hor-productregistration-android.git](ssh://git@atlas.natlab.research.philips.com:7999/pr/hor-productregistration-android.git)

# 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/product-registration-lib/1.0.0/**](http://maartens-mini.ddns.htc.nl.philips.com:8081/artifactory/simple/libs-release-local-android/com/philips/cdp/product-registration-lib/1.0.0/)

compile(group: 'com.philips.cdp', name: 'product-registration-lib', version: ‘1.0.0', ext: 'aar')

{  
 transitive = **true** }

# 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:1.5.0'** classpath **'org.jfrog.buildinfo:build-info-extractor-gradle:3.1.2'** classpath **'com.github.dcendents:android-maven-gradle-plugin:1.3'** }  
  
}  
}  
  
allprojects {  
 repositories {  
 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'** }  
 jcenter()  
 }  
}

# 2.3 Dependencies

# 2.3.1 Gradle dependencies

Just by adding below gradle dependencies, Production Registration and nested possible libraries will be downloaded from artifactory.

compile(group: 'com.philips.cdp', name: 'product-registration-lib', version: '1.0.0-rc.1', ext: 'pom'){

transitive=**true**

}

# 2.3.2 Library dependencies

1: User Registration : 6.0.0

2: PRX Client : 2.0.0

3: Local Match : 2.0.0

# 2.4 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 these proxy settings locally. But Eindhoven, does not use above proxy

settings.

## Prerequisites

1. Application need to know **CTN** number for each product used in app.

Ex: HD8967/01.

ii) Sector.

iii) Catalog.

iv) Serial number. (Optional depends on metadata)

v) Purchase date. (Optional depends on metadata)

Example: **Sector**: B2C, **catalog**: CONSUMER/CARE, Serial number: 1344 and Purchase date: yyyy-mm-dd.

**Note:** Above information is used for backend services and mostly uses PRX system.

Hence please provide valid and complete CTN number as input.

# Steps for API integration

**Following to be invoked in Application class**

1. Initialize Product registration module

ProdRegHelper prodRegHelper = new ProdRegHelper();  
prodRegHelper.init(this);// Pass Application context as parameter

1. Product registration have dependency on User Registration. So for integrating product registration in your app you have to add following User registration code in Application class.

RegistrationConfiguration.getInstance().setPrioritisedFunction(RegistrationFunction.Registration);

String languageCode = Locale.getDefault().getLanguage();

String countryCode = Locale.getDefault().getCountry();

PILLocaleManager localeManager = new PILLocaleManager(this);

localeManager.setInputLocale(languageCode, countryCode);

RegistrationHelper.getInstance().initializeUserRegistration(this);

1. Product Registration can be launched in two ways
   1. **Launch as a fragment in current activity:** To register product list using this mode refer following steps.
      1. **Create fragment launcher instance:**

**//**Parameters:

**//**1**.** fragmentActivity:your activity reference

//2.R.id.parent\_layout: container id where you want fragment to be launched.

FragmentLauncher fragLauncher = **new** FragmentLauncher(**fragmentActivity**, R.id.***parent\_layout***, **new** ActionbarUpdateListener() {  
 @Override  
 **public void** updateActionbar(**final** String var1) {  
   
 }  
});

fragLauncher.setAnimation(0, 0);

* + 1. **Setup configuration:**

//Parameters:

//1. Product List to register

//2. Set true if flow is app flow else set false

ProdRegConfig prodRegConfig = **new** ProdRegConfig(products, **true**);

* + 1. **Invoke Product Registration:**

ProdRegUiHelper.*getInstance*().invokeProductRegistration(fragLauncher, prodRegConfig, **new** ProdRegUiListener() {  
 @Override  
 **public void** onProdRegContinue(**final** List<RegisteredProduct> registeredProducts, **final** UserWithProducts userWithProduct) {  
 //Prod registration is successful and user has pressed continue button.  
 }  
  
 @Override  
 **public void** onProdRegBack(**final** List<RegisteredProduct> registeredProducts, **final** UserWithProducts userWithProduct) {  
 // Pro reg is unsuccessful and user has pressed back key to exit prod reg.  
 }  
});

* 1. **Launch as a new activity:** To register product list using this mode refer steps
     1. **Create activity launcher instance:**

**//**Parameters:

**//**1**.** fragmentActivity:your activity reference

//2. Orientation for activity

//3. UiKit Theme

ActivityLauncher activityLauncher = **new** ActivityLauncher(getActivity(), ActivityLauncher.ActivityOrientation.***SCREEN\_ORIENTATION\_UNSPECIFIED***, 0);

* + 1. **Setup configuration:**

//Parameters:

//1. Product List to register

//2. Set true if flow is app flow else set false

ProdRegConfig prodRegConfig = **new** ProdRegConfig(products, **true**);

* + 1. **Invoke Product Registration:**

ProdRegUiHelper.*getInstance*().invokeProductRegistration(fragLauncher, prodRegConfig, **new** ProdRegUiListener() {  
 @Override  
 **public void** onProdRegContinue(**final** List<RegisteredProduct> registeredProducts, **final** UserWithProducts userWithProduct) {  
 //Prod registration is successful and user has pressed continue button.  
 }  
  
 @Override  
 **public void** onProdRegBack(**final** List<RegisteredProduct> registeredProducts, **final** UserWithProducts userWithProduct) {  
 // Pro reg is unsuccessful and user has pressed back key to exit prod reg.  
 }  
});

1. To register list of products you need to form Product object with CTN, Sector and Catalog details. Refer below code snippet.

// pass CTN, Sector, Catalog

Product product = new Product ("HC540/83", Sector.B2C, Catalog.CONSUMER);

// set serial number

product.setSerialNumber(mSerialNumber);

// set purchase date in ("YYYY-MM-DD") format

product.setPurchaseDate(mPurchaseDate);

//set email configuration as true or false, if true this will email the product that was registered

product.sendEmail(String.valueOf(true));

final ProdRegListener listener = new ProdRegListener() {

@Override

public void onProdRegSuccess(RegisteredProduct registeredProduct, UserWithProducts userWithProducts) {

//on sucess additional implementation

}

@Override

public void onProdRegFailed(RegisteredProduct registeredProduct, UserWithProducts userWithProducts) {

//on failed additional implementation

}

};

// adding call back listener while registering product will trigger callbacks Success and Failed

prodRegHelper.addProductRegistrationListener (listener);

//API to register product for current signed-in user

prodRegHelper.getSignedInUserWithProducts ().registerProduct (product);

**Note:**

Kindly process the call back object registeredProduct to get following information

1. To get CTN and Serial number of product registered
2. To get state of Registration (Registered, Pending, Failed)
3. To get error state when product registration failed. (Ex: Invalid Serial Number, Product Already Registered etc.,)

**Following to invoked to get Registered Products list**

ProdRegHelper prodRegHelper = new ProdRegHelper();

// Listener to be initialized to get call backs

final RegisteredProductsListener registeredProductsListener = new RegisteredProductsListener() {  
 @Override  
 public void getRegisteredProductsSuccess(final List<RegisteredProduct> registeredProducts, final long timeStamp) {}

prodRegHelper.getSignedInUserWithProducts().getRegisteredProducts(registeredProductsListener, Sector.B2C, Catalog.CONSUMER);

# Notes

1. Please refer interface Spec Doc or Java documents for more details on APIs.
2. Please refer demo app for implementation details.