|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Document History** | | | | |
| **Version** | **Date** | **Author** | **Section** | **Changes** |
| 0.1 | 08-07-2015 | Team Huma | All | Initial draft |
| 0.2 | 28-07-2015 | Team Huma | All | Initial draft |
| 0.3 | 13-08-2015 | Team Huma | All | Initial draft |
| 0.4 | 21-10-2015 | Team Huma | All | HSDP incorporation and configuration changes |
| 0.5 | 04-12-2015 | Team Huma | All | Dependant libs update ,Flow configuration,Janrain Initialization,Tagging,Progaurd,Callbacks |
| 0.6 | 04-02-2016 | Team Huma | All | Dynamic Configuration |
| 0.7 | 04-03-2016 | Team Huma | All | Artifactory Support and More UI Configuration with Color |
| 0.8 | 06-04-2016 | Team Huma | All | COPPA changes |
| 0.9 | 19-04-2016 | Team Huma | All | Drawable support ,Complete COPPA module ,Manifest permissions |
| 1.0 | 09-05-2016 | Team Huma | All | LocaleMatch 2.0.0,Tagging 3.0.0,Additional color attributes,COPPA localization,Source path  change |
| 1.1 | 22-07-2016 | Team Huma | All | LocaleMatch 2.1.0,AppInfra 1.1.0,Localization. |
| 1.2 | 29-09-2016 | Team Huma | All | Bug fixes ,API normalization as per mApp,AppInfra implementation for Configuration and removal of the registration configuration from assets and dynamic configuration. |
| 1.3 | 02-12-2016 | Team Huma | 11 | AB testing , China mobile based registration .Removed stand alone server time component .Localization |
| 1.4 | 07-03-2017 | Team Huma | 6 | WeChat and Mobile number reset in App |
| 1.5 | 24-03-2017 | Team Huma | 7 | Google plus web authentication |
| 1.6 | 21-04-2017 | Team Huma | 4.2,9.3 | New opt in strings and flow customization at module. Google merge fix |
| 1.7 | 04-05-2017 | Team Huma | 1,3.1,3.2 | Updated ne source link,Updated the new jars refered in UR |
| 1.8 | 07-07-2017 | Team Huma | 1,2,3,3.4.5.1,5 | Updated min and target android version ,Country selection ,Appinfra new changes and links updated for UR and AppInfra |
| 1.9 | 22-03-2018 | Team Huma | 12 | Consent definition for GDPR compliance |
| 2.0 | 29-03-2018 | Team Huma | 13 | Implementation of MyDetail Screen |
| 2.1 | 04-05-2018 | Team Huma | 9 | Coppa flow integration |
| 2.2 | 11-05-2018 | Team Huma | All | Optimization of document |

Registration Android Integration

[Introduction 3](#_Toc513807947)

[Pre-Requisites 3](#_Toc513807948)

[Maven Repository 3](#_Toc513807949)

[Library Integration 4](#_Toc513807950)

[Configuration 4](#_Toc513807951)

[4.1 UI Configurations [registration\_config.xml] 4](#_Toc513807952)

[4.2 Customizing Opt in screen content 5](#_Toc513807953)

[4.3 UI Priority function 5](#_Toc513807954)

[4.3.1 Create account on Top 6](#_Toc513807955)

[Sign In On top 6](#_Toc513807956)

[Flow Configurations 6](#_Toc513807957)

[Standard 6](#_Toc513807958)

[HSDP 6](#_Toc513807959)

[Registration Configuration 6](#_Toc513807960)

[Static Configuration 6](#_Toc513807961)

[Dynamic Configuration 8](#_Toc513807962)

[SigninProviders 10](#_Toc513807963)

[Hybrid Configuration 10](#_Toc513807964)

[AppInfra Configuration 10](#_Toc513807965)

[WeChat integration 11](#_Toc513807966)

[4.1 Prerequisites 11](#_Toc513807967)

[Application Side 11](#_Toc513807968)

[Google+ integration 12](#_Toc513807969)

[4.1 Prerequisites 12](#_Toc513807970)

[Application Side 13](#_Toc513807971)

[Providing Client ids in resource files 13](#_Toc513807972)

[Register Google Auth activities in manifest files 14](#_Toc513807973)

[Provide the manifest place holders in build.gradle: 14](#_Toc513807974)

[Make sure 14](#_Toc513807975)

[Coppa Flow Integration 15](#_Toc513807976)

[Launching Registration component 15](#_Toc513807977)

[Launch with different UI flows(ABTest) 15](#_Toc513807978)

[Call-back’s 16](#_Toc513807979)

[GDPR Consents 17](#_Toc513807980)

[MyDetails Screen 17](#_Toc513807981)

[Refresh User Session 17](#_Toc513807982)

[HSDP specific API’s 17](#_Toc513807983)

[4.2 HSDP UUID 17](#_Toc513807984)

[HSDP Access Token 18](#_Toc513807985)

[Library version’s 18](#_Toc513807986)

[4.3 Janrain Library 18](#_Toc513807987)

[Locale Match Library 18](#_Toc513807988)

[Registration Library 18](#_Toc513807989)

[Deprecated APIs 18](#_Toc513807990)

# Introduction

This document provides an overview of integration Registration Android UI feature in Android Mobile applications.

Source Path:

<http://tfsemea1.ta.philips.com:8080/tfs/TPC_Region24/CDP2/TEAM%20UR%20Registered%20Android/_git/usr-android-user-registration>

# Pre-Requisites

* Android Studio 2.0 or higher
* A device running Android version 4.4 or newer

# Maven Repository

allprojects {  
 repositories {  
 maven { url 'http:// artifactory-ehv.ta.philips.com:8082/artifactory/jcenter' }  
 maven {  
 url 'http:// artifactory-ehv.ta.philips.com:8082/artifactory/ext-release-local'  
 }  
 maven {  
 url ' artifactory-ehv.ta.philips.com:8082/artifactory/libs-release-local-android'  
 }  
 maven {  
 url ' artifactory-ehv.ta.philips.com:8082/artifactory/libs-stage-local-android'  
 }  
 flatDir {  
 dirs 'libs'  
 }  
 }  
}

# Library Integration

InAppPurchase can be integrated by adding the library to the build.gradle file as compile(**‘com.philips.cdp:registrationApi’**)

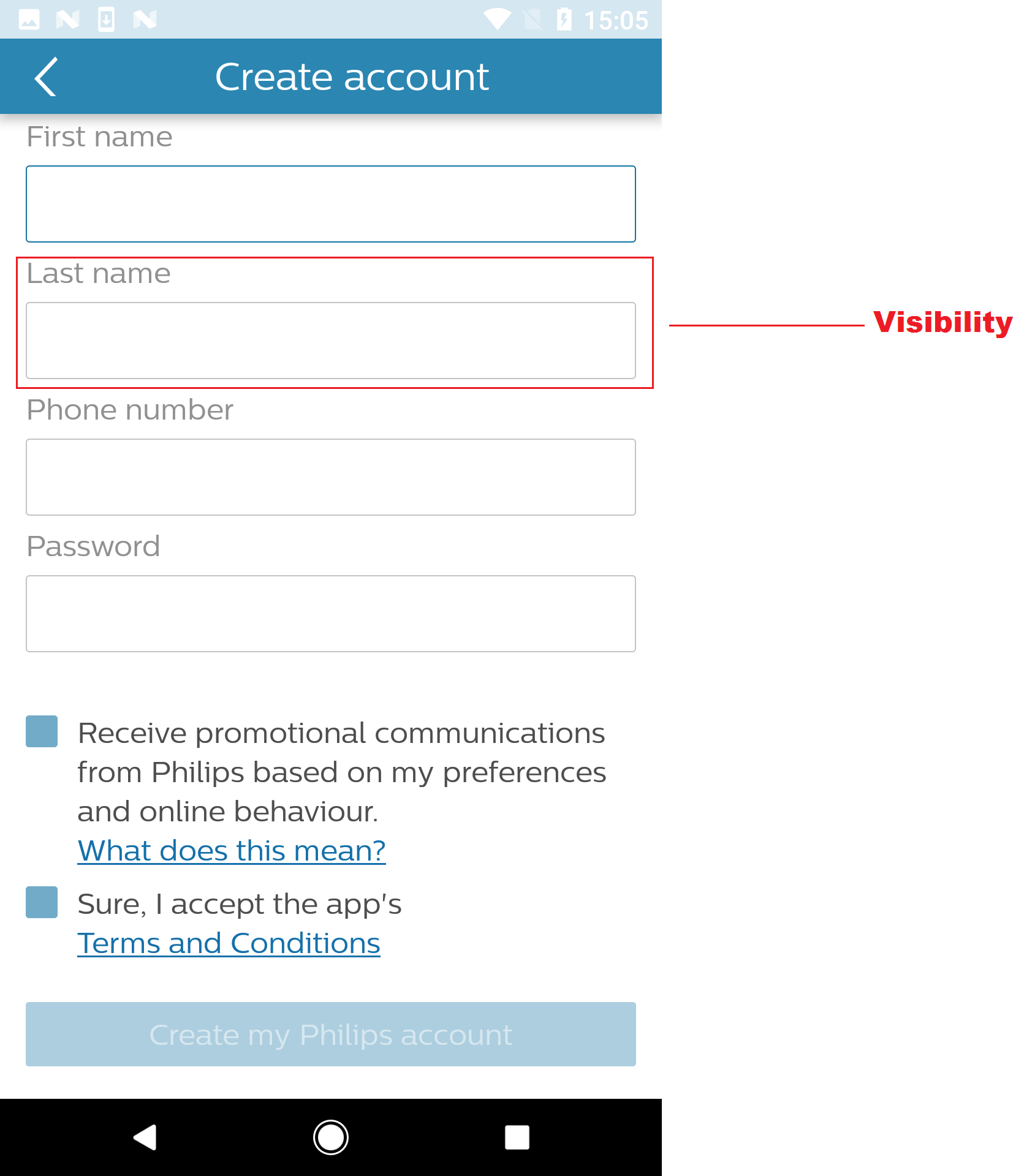
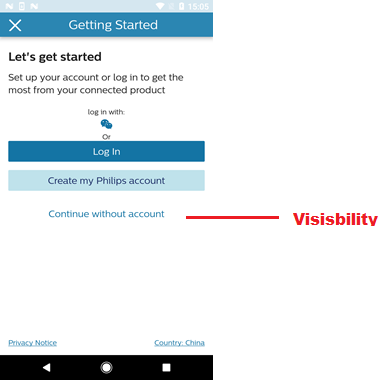
Source code path of the library with the sample demo app: <http://tfsemea1.ta.philips.com:8080/tfs/TPC_Region24/CDP2/_git/plf-android>

# Configuration

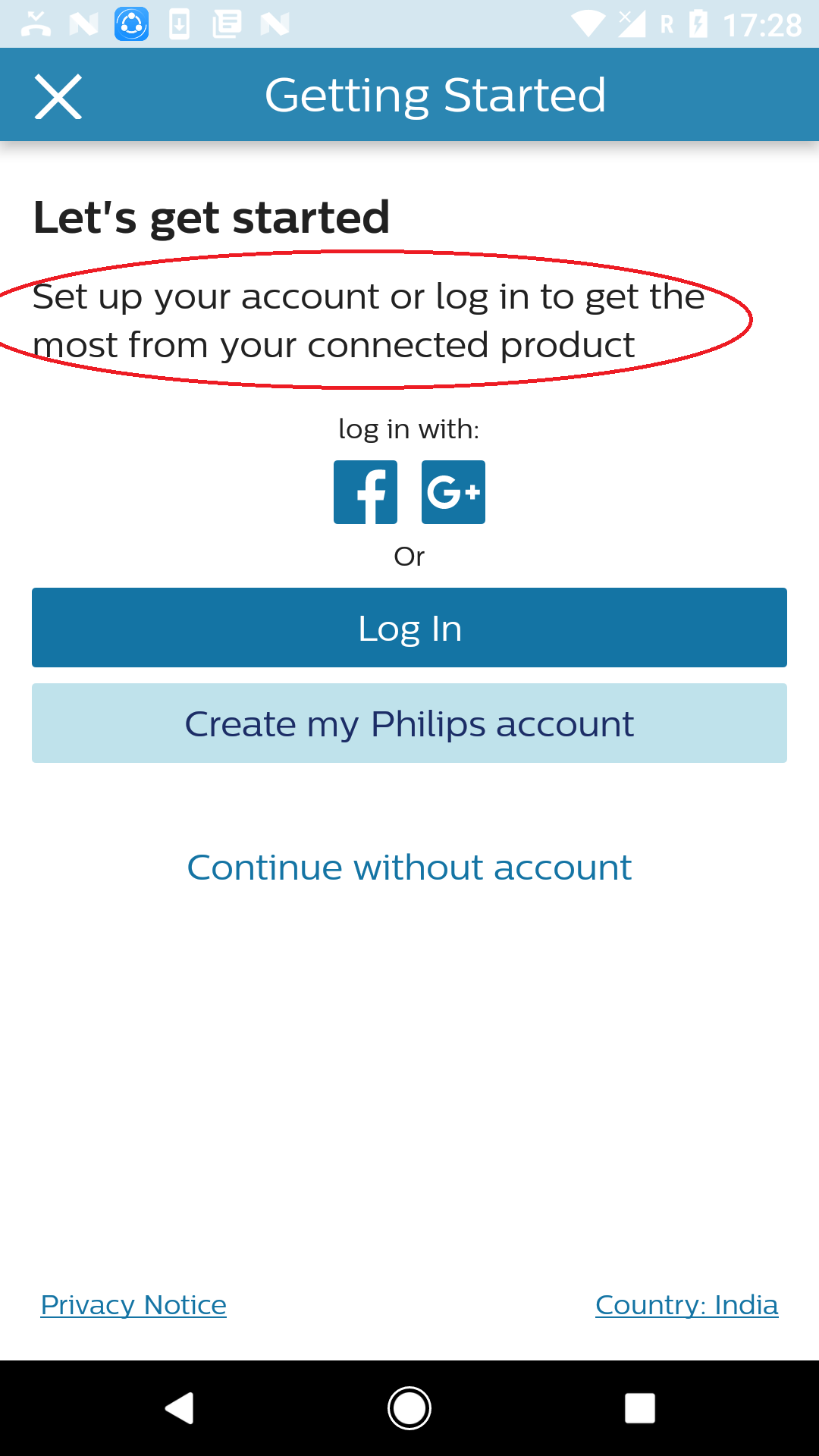
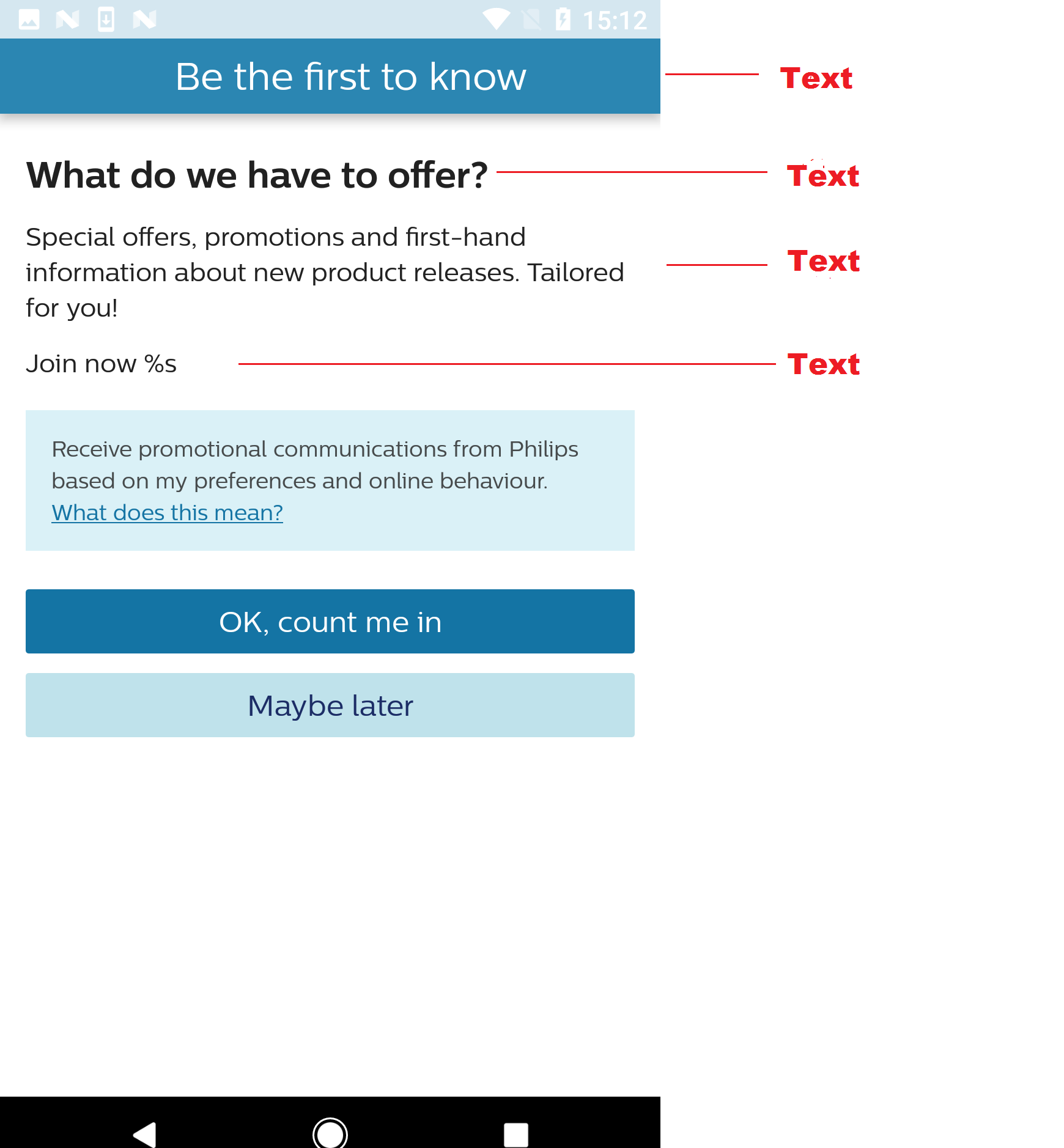
Registration Android can be configurable in two types UI and Flow configurations

## UI Configurations [registration\_config.xml]

**Configurable elements**



UI Theming for altering UI look and feel as per vertical specific follow bellow steps:

****

Copy registration\_config.xml from reference app project and paste into project’s values folder and alter as per vertical need. Color codes need to be defined as follows, if not mentioned default configuration are picked.

#### Vertical specific texts

##### Setting need of account string

Vertical apps may or may not provide why user account is needed.

If need to provide then vertical apps needs to add following string in the respective localized ***string.xml*** file which will be reflected in in following screen

***<string name="*reg\_DLS\_StratView\_Detail\_Txt*">***Set up your account or log in to get the most from your connected product***</string>***

## Customizing Opt in screen content

Opt in screen text can be configured now with the API which will override the default strings. RegistrationContentConfiguration API’s: Please find below the sample code:

**public** RegistrationContentConfiguration getRegistrationContentConfiguration() {  
 String valueForEmailVerification = **"sample"**;  
 String optInTitleText = getResources().getString(R.string.*reg\_DLS\_OptIn\_Navigation\_Bar\_Title*);  
 String optInQuessionaryText = getResources().getString(R.string.*reg\_DLS\_OptIn\_Header\_Label*);  
 String optInDetailDescription = getResources().getString(R.string.*reg\_DLS\_Optin\_Body\_Line1*);  
 String optInBannerText = getResources().getString(R.string.*reg\_Opt\_In\_Join\_Now*);  
 String optInTitleBarText = getResources().getString(R.string.*reg\_DLS\_OptIn\_Navigation\_Bar\_Title*);  
 RegistrationContentConfiguration registrationContentConfiguration = **new** RegistrationContentConfiguration();  
 registrationContentConfiguration.setValueForEmailVerification(valueForEmailVerification);  
 registrationContentConfiguration.setOptInTitleText(optInTitleText);  
 registrationContentConfiguration.setOptInQuessionaryText(optInQuessionaryText);  
 registrationContentConfiguration.setOptInDetailDescription(optInDetailDescription);  
 registrationContentConfiguration.setOptInBannerText(optInBannerText);  
 registrationContentConfiguration.setOptInActionBarText(optInTitleBarText);  
 registrationContentConfiguration.enableLastName(**true**);  
 registrationContentConfiguration.enableContinueWithouAccount(**true**);  
 **return** registrationContentConfiguration;  
}

## UI Priority function

UI priority function enables app to launch Create account at TOP or BOTTOM.Needs to set during URLaunchInput class method during launch of UI . By default Create account is TOP. Possible selection is out of

**public enum RegistrationFunction {  
 Registration,  
 SignIn  
}**

### Create account on Top

**//REGISTRATION UI PRIORITY(CREATE OR SIGNIN**

**URLaunchInput urLaunchInput = new URLaunchInput();  
urLaunchInput.setRegistrationFunction(RegistrationFunction.Registration)**

### Sign In On top

**URLaunchInput urLaunchInput = new URLaunchInput();  
urLaunchInput.setRegistrationFunction(RegistrationFunction.SignIn)**

## Flow Configurations

### Standard

No additional parameter required. Framework will take care of initializing it in Standard flow. Depending on provided AppConfig.json.

### HSDP

HSDP configuration is STANDARD + Dynamic configuration for HSDP part mentioned in configuration key value pairs of UR module.

## Registration Configuration

Registration can be configured in static or dynamic or hybrid way.

### Static Configuration

In Static configuration, Vertical Apps must define one ***AppConfig.json*** file as mentioned below. Once added then cannot be altered by code if any changes need to be done then should go with the new build itself.

The ***AppConfig.json*** should be added to the project and json need to update depend on the project under the user registration module key “UserRegistration”.

* + JanrainConfiguration (Mandatory)
  + PILConfiguration (Optional)
  + Flow (Optional)
  + SigninProviders
  + Country Selection

#### JanRainConfiguration (Mandatory)

#### JanRainConfiguration is required to be able to connect to janrain services. It requires following fields:

**RegistrationClientID:** Assign mandatory client id for each environment.

For e.g. “Development”: “xxxxxxxxxx”

Please contact **Digital Services <digital.services@philips.com>** for Client ID for janrain and Microsite ID. This can be single string value or country specific with country code as key and string as value

***Note :***

***Its highly recommended that app should use STAGING environment mandatorily during development, and STAGING webpages connected to the Evaluation data stores.***

#### PILConfiguration (Optional)

PIL configuration requires following three elements:



#### Flow (Optional)



#### SigninProviders

Add the entire providers array to default, if you need any country specify sign in providers. Add the list of providers with country code, please refer to the example for more details***.[Note My Philips is always shown so need to configure other providers only]***

SignIn Providers supported by registration currently:

facebook

googleplus

wechat

#### Country Selection

User Regietstion now support the country selection via appconfig.json by ShowCountrySelection , supportedHomeCountries, fallbackHomeCountry keys .

**ShowCountrySelection:** True to enable country selction

**supportedHomeCountries :**List of supported country .List will be shown as intersection of default supported list and this attribute listed countries .

**fallbackHomeCountry :** Default fallback country is US if want to change then have mention any country code among the default supported country list mentioned bellow.

## Dynamic Configuration

It is possible to change or define entire Registration configuration in code dynamically by following way. Which are same as static configuration.

Note: Dynamic configuration are taken as priority over the static and only be done by AppInfra key value pairs. It has internal dependency of AppInfra interface.

#### JanRainConfiguration

For replacing or adding Janarin Configuration ids are quite simple now it can be done in following ways. Registration Framework will take care of replacing and adding ids .Use AppInfraInterface object.

******

#### PILConfiguration

For replacing or adding PILConfiguration ids are quite simple now it can be done in following ways . Registration Framework will take care of replacing and adding ids.

******

#### Flow

******

#### HSDP Configuration (Optional)

******

#### **SigninProviders**:

******

### Hybrid Configuration

Hybrid configuration is combination of static and dynamic Vertical app can provide both configuration. Registration framework will prioritize based on being Dynamic configuration higher priorities.

# AppInfra Configuration

User registration is dependent on AppInfra component for logging, tagging, locale-match, Service discovery sub components; Integrating component has to refer AppInfra integration docs explicitly for configuration.

External link :

[https://share-intra.philips.com/sites/STS20140210103002/05%20Horizontal/\_layouts/15/start.aspx#/Shared%20Documents/Forms/AllItems.aspx?RootFolder=%2Fsites%2FSTS20140210103002%2F05%20Horizontal%2FShared%20Documents%2F07%20Mobile%20App%20Infra%2FReleases&FolderCTID=0x012000A2FBB4D36ACF3D4EB72692529B9749F6&View=%7BA5D71447%2DF373%2D434D%2DAF95%2DBA247DAD4B55%7D](https://share-intra.philips.com/sites/STS20140210103002/05%20Horizontal/_layouts/15/start.aspx#/Shared%20Documents/Forms/AllItems.aspx?RootFolder=%2Fsites%2FSTS20140210103002%2F05%20Horizontal%2FShared%20Documents%2F07%20Mobile%20App%20Infra%2FReleases&FolderCTID=0x012000A2FBB4D36ACF3D4EB72692529B9749F6&View=%7BA5D71447%2DF373%2D434D%2DAF95%2DBA247)

Internal Link:

<http://tfsemea1.ta.philips.com:8080/tfs/TPC_Region24/CDP2/TEAM%20App%20Chassis/_git/ail-android-appinfra?path=%2FDocuments%2FExternal%2FAIL000009_Integration%20Document_Appinfra_Android_1.7.0.docx&version=GBdevelop&_a=contents>

For Service discovery we have add following static or dynamic code in AppConfig.json or under key module “appinfra”.

Note: If app state set as production in static configuration then dynamic override of value is ignored and use Production

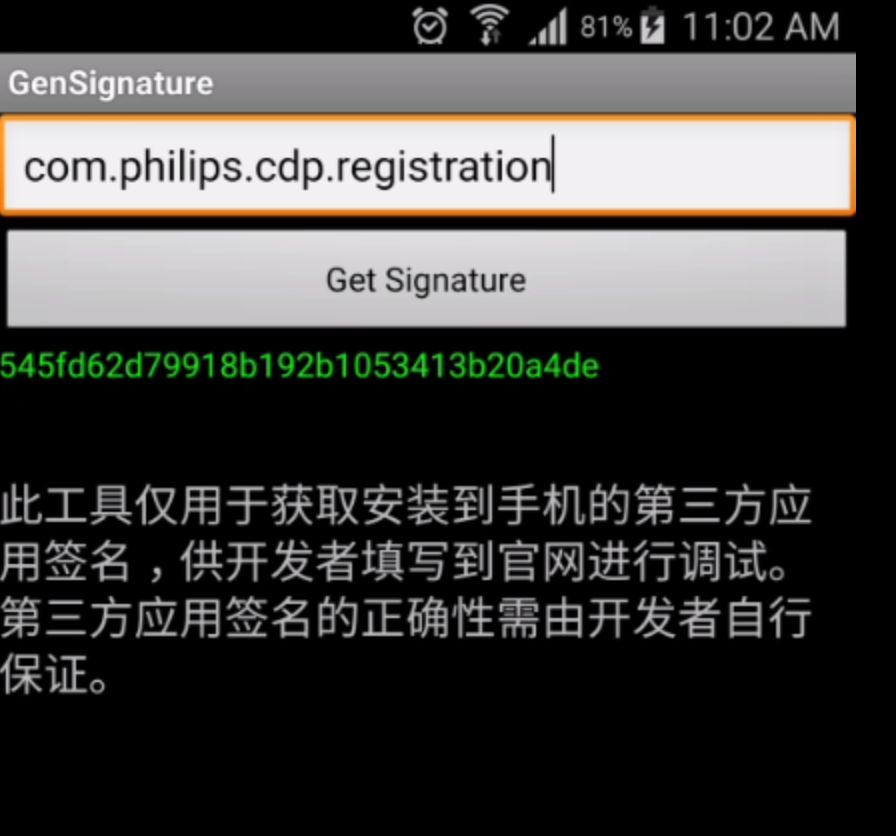
# WeChat integration

We chat social provider is supported in UR module now. To enable WeChat for your application following are steps mandatory to work as desired.

## Prerequisites

* **WeChat App must be installed in Device**
* **Configure Wechat ID and Wechat App Secrete**

To get WeChat ID and WeChat Secrete from the Wechat developer site their is need of AppSignature and App Package name



**AppSignature:** This can be generated by following below steps

* Install Gen\_Signature\_Android221cbf.apk tool [Can be found along with external docs]
* Install your signed APK [One which WeChat has to be configured]
* Launch the Gen\_Signature\_Android221cbf tool enter package name and click on generate button .
* App Signature is generated [Which is used in WeChat developer console]
* Use this signature to generate WeChat Id and Secrete

**Package name:** package name of your application.

* **On completion of above steps share same info with CHINA account management team [as of now <**[**feihong.chen@philips.com**](mailto:feihong.chen@philips.com)**>]**
* **This team provide WeChat AppID and AppSecrete.**

## Application Side

**Modify: AppConfig.json and add weChatAppId , weChatAppSecret and "SigninProviders.CN": ["wechat"] properties**

*"UserRegistration": {*

*"SigninProviders.CN": ["wechat"],*

*"weChatAppId" : "\*\*\*\*",*

*"weChatAppSecret":"\*\*\*\*"*

*}*

***Note :****Currently china country only supporting WeChat social login so mentioned only for CN country code.*

* **Copy and paste WXEntryActivity  Activity in your application package followed by wxapi [Ex:com.philips.cdp.registration.wxapi.WXEntryActivity]**

Provide the Wechat listener activity in your package. Copy and paste the WXEntryActivity.java in ApplicationPackage.wxapi package .If miss or provided in other package Wechat Authentication does not work.

* **Register WXEntryActivity  activity in Manifest file.**
* **Progaurd**

make –keep for this Activity if obfuscated then we can’t listen access token form the  WeChatApp.

Add rules of wechat jar file

*-keep class com.tencent.mm.sdk.openapi.WXMediaMessage {\*;}*

*-keep class com.tencent.mm.sdk.openapi.\*\* implements com.tencent.mm.sdk.openapi.WXMediaMessage$IMediaObject {\*;}*

*-keep class com.janrainphilips.philipsregistration.wxapi.\*\* {\*;}*

* **WeChat feature is tightly coupled with apps signature so app should be always run in release mode .[i.e. app should be always build in release with signing with keystore]**

# Google+ integration

Google+ social provider is supported in UR module now with Web browser sign in .To enable Google+ for your application following are steps mandatory to work as desired.

## Prerequisites

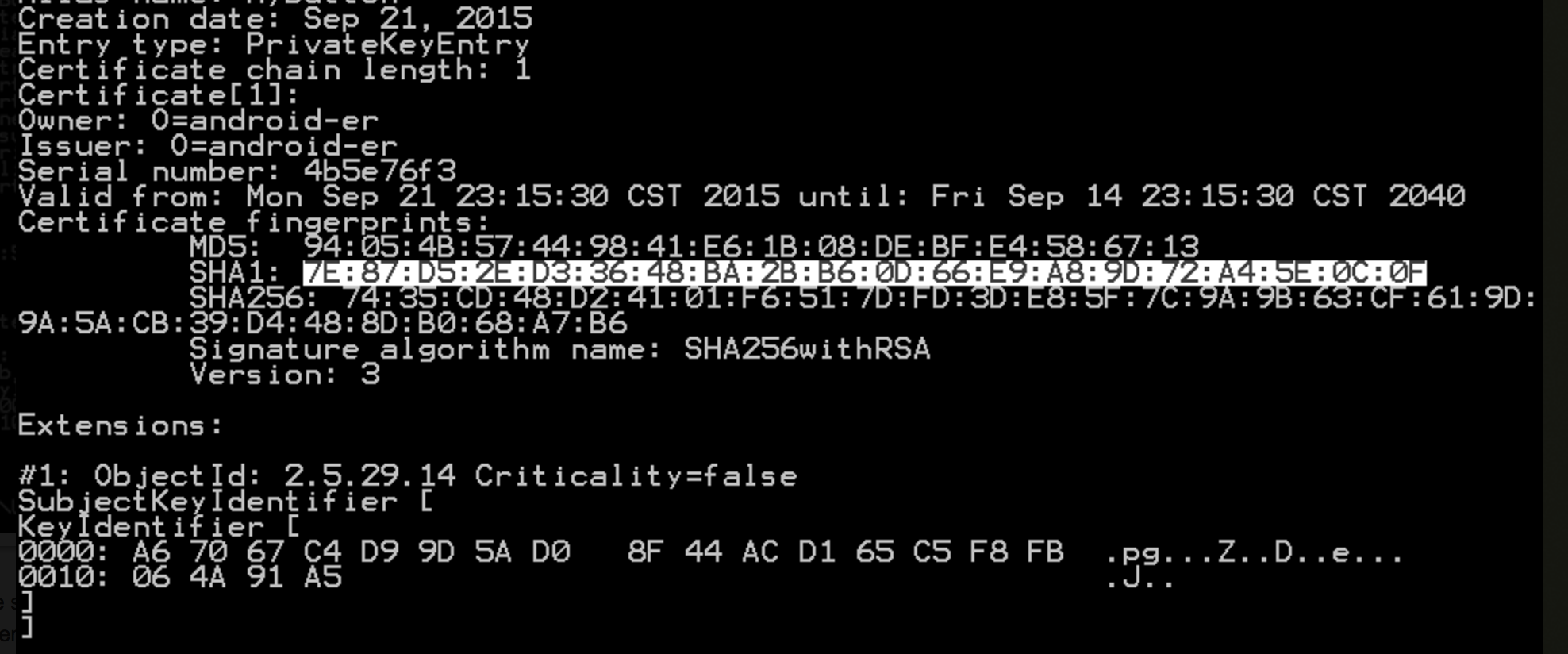
* **Support libraries**
* **OAuth 2.0 client IDs for your project**

**Support libraries**

This is strongly depend on the android Support library for google web authentication. If referring individually make sure referring to relevant support libs. compile **'net.openid:appauth:0.4.1'**explicitly .It is maily depend on Chromes <https://developer.chrome.com/multidevice/android/customtabs>. compile 'com.android.support:customtabs:\*\*.\*\*.\*\*'

**OAuth 2.0 client IDs for your project**

To get **OAuth 2.0 client IDs** from the Google developer site there is need of SHA1 signature and App Package name.

**SHA1 signature :**

Signing-certificate fingerprint is generated by command below :

**Command :** **keytool -exportcert -keystore path-to-debug-or-production-keystore -list –v**

SHA1 Fingerprint is generated keep note of this.

**Package name:** Package name of your application.

This is one of critical information for Google Console for generating client ids their will not be double client ids generated for same package name. So for multiple flavours or Janrain different environments there is different package name should be provided .During the client id request .

**Note:** Managing the different client id and other configuration for different product flavours are responsibility of respective project.

* **On completion of above steps share same info DS team**

Request the Google+ client id from DS team, by raising a ticket per environment. Please use this form for request. [Google Plus Authentication client id form](https://docs.philips.com/personal/viswarad_reddy_philips_com/_layouts/15/guestaccess.aspx?guestaccesstoken=u3p0N3CDPdizz45n0L63ARaWvd0jmBuukiCdRHzbyrg%3d&folderid=2_1761d3919c76745fb9ed16979646b6739&rev=1)

* **DS team provide Google + Client ids .**
  + **Sample output for client ids json**

**{"installed":{"client\_id":"346000571262-saojov4uqs4aiuk1rmn7nu68501tbost.apps.googleusercontent.com","project\_id":"storied-bearing-854","auth\_uri":"https://accounts.google.com/o/oauth2/auth","token\_uri":"https://accounts.google.com/o/oauth2/token","auth\_provider\_x509\_cert\_url":"https://www.googleapis.com/oauth2/v1/certs","redirect\_uris":["urn:ietf:wg:oauth:2.0:oob","http://localhost"]}}**

## Application Side

### Providing Client ids in resource files

Create openid\_appauth\_idp\_configs.xml in values folder and paste the below content. Replace with DS team provided correctly. Don’t change the key of string and don’t do translation on string.

*<?***xml version="1.0" encoding="utf-8"***?>*<**resources**>  
 *<!--  
 This contains the authorization service configuration details that are used to demonstrate  
 authentication. By default, all authorization services are disabled until you modify this file  
 to provide your own configuration details.  
 -->* <**eat-comment**/>  
 <**bool name="google\_enabled"**>true</**bool**>  
 <**string name="google\_client\_id" translatable="false"**>346000571262-m5u27lu130n1h1i2vbsggo34ae6nr1hp.apps.googleusercontent.com</**string**>  
 *<!--  
 NOTE: This scheme is automatically provisioned by Google for Android OAuth2 clients, and is  
 the reverse form of the client ID registered above. Handling of this scheme is registered in an  
 intent filter in the app's manifest.  
 -->* <**string name="google\_auth\_redirect\_uri" translatable="false"**>com.googleusercontent.apps.346000571262-m5u27lu130n1h1i2vbsggo34ae6nr1hp:/oauth2redirect</**string**>  
</**resources**>

### Register Google Auth activities in manifest files

Add below activity in your manifest:

<**activity  
 android:name="net.openid.appauth.RedirectUriReceiverActivity"** >  
 <**intent-filter**>  
 <**action android:name="android.intent.action.VIEW"**/>  
 <**category android:name="android.intent.category.DEFAULT"**/>  
 <**category android:name="android.intent.category.BROWSABLE"**/>  
 <**data android:scheme="https"  
 android:host="com.googleusercontent.apps.346000571262-m5u27lu130n1h1i2vbsggo34ae6nr1hp"  
 android:path="/oauth2redirect"**/>  
 </**intent-filter**>  
</**activity**>

### Provide the manifest place holders in build.gradle:

Add manifestPlaceholders in defaultConfig section:

*// replace the below string with your own Google client ID. Make sure this is consistent  
// with the values used in openid\_appauth\_idp\_configs.xml*manifestPlaceholders = [  
 **'appAuthRedirectScheme'**: **'com.googleusercontent.apps.346000571262-m5u27lu130n1h1i2vbsggo34ae6nr1hp'**]

### Make sure

Make sure values in resource, build.gradle and manifest file keys are same else Google + Authentication will not work.

# Coppa Flow Integration

Create an instance of CoppaExtension

CoppaExtension coppaExtension = **new** CoppaExtension(**mContext**);

coppaExtension.buildConfiguration();

**Note:** buildConfiguration() is the prior call to other APIs

**9.1 Get Consent:**

To get the consent information of the User if any Coppa consents taken from user.

Consent c = coppaExtension.getConsent();

**9.2 Update Consent:**

To update first consent value on user data,

coppaExtension.updateCoppaConsentStatus(**true/false**,new **new** CoppaConsentUpdateCallback(){})

**9.3 Update consent confirmation:**

To update second consent confirmation (should do only after 24 of first consent and the second consent is only for US)

coppaExtension. updateCoppaConsentConfirmationStatus

(**true/false**,new **new** CoppaConsentUpdateCallback(){})

# Launching Registration component

* As Activity/As Fragment
* Different UI flows(ABTest)

## Launch with different UI flows(ABTest)

User Registration now support for Server come Configurable UI flow for different UI flow for user AB testing. This can be done by server and configurable in URLaunch input API provided in URLaunch input. If not set in launch mode then it will be taken always form Server if server also not configured in AppConfig.json then it will be always type A.If you want to fix a flow type at launch, you can use the highlighted method shown below.

The flow type is sent as a parameter to the URLaunchInput’s setUiFlow method.

This will override the server setting.

urLaunchInput = **new** URLaunchInput();  
urLaunchInput.setEndPointScreen(RegistrationLaunchMode.***MARKETING\_OPT***);  
urLaunchInput.setRegistrationFunction(RegistrationFunction.***Registration***);  
urLaunchInput.setRegistrationContentConfiguration(getRegistrationContentConfiguration());

urLaunchInput.setUIFlow(UIFlow.***FLOW\_B***);

# Call-back’s

* On Click of Continue button in Welcome screen
* On Click of Terms & Conditions in Home screen
* On Click of Privacy policy in Home screen
* On Logout Success
* On Logout Failed
* On Logout success, due to invalid access token

Vertical project should implement UserRegistrationUIEventListener which have :

*@Override  
public void onUserRegistrationComplete(Activity activity) {*

*//Called on click of continue button   
}  
@Override  
public void onPrivacyPolicyClick(Activity activity) {  
//Called on click of privacy policy  
}  
@Override  
public void onTermsAndConditionClick(Activity activity) {  
 // Called on click of terms and condition   
}*

And needs to register and unregister listener as follows along with UILaunch input.

*urLaunchInput = new URLaunchInput();  
urLaunchInput.setUserRegistrationUIEventListener(this);*

Example

In onCrate of implementing Activity

*User mUser = new User(mContext);  
mUser.registerUserRegistrationListener(this);*

In onDestoy of calling Activity.

*RegistrationHelper.getInstance().unRegisterUserRegistrationListener(this)*

*mUser.unRegisterUserRegistrationListener(this);*

Vertical project should implement UserRegistrationEventListener which have

*@Override  
public void onUserLogoutSuccess() {  
//Called on logout success  
}  
@Override  
public void onUserLogoutFailure() {  
//on logout failed   
}  
@Override  
public void onUserLogoutSuccessWithInvalidAccessToken() {  
//on login success on invalid access token   
}*

And needs to register and unregister listener as follows

Example

In onCrate of implementing Activity

*User mUser = new User(mContext);  
mUser.registerUserRegistrationListener(this);*

In onDestoy of calling Activity.

*RegistrationHelper.getInstance().unRegisterUserRegistrationListener(this)*

*mUser.unRegisterUserRegistrationListener(this);*

# GDPR Consents

  More details on GDPR can be found [here](<https://confluence.atlas.philips.com/display/HDC/GDPR>)

Proposition can fetch consent definition by using “fetchMarketingConsentDefination()” method. User registration component will provide the consent definition to proposition. Even proposition can also define their own consent definition by using “USER\_MARKETIMNG\_CONSENT” same key.

**If user changes the language from Device Setting, then proposition should handle the configuration change and restart the app to update marketing consent locale in Janrain backend.**

# MyDetails Screen

This screen shows the details of a logged-In User and can be enabled using the endpoint API while launching User Registration.

Below is the code snippet to enable: -

urLaunchInput.setEndPointScreen(RegistrationLaunchMode.***USER\_DETAILS***);

# Refresh User Session

As we have call back for the refresh user session as success or failure. Vertical app has to define the scope of what should refresh failed scenarios as specially when logout happened due to expire or user token and user have to re-login .

Callbacks:

**void** onRefreshLoginSessionSuccess();  
**void** onRefreshLoginSessionFailedWithError(**int** var1);  
**void** onRefreshLoginSessionInProgress(String var1);

# HSDP specific API’s

HSDP UUID and access token can be accessed from the DIUserProfile.java

## HSDP UUID

*User user = new User(context);  
user.getUserInstance(context).getHsdpUUID();*

## HSDP Access Token

*User user = new User(context);  
user.getUserInstance(context).getHsdpAccessToken();*

# Library version’s

Use following methods to get respective versions of registration

## Janrain Library

***Jump.getJumpVersion()***

## Locale Match Library

***PILLocaleManager.getLacaleMatchVersion()***

## Registration Library

***RegistrationHelper.getRegistrationApiVersion()***

# Deprecated APIs

**Deprecated:**

public User(Context context) Will be removed in after this PI release .

**New:**

public User()

**Deprected :**

public void addConsumerInterest(AddConsumerInterestHandler addConsumerInterestHandler,ConsumerArray consumerArray) {} Will be removed by next PI

**Deprected :**

UserWithProduct,HttpClient complete class will be removed

**Deprecated :**

public void setAccountSettings(boolean isAccountSettings)

public boolean isAccountSettings()

**New :**

public void setEndPointScreen(RegistrationLaunchMode registrationLaunchMode) {

public RegistrationLaunchMode getEndPointScreen()