**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 |
| 0.3 | 27-07-2016 | Yogesh HS | All | Introduced UI |
| 0.4 | 22-09-2016 | Yogesh HS | All | Incorporated one roof changes of App-framework |
| 0.5 | 08-03-2017 | Shah Faizal | All | App infra integration service discovery.  PRX client changes defined by App-Infra |
| 0.6 | 15-05-2017 | Shah Faizal | All | Defect fix |
| 0.7 | 07-07-2017 | Shah Faizal | All | Defect fix |
| 0.8 | 12-10-2017 | Shah Faizal | All | DLS Integration |

|  |  |
| --- | --- |
| **Author** | Shah Faizal |
| **Approved By** | Viswarad Reddy |
| **Email Id** | Viswarad.Reddy@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: <http://tfsemea1.ta.philips.com:8080/tfs/TPC_Region24/CDP2/_git/prg-android>

# Integration

Integration can be done in following ways.

# 2.1 Artifactory

All dependent libraries should be downloaded from artifactory.

**Artifactory path:**

[**http://artifactory-ehv.ta.philips.com:8082/artifactory/platform-pkgs-android-release-local/com/philips/cdp/product-registration-lib/3.0.0**](http://artifactory-ehv.ta.philips.com:8082/artifactory/platform-pkgs-android-release-local/com/philips/cdp/product-registration-lib/3.0.0)

compile(group: 'com.philips.cdp', name: 'product-registration-lib', version: ‘3.0.0, ext: 'aar')

{  
 transitive = **true** }

# 2.2 Root gradle changes

buildscript {  
 repositories {  
 maven { url **'http://artifactory-ehv.ta.philips.com:8082/artifactory/platform-jcenter-remote'** credentials {  
 username = **"readonly"** password = **"123qweasdzxc"** }  
 }  
 maven {  
 url **"http://artifactory-ehv.ta.philips.com:8082/artifactory/platform-plugins-release-local"** credentials {  
 username = **"readonly"** password = **"123qweasdzxc"** }  
 }  
 }  
 dependencies {  
 classpath **'com.android.tools.build:gradle:2.3.1'** classpath **'com.neenbedankt.gradle.plugins:android-apt:1.4'** classpath **'org.jfrog.buildinfo:build-info-extractor-gradle:4.4.12'** classpath **group**: **'com.philips.cdp'**, **name**: **'cdpSCM'**, **version**: **'2.1.2'**, **changing**: **true** classpath **'com.github.jruby-gradle:jruby-gradle-plugin:1.1.5'** }  
}  
  
allprojects {  
 repositories {  
 maven {  
 url **"http://artifactory-ehv.ta.philips.com:8082/artifactory/platform-plugins-release-local"** credentials {  
 username = **"readonly"** password = **"123qweasdzxc"** }  
 }  
 maven {  
 url objcdp.getArtifactoryUrl()  
 credentials {  
 username = **"readonly"** password = **"123qweasdzxc"** }  
 }  
   
 apply **plugin**: **'com.jfrog.artifactory'** group = **'com.philips.cdp'** flatDir {  
 dirs **'libs'** }   
 }

**task clean(type: Delete) {  
 delete rootProject.buildDir  
}**

# 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: ‘3.0.0, ext: 'aar')

{  
 transitive = **true** }

# 2.3.2 Library dependencies

1: User Registration : 11.0.0

2: UID : 0.1.1

3: prx : 3.4.0

# 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

PRDependencies prodRegDependencies = **new** PRDependencies (*mAppInfra*);// **Pass App-infra instance as parameter**PRSettings prodRegSettings = **new** PRSettings (getApplicationContext());// **Pass Application context as parameter**  
**new** PRInterface().init(prodRegDependencies, prodRegSettings);// **Pass dependencies and settings required for initialization**

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** ActionBarListener () {  
 @Override  
 **public void** updateActionBar (@StringRes final int resId, final boolean enableBack) {  
   
 }

@Override

public void updateActionBar(final String actionBarTitle, final boolean enableBack) {}  
});

// enableBack will return false for first screen/fragment and true for other screen/fragment to support enabling hamburger icon.

fragLauncher.setCustomAnimation (0, 0);

* + 1. **Setup configuration:**

//Parameters:

//1. Product List to register

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

PRLaunchInput prodRegLaunchInput = new PRLaunchInput (products, true);

// Set the call back listener or else library throws Runtime Exception

prodRegLaunchInput.setProdRegUiListener( new ProdRegUiListener() {

@Override

public void onProdRegContinue(final List<RegisteredProduct> registeredProduct, final UserWithProducts userWithProduct) {

}

@Override

public void onProdRegBack(final List<RegisteredProduct> registeredProduct, final UserWithProducts userWithProduct) {

}

@Override

public void onProdRegFailed(final ProdRegError prodRegError) {

}

});

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

PRInterface prInterface = new PRInterface();

prInterface.launch(fragLauncher, prodRegLaunchInput);// pass launcher type and launch Input as parameters.

* 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. UID DLS Theme

ActivityLauncher activityLauncher = new ActivityLauncher(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

PRLaunchInput prodRegLaunchInput = new PRLaunchInput (products, true);

// Set the call back listener or else library throws Runtime Exception

prodRegLaunchInput.setProdRegUiListener( new ProdRegUiListener() {

@Override

public void onProdRegContinue(final List<RegisteredProduct> registeredProduct, final UserWithProducts userWithProduct) {

}

@Override

public void onProdRegBack(final List<RegisteredProduct> registeredProduct, final UserWithProducts userWithProduct) {

}

@Override

public void onProdRegFailed(final ProdRegError prodRegError) {

}

});

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

PRInterface prInterface = new PRInterface();

prInterface.launch launch(activityLauncher, prodRegLaunchInput);

1. For handling back event kindly refer the below code

@Override  
**public void** onBackPressed() {  
 FragmentManager fragmentManager = getSupportFragmentManager();

boolean backState = false;

Fragment currentFrag = fragmentManager

.findFragmentById(R.id.parent\_layout);

if (currentFrag != null && currentFrag instanceof BackEventListener) {

backState = ((BackEventListener) currentFrag).handleBackEvent();

}

if (!backState) {

super.onBackPressed();

}

}

1. To register products without UI you need to form Product object with CTN, Sector and Catalog details. Refer below code snippet.

// pass CTN, Sector, Catalog

Product product = new Product ("HC5410/83", PrxConstants.Sector.B2C, PrxConstants .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);

**Pro-guard Scripts**

*#Product Registration library***-keep** class com.philips.cdp.prodreg.model.\*\* {**\*;**}  
**-keep** class com.philips.cdp.prodreg.register.\*\* {**\*;**}  
**-keep** class com.philips.cdp.prodreg.localcache.\*\* {**\*;**}

# Notes

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