

Registration Android Integration

Document History				
Version	Date	Author	Section	Changes
0.1	08-07-2015	Vinayakkumar G Udikeri	All	Initial draft
0.2	28-07-2015	Vinayakkumar G Udikeri	All	Initial draft
0.3	13-08-2015	Vinayakkumar G Udikeri	All	Initial draft
0.4	21-10-2015	Vinayakkumar G Udikeri	All	HSDP incorporation and configuration changes
0.5	04-12-2015	Vinayakkumar G Udikeri	All	Dependant libs update ,Flow configuration,Janrain Initialization,Tagging,Proguard,Callbacks
0.6	04-02-2016	Vinayakkumar G Udikeri		Dynamic Configuration
0.7	04-03-2016	Vinayakkumar G Udikeri		Artifactory Support and More UI Configuration with Color

Author	Vinayakkumar G Udikeri
Approved by	
Email Id	vinayak.kumar.udikeri@philips.com

1. Introduction	4
2. Prerequisites	4
3. Integration	4
3.1 Referring from Artifactory	5
3.1.1 Changes at top level build.gradle	6
3.1.2 Changes at app level build.gradle	7
3.2 Referring from Local libs	8
3.2.1 Dependencies	9
3.3 Integration jar/aar files	9
3.3.1 Select Project mode view for Project	9
3.3.2 Paste .jar/.aar files to libs directory from reference app.	10
3.3.3 Select build.gradle file	11
3.3.4 Refer Registration dependencies along with .aar files in build.gradle as mentioned below	11
3.3.5 Clean project	12
3.3.6 Paste Registration related assets in assets folder from reference app	12
3.3.7 Repeat step 3.2.6	13
3.3.8 Android Manifest Changes	13
4. Configuration	13
4.1 UI Configurations [registration_config.xml]	14
4.1.1 Copy registration_config.xml from reference app project and paste into project's values folder and alter as per vertical need	17
4.2 Flow Configurations	22
4.2.1 Standard	22
4.2.2 COPPA	22
4.2.3 HSDP	22
4.3 Registration Configuration	23
4.3.1 Static Configuration	23
4.3.2 Dynamic Configuration	28
4.3.3 Hybrid Configuration	35
5. Janrain Initialization	36
5.1 UI Priority function	36
5.1.1 Create account on Top	36
5.1.2 Sign In On top	37
5.2 STANDARD and HSDP configuration initialization	38
5.3 COPPA configuration initialization	39
5.4 Custom Language in Registration UI	39

6. Launching Registration component	39
6.1 Activity	40
6.1.1 Launching with out orientation	40
6.1.2 Launching with orientation lock	40
6.2 Fragment	41
6.2.1 Launching as fragment	41
6.2.2 Backstack handling	42
6.2.3 Updating Title	42
7. Call-back's	43
8. APP tagging	44
9. COPPA specific API's	45
10. HSDP specific API's	46
10.1 HSDP UUID	46
10.2 HSDP Access Token	46
11. Library version's	46
11.1 Janrain Library	46
11.2 Locale Match Library	46
11.3 Registration Library	46
12. Supporting apps with Over 65K Methods	46
13. Progaurd rules for Registration API component	47
13.1 Hockey SDK	<i>Error! Bookmark not defined.</i>
13.2 Tagging	47
13.3 Jump	48
13.4 Localematch	49
13.5 Registration API	49
13.6 Network APIs	49
13.7 HSDP	49
13.8 Other platform specific	49
13.9 Example for configuration	50
14. Demo apps	51

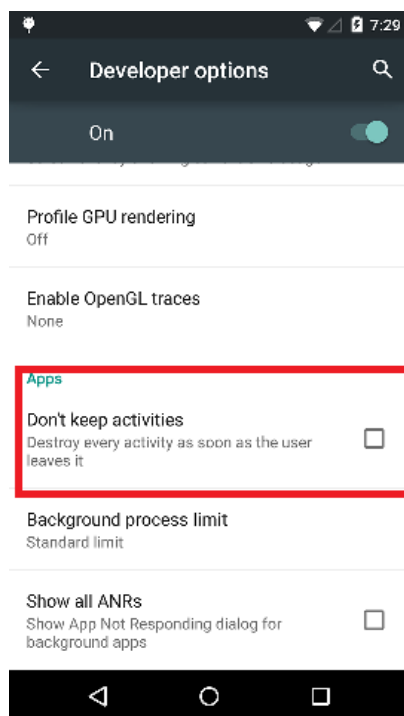
1. Introduction

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

Source Path: <http://161.85.28.146:9000/scm/git/hor-registration-android>

2. Prerequisites

- I. Vertical project is configured for Android Studio
- II. Setting->Developer Options->Don't Keep Activities should be unchecked.[If developer mode is on]
- III. Android API version should on 10[Gingerbread]



3. Integration

Integration can be done in following ways

3.1 Referring from Artifactory

Artifactory is software that The SOA and Internet development teams have been using for a while. If you have never heard of it, Artifactory is a binary repository manager. We use source control for our source code, Artifactory is version control (and more) for your binary artifacts (jar/war files, etc). Artifactory is also a place where you can put a shared library so that it is easily accessible in other projects across the enterprise.

For someone who has never used a binary repository and wants to understand the benefits of using a tool like Artifactory here is a great (short) presentation: Artifactory – Sharing Binaries the Smart Way!

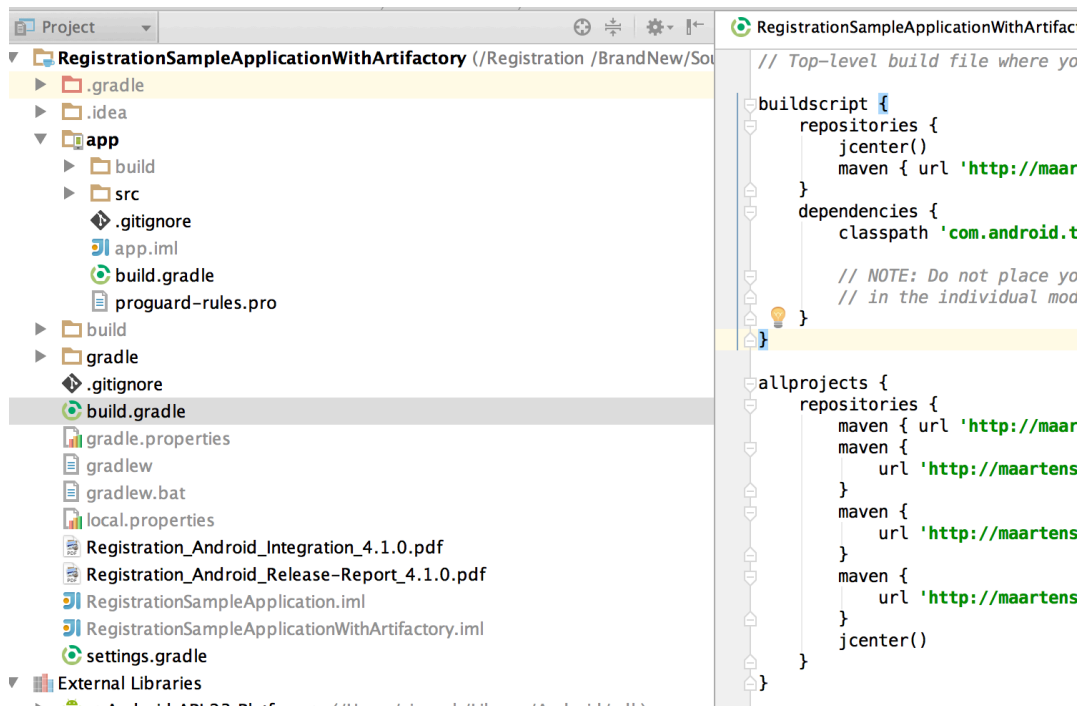
Why use Artifactory?

- Rebuilding from source introduces points of failure and change. By having an artifact in a binary repository you have a versioned copy of the tested artifact.
- Since the binaries are version it is easy to have multiple versions of libraries so that all projects do not have to be on the same version.
- Artifactory can be integrated with a CI server such as Jenkins so that clean builds are automatically published with meta data about the build.
- Artifactory has “virtual repositories” that are proxies to all of the “well known” maven repositories on the internet. This give you access to most of the public shared libraries such as Apache Commons, Spring, Hibernate, etc..; through one conduit.
- It is expensive and inefficient for every developer inside the same organization to go and retrieve remote artifacts that are shared by nature
- <https://www.youtube.com/watch?v=aa4YBDUDWy0>

So we are now with Artifactory support for User Registration Module

Steps to consume User registration following things needs to be done from App side

3.1.1 Changes at top level build.gradle



Need to mention the maven repo links and dependent url link in top build.gradle
Root build.gardle must contain following dependent links.

```
// Top-level build file where you can add configuration options common to all
sub-projects/modules.

buildscript {
    repositories {
        jcenter()
        maven { url 'http://maartens-mini.ddns.htc.nl.philips.com:8081/artifactory/jcenter' }
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:1.5.0'
    }
}

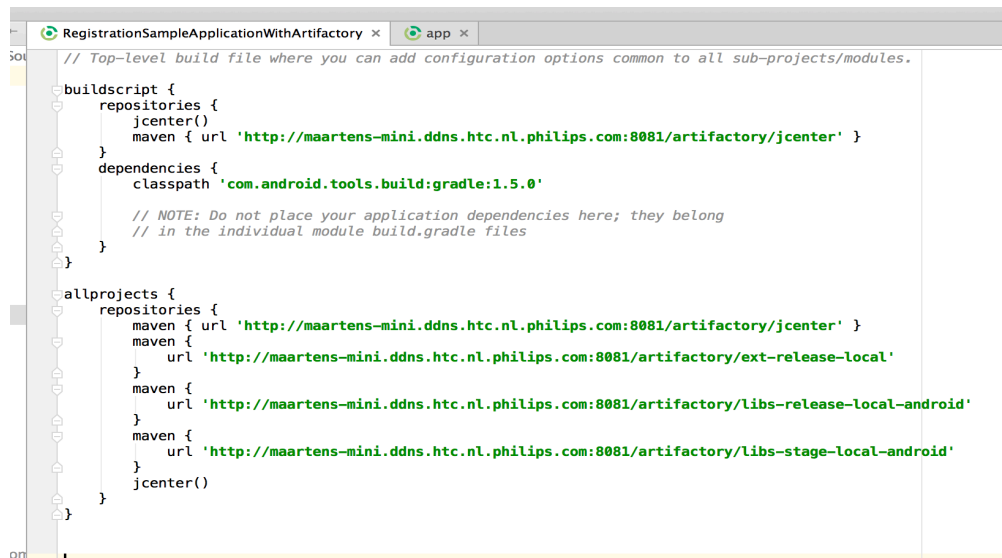
// NOTE: Do not place your application dependencies here; they belong
// in the individual module build.gradle files

allprojects {
    repositories {
        maven { url 'http://maartens-mini.ddns.htc.nl.philips.com:8081/artifactory/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()
}
}
}

```



3.1.2 Changes at app level build.gradle

Need to mention artifactory dependences for registration in dependencies {} block

It can be mentioned in any one of the following ways

- i. `compile(group:'com.philips.cdp', name:'registrationApi', version:'5.0.0')`
- ii. `compile(group: 'com.philips.cdp', name: 'registrationApi', version: "5.0.0", ext: 'aar') {
 exclude group: 'com.android.support'
 transitive = true
}`
- iii. `compile 'com.philips.cdp:registrationApi:5.0.0'`

- iv. `apply plugin: 'com.android.application'`

```

android {
    compileSdkVersion 23

```

```

        buildToolsVersion "23.0.2"

        packagingOptions {
            exclude 'META-INF/DEPENDENCIES.txt'
            exclude 'META-INF/LICENSE.txt'
            exclude 'META-INF/NOTICE.txt'
            exclude 'META-INF/NOTICE'
            exclude 'META-INF/LICENSE'
            exclude 'META-INF/DEPENDENCIES'
            exclude 'META-INF/notice.txt'
            exclude 'META-INF/license.txt'
            exclude 'META-INF/dependencies.txt'
            exclude 'META-INF/LGPL2.1'
        }

        defaultConfig {
            applicationId "com.philips.cdp.registration.demo.app"
            minSdkVersion 15
            targetSdkVersion 23
            versionCode 1
            versionName "1.0"
        }
        buildTypes {
            release {
                minifyEnabled false
                proguardFiles getDefaultProguardFile('proguard-android.txt'),
'proguard-rules.pro'
            }
        }
    }

    dependencies {
        compile fileTree(dir: 'libs', include: ['*.jar'])
        compile 'com.android.support:appcompat-v7:23+'

        compile(group: 'com.philips.cdp', name: 'registrationApi', version:
"5.0.0", ext: 'aar') {
            exclude group: 'com.android.support'
            transitive = true
        }
    }
}

```

Note : Demo app is available for reference along with this package

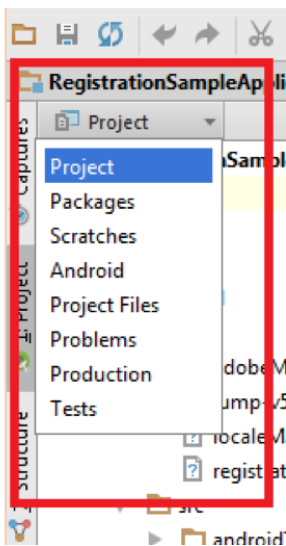
3.2 Referring from Local libs

3.2.1 Dependencies

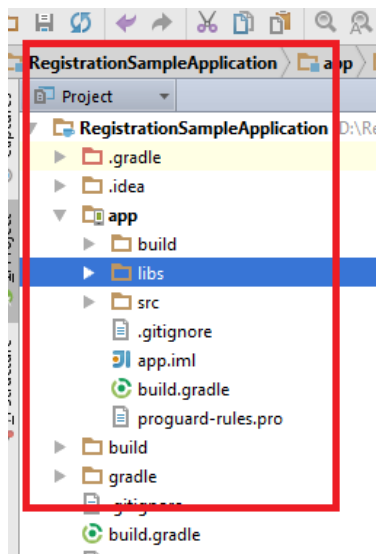
- jump.aar[v.5.0.3]
- localeMatch.aar[v.1.1.1]
- registrationApi.aar[v.5.0.0]
- tagging.aar[v.2.0.0]
- hsdp.aar[v.1.0.2]
- ntpClient.aar[v.1.0.0]
- adobeMobileLibrary.jar[v.4.5.1]
- org.apache.http.legacy.jar
- jackson-annotations.jar[v.2.6.0]
- jackson-core.jar[v.2.6.1]
- jackson-databind.jar[v.2.6.1]

3.3 Integration jar/aar files

3.3.1 Select Project mode view for Project

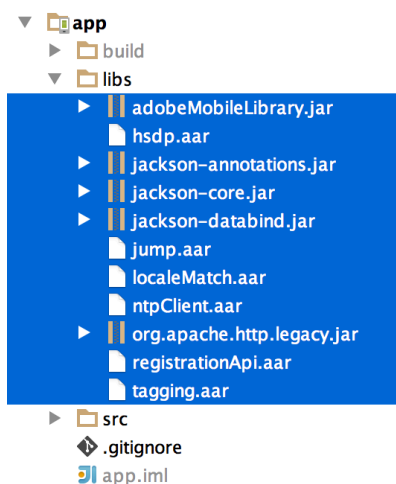


Crate lib folder if not exists.

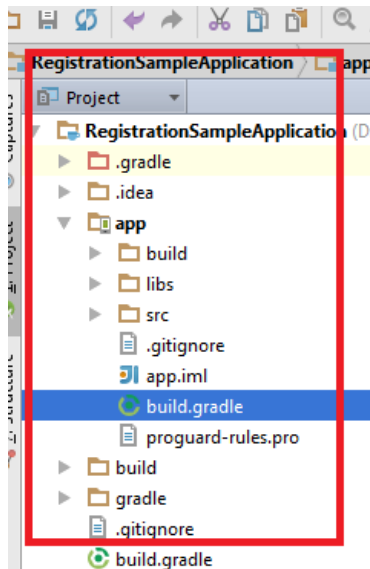


3.3.2 Paste .jar/.aar files to libs directory from reference app.

- i. jump.aar
- ii. localeMatch.aar
- iii. registrationApi.aar
- iv. tagging.aar
- v. hsdp.aar
- vi. ntpClient.aar
- vii. adobeMobileLibrary.jar
- viii. org.apache.http.legacy.jar
- ix. jackson-annotations.jar
- x. jackson-core.jar
- xi. jackson-databind.jar



3.3.3 Select build.gradle file



3.3.4 Refer Registration dependencies along with .aar files in build.gradle as mentioned below

Note: Since Registration Android was built with 'com.android.support:appcompat-v7:23+'. version so vertical app support libs should be greater or equal to mentioned versions

```
android {
    packagingOptions {
        exclude 'META-INF/DEPENDENCIES.txt'
        exclude 'META-INF/LICENSE.txt'
        exclude 'META-INF/NOTICE.txt'
        exclude 'META-INF/NOTICE'
        exclude 'META-INF/LICENSE'
        exclude 'META-INF/DEPENDENCIES'
        exclude 'META-INF/notice.txt'
        exclude 'META-INF/license.txt'
        exclude 'META-INF/dependencies.txt'
        exclude 'META-INF/LGPL2.1'
    }
}
repositories {
    flatDir {
        dirs 'libs'
    }
}
dependencies {
```

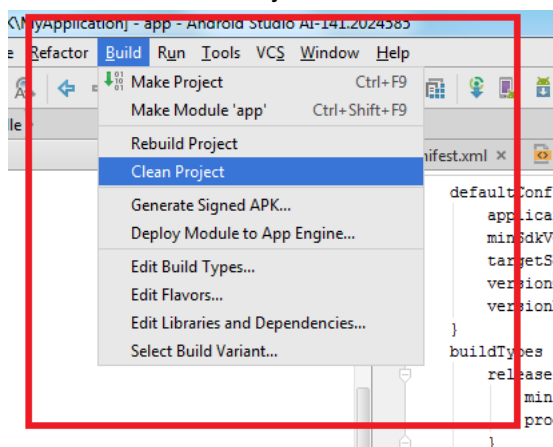
```

compile fileTree(dir: 'libs', include: ['*.jar'])
compile 'com.android.support:appcompat-v7:23+'
//Dependant jars
compile files('libs/adobeMobileLibrary.jar')
compile files('libs/org.apache.http.legacy.jar')
compile files('libs/jackson-annotations.jar')
compile files('libs/jackson-core.jar')
compile files('libs/jackson-databind.jar')
//Dependant aars
compile(name: 'localeMatch', ext: 'aar')
compile(name: 'jump', ext: 'aar')
compile(name: 'registrationApi', ext: 'aar')
compile(name: 'tagging', ext: 'aar')
compile(name: 'ntpClient', ext: 'aar')
compile(name: 'hsdp', ext: 'aar')
}

```

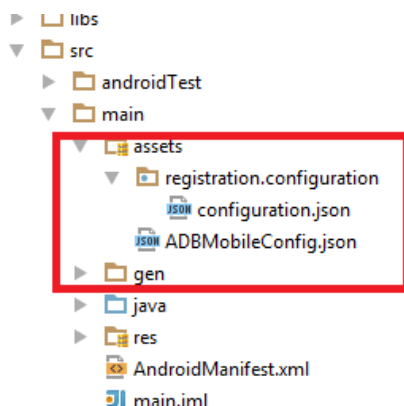
3.3.5 Clean project

Build->Clean Project



3.3.6 Paste Registration related assets in assets folder from reference app

Note : Don't change or rename folder structure



3.3.7 Repeat step 3.2.6

3.3.8 Android Manifest Changes

To support Registration Android following changes needs to be done mandatorily

- i. Needs to mention additional xmlns
`xmlns:tools="http://schemas.android.com/tools"`
- ii. Project should have one application class and should be defined in manifest.
 Ex : `android:name=".RegistrationApplication"`
- iii. Asset merging preference
`tools:replace="android:icon,android:theme"`

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  package="com.philips.cdp.registration.sample">

  <application
    android:name="com.philips.cdp.registration.sample.RegistrationApp1
    ication"
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme"
    tools:replace="android:icon,,android:label,android:theme">
    <activity

    android:name="com.philips.cdp.registration.sample.RegistrationSamp
    leActivity"
    android:label="@string/app_name">
    <intent-filter>
      <action android:name="android.intent.action.MAIN"

      <category
        android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
    </activity>
    </application>

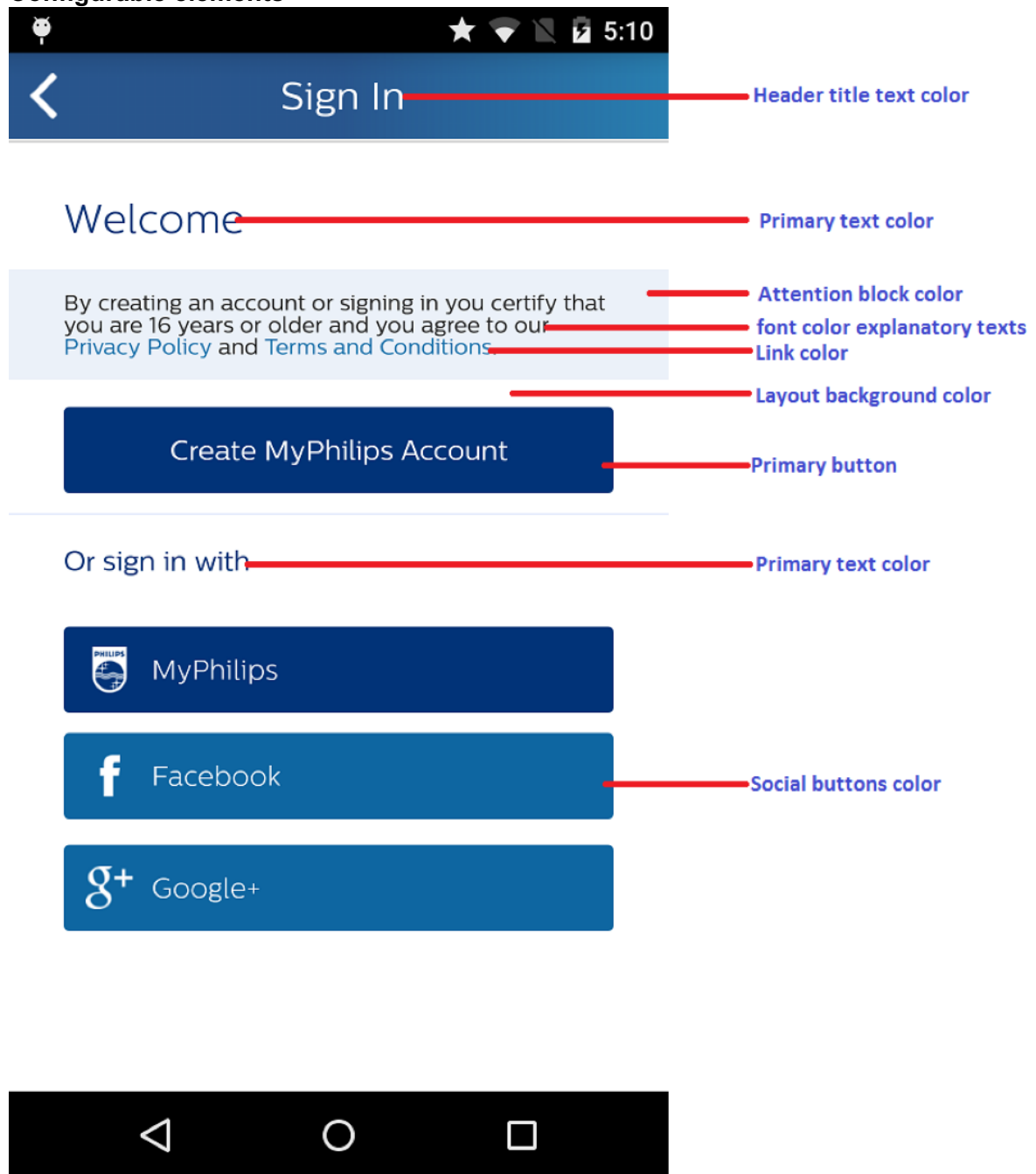
  </manifest>
```

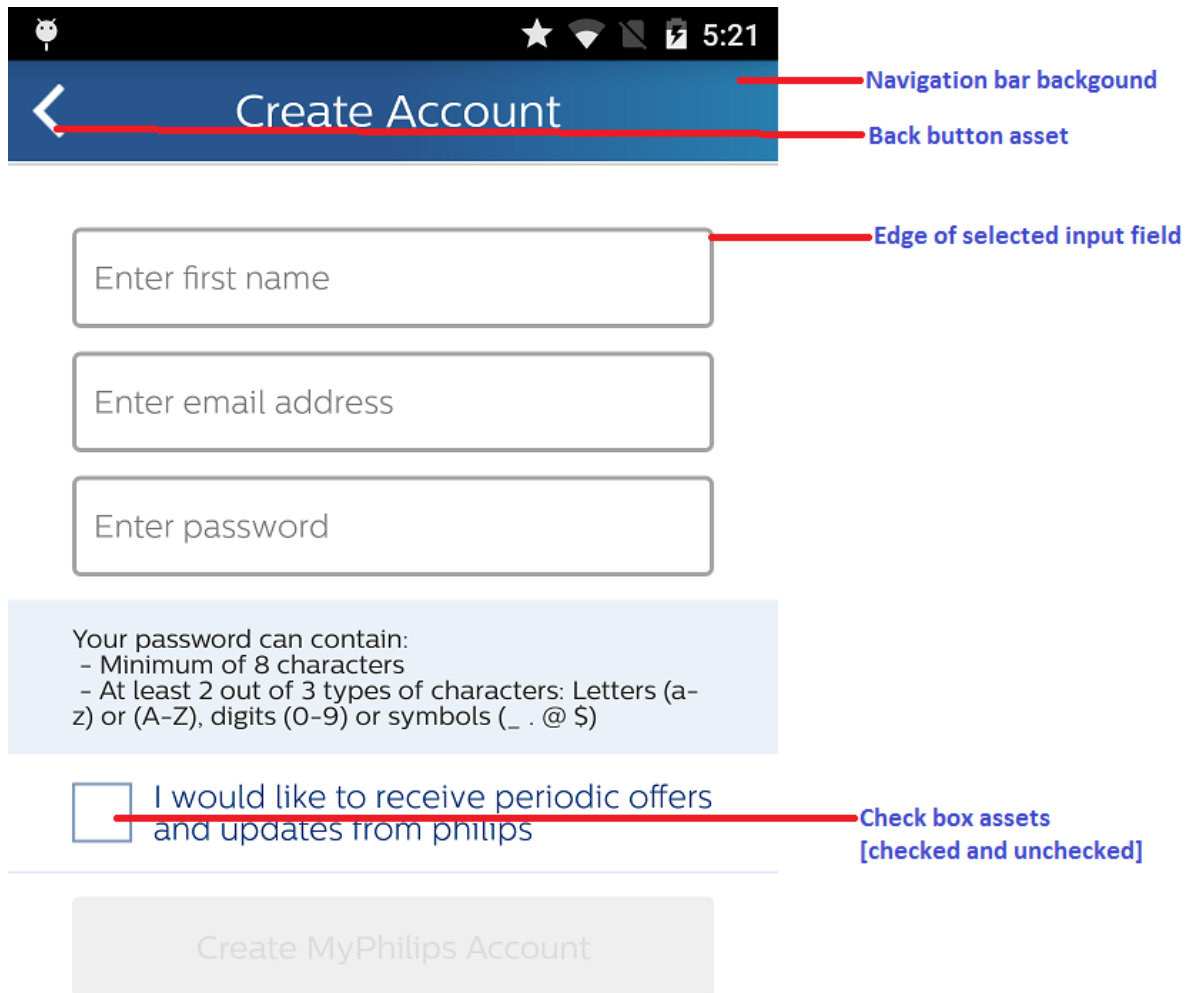
4. Configuration

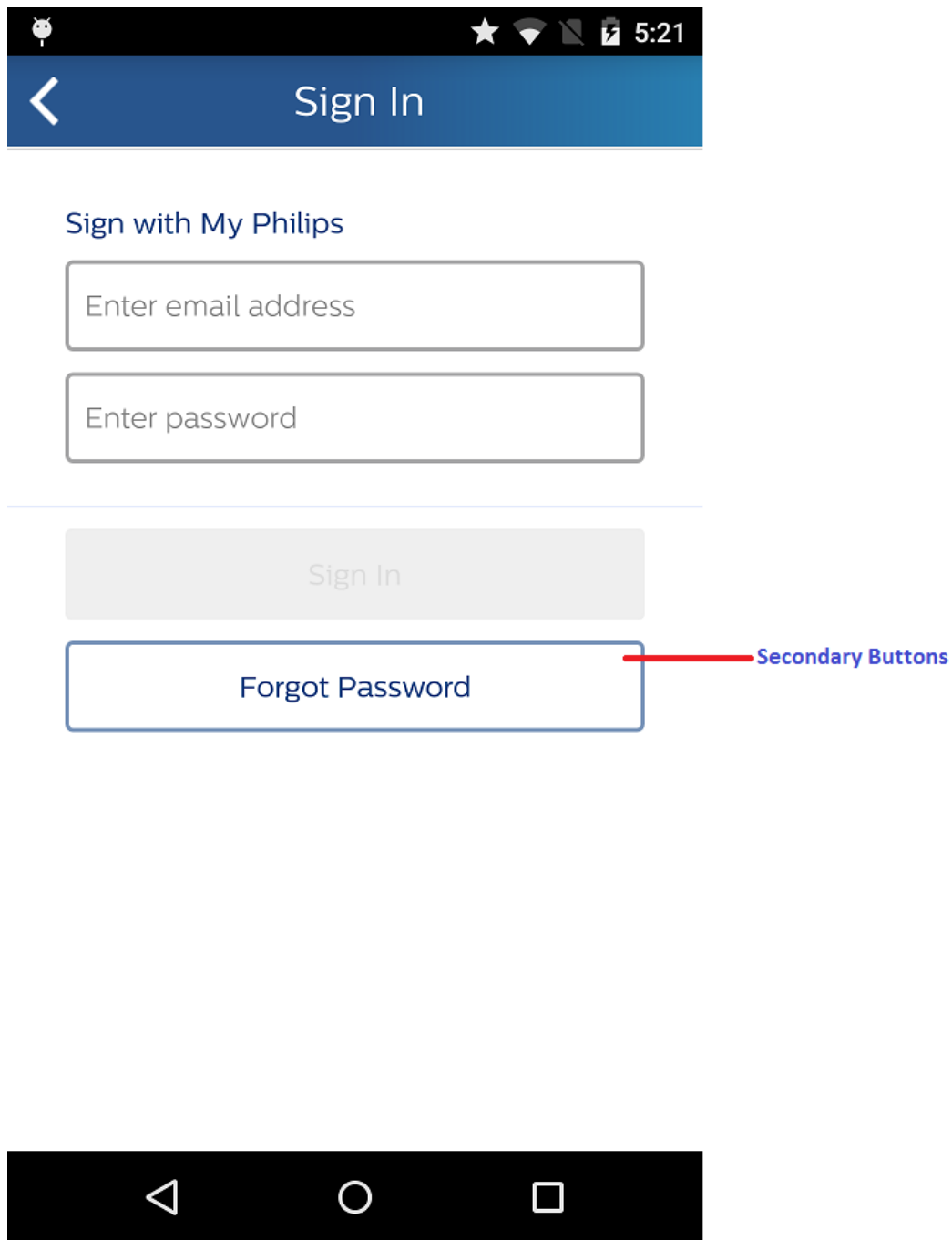
Registration Android can be configurable in two types UI and Flow configurations

4.1 UI Configurations [registration_config.xml]

Configurable elements



A screenshot of a mobile application's "Create Account" screen. The screen has a dark blue navigation bar at the top with a white back arrow icon on the left and the text "Create Account" in the center. Below the navigation bar are three white input fields with gray borders, labeled "Enter first name", "Enter email address", and "Enter password". Below these fields is a light blue box containing text about password requirements: "Your password can contain: - Minimum of 8 characters - At least 2 out of 3 types of characters: Letters (a-z) or (A-Z), digits (0-9) or symbols (_ . @ \$)". Below this box is a checkbox followed by the text "I would like to receive periodic offers and updates from philips". At the bottom is a large, light gray button labeled "Create MyPhilips Account". Red lines point from various labels to specific UI elements: "Navigation bar background" points to the dark blue bar, "Back button asset" points to the back arrow, "Edge of selected input field" points to the right edge of the first input field, and "Check box assets [checked and unchecked]" points to the checkbox.



The image shows a mobile application interface for a sign-in screen. At the top, there is a status bar with an Android robot icon, a star, Wi-Fi and battery icons, and the time 5:21. Below this is a blue header bar with a white back arrow on the left and the text "Sign In" in the center. The main content area has a light gray background. It starts with the text "Sign with My Philips" in blue. Below this are two white input fields with gray borders; the first contains the placeholder text "Enter email address" and the second contains "Enter password". A thin horizontal line separates these from the buttons below. There are two buttons: a gray "Sign In" button and a white "Forgot Password" button with a blue border. A red line points from the text "Secondary Buttons" to the "Forgot Password" button. At the bottom of the screen is a black navigation bar with three white icons: a triangle, a circle, and a square.

Sign In

Sign with My Philips

Enter email address

Enter password

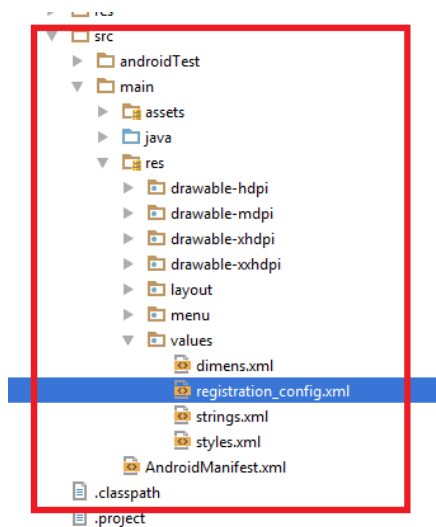
Sign In

Forgot Password

Secondary Buttons

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

4.1.1 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

4.1.1.1 Primary Button

Vertical specific configuration:

```
<!-- All primary buttons -->
<!-- Primary button enabled color -->
<color name="reg_btn_bg_enabled_color">#013378</color>
<!-- Primary button disabled color -->
<color name="reg_btn_bg_disabled_color">#EFFFFFF</color>
<!-- Primary button pressed color -->
<color name="reg_btn_pressed_color">#0F204B</color>
```

Default configuration:

```
<!-- All primary buttons -->
<!-- Primary button enabled color -->
<color name="reg_btn_bg_enabled_color">#013378</color>
<!-- Primary button disabled color -->
<color name="reg_btn_bg_disabled_color">#EFFFFFF</color>
<!-- Primary button pressed color -->
<color name="reg_btn_pressed_color">#0F204B</color>
```

4.1.1.2 Edge of input field's

Vertical specific configuration:

```
<!-- Edge of selected input field -->
<!-- Edit Text box border highlight color -->
<color name="reg_et_highlight_color">#28558D</color>
```

Default configuration:

```
<color name="reg_et_highlight_color">#28558D</color>
```

4.1.1.3 Secondary button color**Vertical specific configuration:**

```
<!-- Secondary buttons (forgot password) -->
<!-- Forgot password button -->
<!-- Forgot password button border color -->
<color name="reg_btn_forgot_border">#7793B9</color>
<!-- Forgot password highlight border color -->
<color name="reg_highlight_forgot_bg">#7793B9</color>
```

Default configuration:

```
<!-- Secondary buttons (forgot password) -->
<!-- Forgot password button -->
<!-- Forgot password button border color -->
<color name="reg_btn_forgot_border">#7793B9</color>
<!-- Forgot password highlight border color -->
<color name="reg_highlight_forgot_bg">#7793B9</color>
```

4.1.1.4 Attention block color**Vertical specific configuration:**

```
<!-- Attention block color -->
<!-- Description text background color -->
<color name="reg_text_desc_bg">#EBF1F9</color>
```

Default configuration:

```
<!-- Description text background color -->
<color name="reg_text_desc_bg">#EBF1F9</color>
```

4.1.1.5 Link color**Vertical specific configuration:**

```
<!-- Link color -->
<color name="reg_hyperlink_highlight_color">#1470AA</color>
```

Default configuration:

```
<color name="reg_hyperlink_highlight_color">#1470AA</color>
```

4.1.1.6 Navigation bar color**Vertical specific configuration:**

```
<!-- Navigation bar background -->
<!-- Starting color of Header:background start gradient color -->
<color name="reg_header_bg_start_color">#28558D</color>
<!-- Ending color of Header:background end gradient color -->
<color name="reg_header_bg_end_color">#287EB0</color>
```

Default configuration:

```
<!-- Navigation bar background -->
<!-- Starting color of Header:background start gradient color -->
<color name="reg_header_bg_start_color">#28558D</color>
```

```
<!-- Ending color of Header:background end gradient color -->
<color name="reg_header_bg_end_color">#287EB0</color>
```

4.1.1.7 Navigation bar font color

Vertical specific configuration:

```
<!-- Navigation bar font color -->
<!-- Header title text color -->
<color name="reg_header_text_color">#FFFFFF</color>
```

Default configuration:

```
<!-- Header title text color -->
<color name="reg_header_text_color">#FFFFFF</color>
```

4.1.1.8 Back button asset

Vertical specific configuration:

```
<!--Back arrow color -->
<color name="reg_back_arrow_color">#FFFFFF</color>
```

Default configuration:

```
<!--Back arrow color -->
<color name="reg_back_arrow_color">#FFFFFF</color>
```

4.1.1.9 Font color explanatory text

Vertical specific configuration:

```
<!-- Font color explanatory texts -->
<!-- Description text color -->
<color name="reg_text_desc_color">#252525</color>
```

Default configuration:

```
<!-- Font color explanatory texts -->
<!-- Description text color -->
<color name="reg_text_desc_color">#252525</color>
```

4.1.1.10 Layout background

Vertical specific configuration:

```
<!-- Background -->
<color name="reg_layout_bg">#FFFFFF</color>
```

Default configuration:

```
<!-- Background -->
<color name="reg_layout_bg">#FFFFFF</color>
```

4.1.1.11 Primary text color

Vertical specific configuration:

```
<!-- Primary text color -->
<color name="reg_text_heading_one_color">#003377</color>
```

Default configuration:

```
<!-- Primary text color -->
<color name="reg_text_heading_one_color">#003377</color>
```

4.1.1.12 Warning color**Vertical specific configuration:**

```
<!-- Warning color -->
<color name="reg_error_box_color">#E88200</color>
<color name="reg_err_msg_color">#E88200</color>
```

Default configuration:

```
<!-- Warning color -->
<color name="reg_error_box_color">#E88200</color>
<color name="reg_err_msg_color">#E88200</color>
```

4.1.1.13 Social button color**Vertical specific configuration:**

```
<!-- Social button enabled color -->
<color name="reg_btn_social_bg_enabled_color">#0E66A1</color>
<!-- Social button pressed color -->
<color name="reg_btn_social_highlight_color">#033D78</color>
```

Default configuration:

```
<!-- Social button enabled color -->
<color name="reg_btn_social_bg_enabled_color">#0E66A1</color>
<!-- Social button pressed color -->
<color name="reg_btn_social_highlight_color">#033D78</color>
```

4.1.1.14 Check box**Vertical specific configuration:**

```
<!-- Check box border color -->
<color name="reg_check_box_border_color">#C4150C</color>

<!-- Check box tick color -->
<color name="reg_check_box_tick_color">#C4150C</color>
```

Default configuration:

```
<!-- Check box border color -->
<color name="reg_check_box_border_color">#C4150C</color>

<!-- Check box tick color -->
<color name="reg_check_box_tick_color">#C4150C</color>
```

4.1.1.15 Error alert**Vertical specific configuration:**

```
<!-- Check error symbol color -->
<color name="reg_error_symbol_color">#E88200</color>
```

```
<!-- Check error exclamation symbol color -->
<color name="reg_exclamation_color">#FFFFFF</color>
```

Default configuration:

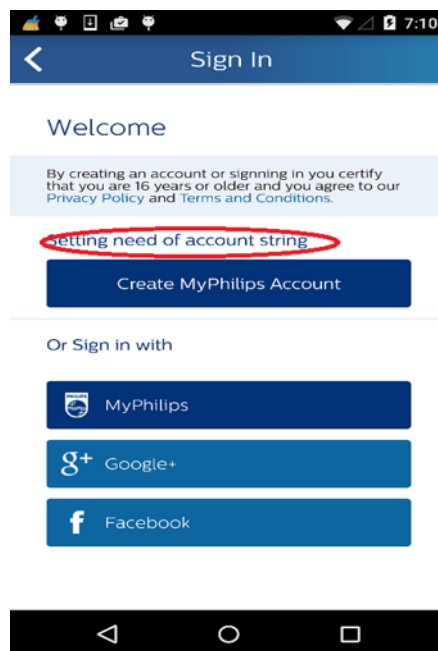
```
<!-- Check error symbol color -->
<color name="reg_error_symbol_color">#E88200</color>
```

```
<!-- Check error exclamation symbol color -->
<color name="reg_exclamation_color">#FFFFFF</color>
```

4.1.1.16 Vertical specific texts**4.1.1.16.1 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

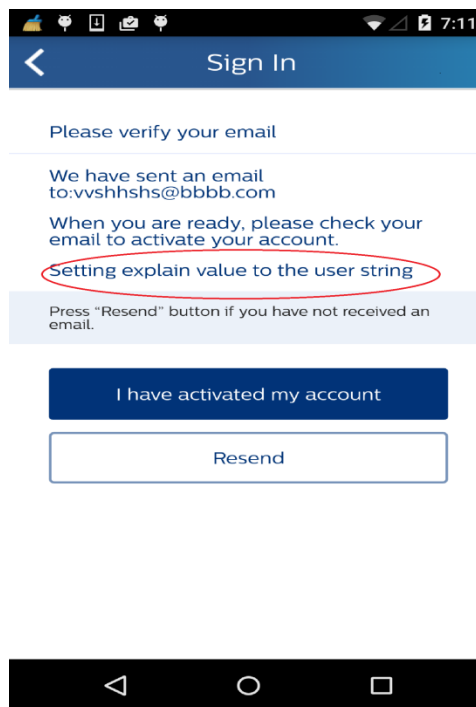
```
<string name="Welcome_needAccountto_lbltxt">Setting need of account string </string>
```



4.1.1.16.2 Setting explain value to the user string

Repeat 4.1.1.16.1 step but use string key as bellow mentioned

```
<string me="Activation_explain_value_to_user"></string>
```



4.2 Flow Configurations

4.2.1 Standard

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

4.2.2 COPPA

Additional parameter for enabling CoppaFlow is essential before init - The coppaFlow value need to set to **“true”** when the app need to use COPPA flow. *[Currently locale is hardcoded to en_US for COPPA please refer sample app]*

4.2.3 HSDP

No additional parameter required. Framework will take care of initializing It in HSDP flow. Depending on provided configuration.json

4.3 Registration Configuration

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

4.3.1 Static Configuration

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

The **configuration.json** should be added to the project and json need to update depend on the project.

- JanrainConfiguration
- PILConfiguration
- Flow
- HSDPConfiguration
- SigninProviders

Example RegistrationConfiguration.json:

```
{
  "JanRainConfiguration": {
    "RegistrationClientID": {
      "Development": "xxxxxxxxxxxxxxxxxxxx",
      "Testing": "xxxxxxxxxxxxxxxxxxxx",
      "Evaluation": "xxxxxxxxxxxxxxxxxxxx",
      "Production": "xxxxxxxxxxxxxxxxxxxx"
    }
  },
  "PILConfiguration": {
    "MicrositeID": xxxxxxxxx,
    "RegistrationEnvironment": " Evaluation",
    "CampaignID": "xxxxxxxxxxxxxxxxxxxx"
  },
  "Flow": {
    "EmailVerificationRequired": true
    "TermsAndConditionsAcceptanceRequired": "true",
    "MinimumAgeLimit": {
      "NL": 16,
      "GB": 14,
      "default": 16
    }
  },
  "HSDPConfiguration": {
    "Development": {
      "ApplicationName": "yourApplicationName",
      "Shared": "your-app's-Shared-Key-for-development",
      "Secret": "your-app's-Secret-Key-for-development",
      "BaseURL": "http://your_app_BaseURL_for_development"
    },
    "Testing": {
      "Shared": "your-app's-Shared-Key-for-testing",
      "Secret": "your-app's-Secret-Key-for-testing",
      "BaseURL": "http://your_app_BaseURL_for_testing"
    }
  },
  "SigninProviders": {
    "NL": [
      "myphilips",
      "provider_1",
      "provider_2",
      "provider_3"
    ],
    "default": [
      "myphilips",
      "provider_1",
      "provider_2"
    ]
  }
}
```


4.3.1.1 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

4.3.1.2 PILConfiguration (Mandatory)

PIL configuration requires following three elements:

MicrositeID	MicrositeID of the application.
	Mandatory field
RegistrationEnvironment	The environment want to use. Assign only Development, Testing, Evaluation or Production
	Mandatory Field
CampaignID	Add CampaignID only if it using COPPA Flow
	Optional field

Note: Please contact **Digital Services** <<mailto:digital.services@philips.com>> for Client ID for janrain and Microsite ID

4.3.1.3 Flow (Optional)

EmailVerificationRequired	If email verification is not needed please assign false to this property, Default value is TRUE
	Optional field
TermsAndConditionsAcceptanceRequired	If email terms and conditions must for app then please assign TRUE to this property, Default value is FALSE
	Optional field
MinimumAgeLimit	If Terms and condition field is assigned true then it is mandatory to define minimum age .
	Mandatory filed if Terms and condition filed is TRUE Else option filed.

4.3.1.4 HSDP Configuration (Optional)

HSDPConfiguration is required to use HSDP services. The elements of HSDP configuration are as below. All the elements are environment specific. The configuration is required for all the environments. If configuration were not provided for any of the environments, application would bypass any communication to HSDP server for that environment. Elements for any environment cannot be provided partially.

ApplicationName	The environment specific application display name
	Mandatory field
Shared	An application and environment specific Shared Key to authenticate the application with HSDP Server
	Mandatory field
Secret	An application and environment specific Secret Key to authenticate the application with HSDP Server
	Mandatory Field
BaseURL	An application and environment specific URL to be used to communicate with HSDP server
	Mandatory Field

4.3.1.5 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
twitter

Ex configuration.json for COPPA flow

```
{
  "JanRainConfiguration": {
    "RegistrationClientID": {
      "Development": "xxxxxxxxxxxxxxxxxxxx",
      "Evaluation": "xxxxxxxxxxxxxxxxxxxx",
      "Production": "xxxxxxxxxxxxxxxxxxxx"
    }
  },
  "PILConfiguration": {
    "MicrositeID": xxxxxxxx,
    "RegistrationEnvironment": " Evaluation",
  }
}
```

```

    "CampaignID": "xxxxxxxxxxxxxxxxx"
  },
  "Flow": {
    "EmailVerificationRequired": "true"
  },
  "SigninProviders": {
    "NL": [
      "twitter",
      "facebook",
      "googleplus"
    ],
    "default": [
      "facebook",
      "googleplus"
    ]
  }
}

```

Ex configuration.json for Standard flow

```

{
  "JanRainConfiguration": {
    "RegistrationClientID": {
      "Development": "xxxxxxxxxxxxxxxxxxxxx",
      "Evaluation": "xxxxxxxxxxxxxxxxxxxxx",
      "Production": "xxxxxxxxxxxxxxxxxxxxx"
    }
  },
  "PILConfiguration": {
    "MicrositeID": "xxxxxxxx",
    "RegistrationEnvironment": "Evaluation",
  },
  "Flow": {
    "EmailVerificationRequired": "true"
  },
  "SigninProviders": {
    "NL": [
      "twitter",
      "facebook",
      "googleplus"
    ],
    "default": [
      "facebook",
      "googleplus"
    ]
  }
}

```

4.3.2 Dynamic Configuration

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

This has been enabled with the RegistrationDynamicConfiguration class

Note : Dynamic configuration are take as priority over the static

4.3.2.1 *JanRainConfiguration*

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

API :

Present in **RegistrationDynamicConfiguration** class.

```

/**
 * Set Janrain Configuration
 *
 * @param janRainConfiguration
 */
public void setJanRainConfiguration(JanRainConfiguration janRainConfiguration) {}

/**
 * Get Janrain Configuration
 *
 * @return RegistrationDynamicConfiguration
 */
public JanRainConfiguration getJanRainConfiguration() {}

/**
 * Set Janrain client Ids
 *
 * @param clientIds Object of RegistrationClientId
 */
public void setClientIds(RegistrationClientId clientIds) {}

```

Present in **RegistrationClientId** class

```

/**
 * Set production id
 *
 * @return String
 */
public void setProductionId(String productionId) {}

/**
 * Set evaluation id
 *
 * @return String
 */
public void setEvaluationId(String evaluationId) {}

/**
 * Set development id
 *
 * @return String
 */
public void setDevelopmentId(String developmentId) {}

```

```

/**
 * Set testing id
 *
 * @return String
 */
public void setTestingId(String testingId) {}

```

```

/**
 * Set staging id
 *
 * @return String
 */
public void setStagingId(String stagingId) {}

```

Example :

```

RegistrationClientId registrationClientId = new RegistrationClientId();
registrationClientId.setEvaluationId("Evaluation Id");
registrationClientId.setDevelopmentId("Development Id");
registrationClientId.setStagingId("Staging Id");
registrationClientId.setTestingId("Testing Id");
registrationClientId.setProductionId("Production Id");

RegistrationDynamicConfiguration.getInstance().getJanRainConfiguration().setClientIds(registrationClientId);

```

4.3.2.2 PIL Configuration

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

API :

Present in **RegistrationDynamicConfiguration** class.

```

/**
 * Get PIL Configuration
 *
 * @return PILConfiguration
 */
public PILConfiguration getPilConfiguration() {}

/**
 * Set PIL Configuration
 *
 * @param pilConfiguration
 */
public void setPilConfiguration(PILConfiguration pilConfiguration) {}

```

Present in **PILConfiguration** class

```

/**
 * Set Microsite Id
 *
 * @param micrositeId String

```

```

*/
public void setMicrositeId(String micrositeId) {}

/**
 * Set Registration Environment
 *
 * @param registrationEnvironment String
 */
public void setRegistrationEnvironment(final String registrationEnvironment) {}

/**
 * Set Registration Environment
 *
 * @param registrationEnvironment
 */
public void setRegistrationEnvironment(final Configuration registrationEnvironment) {}

public void setCampaignID(String campaignID) {}

```

Example:

```

RegistrationDynamicConfiguration.getInstance().getPilConfiguration().setCampaignID("Campain ID");
RegistrationDynamicConfiguration.getInstance().getPilConfiguration().setMicrositeId("Microsite ID");
RegistrationDynamicConfiguration.getInstance().getPilConfiguration().setRegistrationEnvironment(Configuration);

```

4.3.2.3 Flow

API :

Present in **RegistrationDynamicConfiguration** class.

```

/**
 * Get Flow
 *
 * @return Flow
 */
public Flow getFlow() {}

/**
 * Set Flow
 *
 * @param flow
 */
public void setFlow(Flow flow) {}

```

Present in **Flow** class

```

/**
 * Status of email verification required
 *
 * @return boolean
 */
public Boolean isEmailVerificationRequired() {}

/**
 * Set Email verification status
 *
 * @param emailVerificationRequired
 */

```

```

public void setEmailVerificationRequired(Boolean emailVerificationRequired) {}

/**
 * Status of terms and condition acceptance
 *
 * @return boolean
 */
public Boolean isTermsAndConditionsAcceptanceRequired() {}

/**
 * Set terms and condition acceptance required or no
 *
 * @param termsAndConditionsAcceptanceRequired
 */
public void setTermsAndConditionsAcceptanceRequired(Boolean
termsAndConditionsAcceptanceRequired) {}

/**
 * Get min age limit map
 *
 * @return Map of Country and Age restrictions
 */
public HashMap<String, String> getMinAgeLimit() {}

/**
 * Set Min age Limit
 *
 * @param minAgeLimit Map with Key value pair of country code and min age
 */
public void setMinAgeLimit(HashMap<String, String> minAgeLimit) {}

/**
 * Get minimum age for country
 *
 * @param countryCode Country code
 * @return integer value of min age if mapping available else 0
 */
public int getMinAgeLimitByCountry(String countryCode) {}

```

Example :

```

RegistrationDynamicConfiguration.getInstance().getFlow().setEmailVerificationRequired(false);
RegistrationDynamicConfiguration.getInstance().getFlow().setTermsAndConditionsAcceptanceRequired(false);

HashMap<String, String> ageMap = new HashMap<>();
map.put("Country code", "Age Limit");

RegistrationDynamicConfiguration.getInstance().getFlow().setMinAgeLimit(ageMap);

```

4.3.2.4 HSDP Configuration (Optional)

API :

Present in **RegistrationDynamicConfiguration** class.

```

/**
 * Get HSDP configuration
 *
 * @return HSDPConfiguration

```



```

*/
public HSDPConfiguration getHsdPConfiguration() {}

/**
 * Set HSDP configuration
 *
 * @param hsdPConfiguration
 */
public void setHsdPConfiguration(HSDPConfiguration hsdPConfiguration) {}

```

Present in **HSDPConfiguration** class

```

/**
 * Get All HSDP information with all configuration
 *
 * @return Map of Configuration and its informantion.
 */
public HashMap<Configuration, HSDPInfo> getHsdPInfos() {}

/**
 * Set HSDP Informations for different Configurations
 *
 * @param hsdPInfos Map of Configuration and its informantion.
 */
public void setHsdPInfos(HashMap<Configuration, HSDPInfo> hsdPInfos) {}

/**
 * Get HSDP information for specified configuration
 *
 * @param configuration
 * @return HSDPInfo Object
 */
public HSDPInfo getHSDPInfo(Configuration configuration) {}

/**
 * Set HSDP Information for specific Configuration
 *
 * @param configuration
 * @param hsdPInfo HSDP Information
 */
public void setHSDPInfo(Configuration configuration, HSDPInfo hsdPInfo) {}

```

Present in **HSDPInfo** class

```

/**
 * Get Shared Id
 *
 * @return String shared id
 */
public String getSharedId() {}

/**
 * Set shared id
 *
 * @param sharedId String
 */
public void setSharedId(String sharedId) {}

/**
 * Get secret id
 *
 * @return String

```

```

    */
    public String getSecretId() {}

    /**
     * Set secrete id
     *
     * @param secretId String
     */
    public void setSecretId(String secretId) {}

    /**
     * Get base URL
     *
     * @return String
     */
    public String getBaseUrl() {}

    /**
     * Set base URL
     *
     * @param baseUrl
     */
    public void setBaseUrl(String baseUrl) {}

    /**
     * Get Application name
     *
     * @return String Application
     */
    public String getApplicationName() {}

    /**
     * Set application name
     *
     * @param applicationName
     */
    public void setApplicationName(String applicationName) {}

```

Example

```

HSDPConfiguration hsdpConfiguration = new HSDPConfiguration();

HSDPInfo hsdplInfo = new HSDPInfo();
hsdplInfo.setApplicationName("Appname");
hsdplInfo.setBaseUrl("url");
hsdplInfo.setSecretId("secrete id");
hsdplInfo.setSharedId("shared id");

hsdpConfiguration.setHSDPInfo(Configuration.EVALUATION, hsdplInfo);

RegistrationDynamicConfiguration.getInstance().setHsdpConfiguration(hsdpConfiguration);

```

4.3.2.5 SigninProviders

API :

Present in **RegistrationDynamicConfiguration** class.

```

/**
 * Get Sign in providers
 *
 * @return SigninProviders
 */
public SigninProviders getSignInProviders() {}

/**
 * Set Social providers
 *
 * @param socialProviders
 */
public void setSignInProviders(SigninProviders socialProviders) {}

```

Present in **SigninProviders** class

```

/**
 * Set providers in Hash map first param is country code and other is enabled providers list in Hash map
 *
 * @param providers
 */
public void setProviders(HashMap<String, ArrayList<String>> providers) {}

/**
 * Get providers
 *
 * @param countryCode Country code
 * @return List of providers
 */
public ArrayList<String> getProvidersForCountry(String countryCode) {}

```

Example :

```

//Configure Signin Providers
HashMap<String, ArrayList<String>> providers = new HashMap<String, ArrayList<String>>();
ArrayList<String> values1 = new ArrayList<String>();
ArrayList<String> values2 = new ArrayList<String>();
ArrayList<String> values3 = new ArrayList<String>();
values1.add("googleplus");
values1.add("facebook");

values2.add("googleplus");
values2.add("facebook");

values3.add("googleplus");
values3.add("facebook");

providers.put("NL", values1);
providers.put("US", values2);
providers.put("DEFAULT", values3);
RegistrationDynamicConfiguration.getInstance().getSignInProviders().setProviders(providers);

```

4.3.3 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.

5. Janrain Initialization

Janrain Initialization should be done from onCreate() of Application class of vertical projects along with Tagging setting and UI priority

5.1 UI Priority function

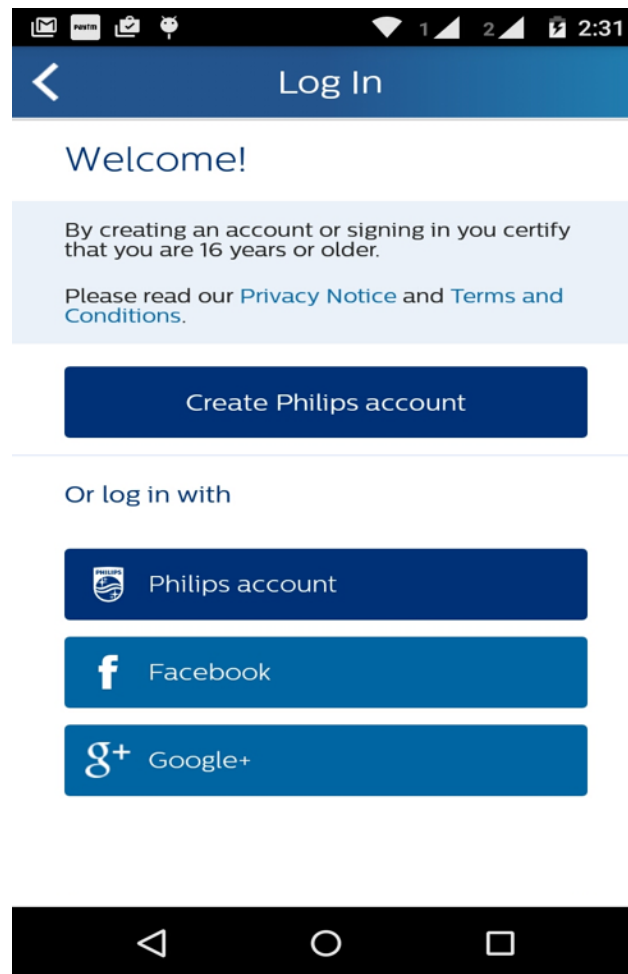
UI priority function enables app to launch Create account at TOP or BOTTOM. Needs to set during initialization. By default Create account is TOP.

Possible selection is out of

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

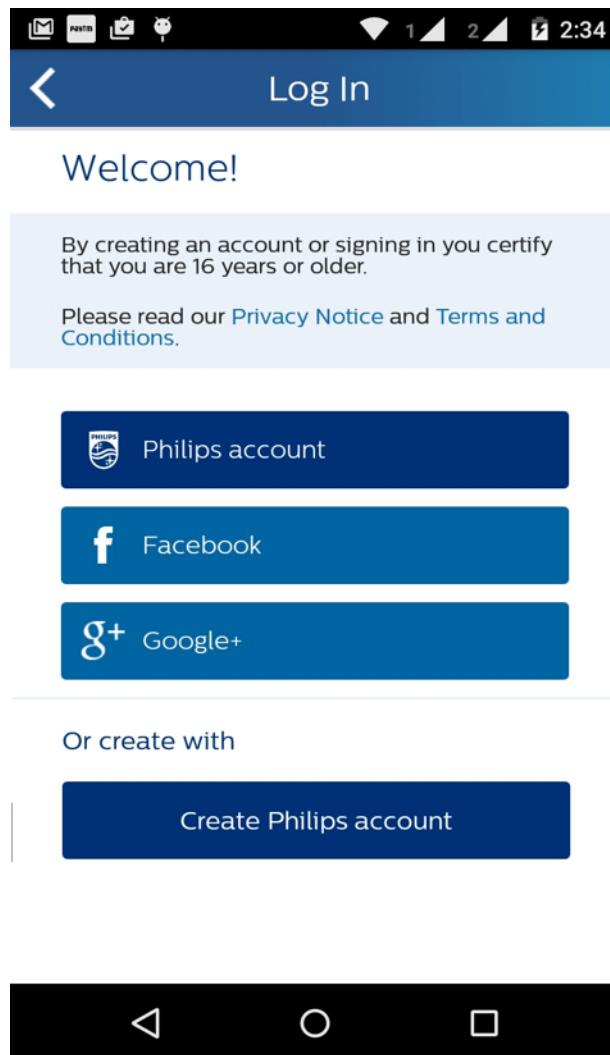
5.1.1 Create account on Top

```
//REGISTRATION UI PRIORITY(CREATE OR SIGNIN  
RegistrationConfiguration.getInstance().setPrioritisedFunction(RegistrationFun  
ction.Registration);
```



5.1.2 Sign In On top

```
RegistrationConfiguration.getInstance().setPrioritisedFunction(RegistrationFunction.SignIn);
```



5.2 STANDARD and HSDP configuration initialization

```
public class XXXXX extends Application {

    @Override
    public void onCreate() {
        super.onCreate();
        //TAGGING
        Tagging.enableAppTagging(true);
        Tagging.setTrackingIdentifier("integratingApplicationAppsId");
        Tagging.setLaunchingPageName("demoapp:home");
        //REGISTRATION UI PRIORITY(CREATE OR SIGNIN
```

```

RegistrationConfiguration.getInstance().setPrioritisedFunction(RegistrationFunction.Registration);
//REGISTRATION INITIALIZATION
RegistrationHelper.getInstance().initializeRegistrationSettings(this,
    Locale.getDefault());
//TAGGING INITIALIZATION
Tagging.init(RegistrationHelper.getInstance().getLocale(), this);
}
}

```

5.3 COPPA configuration initialization

```

public class XXXXX extends Application {

    private RegistrationHelper mRegistrationHelper;

    @Override
    public void onCreate() {
        super.onCreate();
        //TAGGING
        Tagging.enableAppTagging(true);
        Tagging.setTrackingIdentifier("integratingApplicationAppsId");
        Tagging.setLaunchingPageName("demoapp:home");
        //REGISTRATION UI PRIORITY(CREATE OR SIGNIN
        RegistrationConfiguration.getInstance().setPrioritisedFunction(RegistrationFunction.Registration);
        //COPPA FLOW NEEDS TO SET WITH TRUE VALUE
        RegistrationHelper.getInstance().setCoppaFlow(true);

        //REGISTRATION INITIALIZATION
        RegistrationHelper.getInstance().initializeRegistrationSettings(this,
            Locale.getDefault());
        //TAGGING INITIALIZATION
        Tagging.init(RegistrationHelper.getInstance().getLocale(), this);
    }
}

```

5.4 Custom Language in Registration UI

There is no additional API provided in Registration if App needs specific language UI to be displayed. It will be part of App initialization.

Note : Need to initialize once again to change the language

6. Launching Registration component

- As Activity
- As Fragment

6.1 Activity

6.1.1 Supported orientation

Supported orientation integer constant

- `ActivityInfo.SCREEN_ORIENTATION_PORTRAIT`
- `ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE`
- `ActivityInfo.SCREEN_ORIENTATION_SENSOR_PORTRAIT`
- `ActivityInfo.SCREEN_ORIENTATION_SENSOR_LANDSCAPE`

6.1.2 Launching default mode

Support in all orientation

Should call `RegistrationLaunchHelper.launchDefaultRegistrationActivity(this);` which takes Context as param

6.1.3 Launching default mode with fixed orientation

Should call

`RegistrationLaunchHelper.launchDefaultRegistrationActivityWithFixedOrientation(this,orientationInetger);` which takes Context and Orientation integer as params

6.1.4 Launching with context of Account settings

6.1.4.1 *Launching with account settings in default mode*

Support all orientation should call

`RegistrationLaunchHelper.launchRegistrationActivityWithAccountSettings(this);` which takes Context as param

6.1.4.2 *Launching with account settings with fixed orientation*

Should call `RegistrationLaunchHelper.`

`launchRegistrationActivityWithFixedOrientationWithAccountSettings(this,orientationInetger);`

which takes Context and Orientation integer as params

6.1.4.3 *Launching without account settings with fixed orientation*

Should call `RegistrationLaunchHelper.launchRegistrationActivityWithFixedOrientationWithoutAccountSettings(this,orientationInetger);`
which takes Context and Orientation integer as params

6.2 Fragment

- Launching as fragment
- Back handling
- Updating title

6.2.1 Launching as fragment

Should write and call following method to launch as fragment which take container Id ,
FragmentActivity ,AccountSettings screen as params.

```
launchRegistrationFragment(R.id.fl_reg_fragment_container, this);

/**
 * Launch registration fragment
 */
private void launchRegistrationFragment(int container, FragmentActivity
fragmentActivity,boolean isAccountSettings ) {
    try {
        FragmentManager mFragmentManager =
fragmentActivity.getSupportFragmentManager();
        RegistrationFragment registrationFragment = new
RegistrationFragment();
        Bundle bundle = new Bundle();

        bundle.putBoolean(RegConstants.ACCOUNT_SETTINGS,isAccountSettings);
        registrationFragment.setArguments(bundle);
        registrationFragment.setOnUpdateTitleListener(this);
        FragmentTransaction fragmentTransaction =
mFragmentManager.beginTransaction();
        fragmentTransaction.replace(container, registrationFragment,
RegConstants.REGISTRATION_FRAGMENT_TAG);
        fragmentTransaction.commitAllowingStateLoss();
    } catch (IllegalStateException e) {
        RLog.e(RLog.EXCEPTION,
            "RegistrationActivity :FragmentTransaction Exception
occured in addFragment :"+
            e.getMessage());
    }
}
```

6.2.2 Backstack handling

Should follow following code to handle back stack

```
@Override
public void onBackPressed() {
    if(!RegistrationLaunchHelper.isBackEventConsumedByRegistration(this)) {
        // not consumed vertical code goes here // actual code
        super.onBackPressed();
    }
}
```

6.2.3 Updating Title

Vertical projects should implement **RegistrationTitleBarListener** and should be referenced during launching of Registration fragment. Which override following methods

- **updateRegistrationTitle** : Triggered when fragment count is only 1 and Vertical app should write logic to update title and showing hamburger menu
- **updateRegistrationTitleWithBack**: Triggered when fragment count is greater than 1 and vertical app should update title and show back button
- **updateRegistrationTitleWithOutBack** : Triggered when welcome screen comes and need to do same activity as above two methods .

Ex:

```
@Override
public void updateRegistrationTitle(int titleResourceID) {
    // Update title and show hamburger
    //ivBack.setVisibility(View.INVISIBLE);
    ivBack.setVisibility(View.VISIBLE);
    TextView tvTitle = ((TextView)
    findViewById(R.id.tv_reg_header_title));
    tvTitle.setText(getString(titleResourceID));
}

@Override
public void updateRegistrationTitleWithBack(int titleResourceID) {
    // update title and show back
    ivBack.setVisibility(View.VISIBLE);
    TextView tvTitle = ((TextView)
    findViewById(R.id.tv_reg_header_title));
    tvTitle.setText(getString(titleResourceID));
}

@Override
public void updateRegistrationTitleWithOutBack(int titleResourceID) {
    // update title and show back
    ivBack.setVisibility(View.INVISIBLE);
    TextView tvTitle = ((TextView)
    findViewById(R.id.tv_reg_header_title));
    tvTitle.setText(getString(titleResourceID));
}
```

7. 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 `UserRegistrationListener` 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
}

@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 onCreate of implementing Activity

```
RegistrationHelper.getInstance().registerUserRegistrationListener(this)
```

In onDestroy of calling Activity.

```
RegistrationHelper.getInstance().unRegisterUserRegistrationListener(this)
```

8. APP tagging

App tagging is supported in Registration Android and tested against

Adobe JAR version : [v4.5.1 - Android](#)

ADBMobileConfig.json :

```
{
  "version" : "1.0",
  "acquisition": {
    "server": "c00.adobe.com"
  },
  "analytics" : {
    "referrerTimeout": 5,
    "rsids" : "philipsmobileappsdev",
    "server" : "philips.112.2o7.net",
    "charset" : "UTF-8",
    "ssl" : false,
    "offlineEnabled" : true,
    "lifecycleTimeout" : 30,
    "batchLimit" : 0,
    "privacyDefault" : "optedin",
    "poi" : [
    ]
  },
  "target" : {
    "clientCode" : "amsdk",
    "timeout" : 5
  },
  "audienceManager" : {
    "server" : ""
  }
}
```

Tagging has following params to be set during initialization

```
public class RegistrationApplication extends Application {

    private RegistrationHelper mRegistrationHelper;

    @Override
    public void onCreate() {
        super.onCreate();
        //Tagging
        Config.setContext(getApplicationContext());

        //Janrain Initialization
        mRegistrationHelper = RegistrationHelper.getInstance();

        //Params context and locale should be passed from application
        //Coppa flow should be set to true
        mRegistrationHelper.setCoppaFlow(false);

        //Tagging specific
        //Enabling or disabling registration tagging
        AppTagging.enableAppTagging(true);
    }
}
```

```

//Mandatory to set
AppTagging.setTrackingIdentifier("IntegratingApplicationAppId");

//Optional may set or may not set
AppTagging.setLaunchingPageName("RegistrationLaunchingPageName");

mRegistrationHelper.initializeRegistrationSettings(this,
    Locale.getDefault());
}
}

```

9. COPPA specific API's

There are 3 API which can be used in COPPA specific flow for fulfilling need.

```
1. public CoppaStatus getCoppaEmailConsentStatus() {}
```

This API provides current consent status which can be any one of following values

- kDICOPPAConsentPending,
- kDICOPPAConsentNotGiven,
- kDICOPPAConsentGiven,
- kDICOPPAConfirmationPending,
- kDICOPPAConfirmationNotGiven,
- kDICOPPAConfirmationGiven;

```
2. public void fetchCoppaEmailConsentStatus(final Context context, final
FetchCoppaEmailConsentStatusHandler handler) {}
```

This API provide email consent from server and give consent status in callback.

Params :

Context : Application context.

FetchCoppaEmailConsentStatusHandler :

Call back handler which has following methods which is triggered on completion of fetching activity

```
void didCoppaStatusFetchingSuccess(CoppaStatus var1);
```

```
void didCoppaStatusFetchingFailedWithError(CaptureAPIError var1);
```

```
3. public void resendCoppaEmailConsentForUserEmail(final String email, final
ResendCoppaEmailConsentHandler resendCoppaEmailConsentHandler) {}
```

This API provide resending email capability

Params :

Email: Email id

ResendCoppaEmailConsentHandler : Callback handler

```
void didResendCoppaEmailConsentSuccess();
```

```
void didResendCoppaEmailConsentFailedWithError(CoppaResendError var1);
```

10. HSDP specific API's

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

10.1 HSDP UUID

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

10.2 HSDP Access Token

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

11. Library version's

Use following methods to get respective versions of registration

11.1 Janrain Library

```
Jump.getJumpVersion()
```

11.2 Locale Match Library

```
PILLocaleManager.getLacaleMatchVersion()
```

11.3 Registration Library

```
RegistrationHelper.getRegistrationApiVersion()
```

12. Supporting apps with Over 65K Methods

This is special case if app which has more than 65K methods do follow below link for reference .

<https://developer.android.com/tools/building/multidex.html>

According to this do following changes in gradle and application class

In Gradle file:

```
android {
    compileSdkVersion 21
    buildToolsVersion "21.1.0"

    defaultConfig {
        ...
        minSdkVersion 14
        targetSdkVersion 21
        ...
    }
    ...
}

// Enabling multidex support.
multiDexEnabled true

dependencies {
    compile 'com.android.support:multidex:1.0.1'
}
```

In Application Class:

```
@Override
public void onCreate() {
    MultiDex.install(this);
    Super.onCreate();
}
```

13. Proguard rules for Registration API component

As per Android Documentations mentioned in link <http://developer.android.com/tools/help/proguard.html> code will be behave unexpected way due to arbitrary rules .So following are proguard rules one which needs to be added in vertical apps proguard file to work registration as desired.

Note : Here all the configuration is mentioned pick one is needed as per your project flow

13.1 General

```
##General and network
-keep public class javax.net.ssl.**
-keepclassmembers public class javax.net.ssl.** {*; }
-keepclassmembers public class org.apache.http.** {*; }
-dontwarn org.apache.**
-keep class org.apache.http.** { *; }
-keep class android.net.http.** { *; }
```

```
-renamesourcefileattribute SourceFile
-keepattributes SourceFile,LineNumberTable
-keepattributes InnerClasses,Exceptions
Tagging

-keep public class com.adobe.mobile.** {*; }
-keep public class com.philips.cdp.tagging.** {*; }
```

13.2 Jump

```
#Janrain lib
-keep public class com.janrain.android.** {*; }
-keep class com.janrain.android.Jump$* {*; }
-keep class com.philips.cdp.registration.User$* {*; }
-keep class com.janrain.android.capture.Capture$* {*; }

-keep public class com.philips.cdp.security.SecureStorage {
    public static void init(android.content.Context);
}

-keep public class com.philips.cdp.security.SecureStorage {
    public static java.lang.String objectToString(java.io.Serializable);
}

-keep public class com.philips.cdp.security.SecureStorage {
    public static java.lang.Object stringToObject(java.lang.String);
}

-keep public class com.philips.cdp.security.SecureStorage {
    public static void migrateUserData(java.lang.String);
}

-keep public class com.philips.cdp.security.SecureStorage {
    public static byte[] encrypt(java.lang.String);
}

-keep public class com.philips.cdp.security.SecureStorage {
    public static byte[] decrypt(byte[]);
}

-keep public class com.philips.cdp.security.SecureStorage {
    public static void generateSecretKey();
}

-keepclasseswithmembernames public class com.janrain.android.** {*; }
-keepclasseswithmembernames public class com.janrain.android.Jump {*; }
-keepclasseswithmembernames public class com.janrain.android.JumpConfig
{*; }
-keepclasseswithmembernames public class com.janrain.android.TradSignInUi
{*; }
```


13.3 Localematch

```
-keep public class com.philips.cdp.localematch.** {*;}
```

13.4 Registration API

```
-keep class com.philips.cdp.registration.** {*;}  
-dontwarn com.philips.cdp.registration.**
```

13.5 Network APIs

```
-dontwarn org.apache.**  
-keep class org.apache.http.** { *; }  
-keep class android.net.http.** { *; }
```

13.6 HSDP

```
-keep class com.philips.dhpcclient.** {*;}  
-keep class com.fasterxml.jackson.annotation.** {*;}  
-keep class com.fasterxml.jackson.core.** {*;}  
-keep class com.fasterxml.jackson.databind.** {*;}
```

13.7 Other platform specific

```
#GSM  
-keep class com.google.android.gms.* { public *; }  
-dontwarn com.google.android.gms.**  
-dontwarn org.w3c.dom.bootstrap.DOMImplementationRegistry  
  
#webkit  
-keep class android.net.http.SslError  
-keep class android.webkit.WebViewClient  
  
-dontwarn android.webkit.WebView  
-dontwarn android.net.http.SslError  
-dontwarn android.webkit.WebViewClient  
  
#notification  
-dontwarn android.app.Notification
```

13.8 Hockey SDK

```
-keepclassmembers class net.hockeyapp.android.UpdateFragment {*;}
```

13.9 Complete for configuration for Registration API

```
##Registration API specific

##General and network
-keep public class javax.net.ssl.**
-keepclassmembers public class javax.net.ssl.** {*; }
-keepclassmembers public class org.apache.http.** {*; }
-dontwarn org.apache.**
-keep class org.apache.http.** { *; }
-keep class android.net.http.** { *; }
-renamesourcefileattribute SourceFile
-keepattributes SourceFile,LineNumberTable
-keepattributes InnerClasses,Exceptions

#Hockey app and enabling exception catching
-keepclassmembers class net.hockeyapp.android.UpdateFragment {*; }

#Tagging lib and jar
-keep public class com.adobe.mobile.** {*; }
-keep public class com.philips.cdp.tagging.** {*; }

#Janrain lib
-keep public class com.janrain.android.** {*; }
-keep class com.janrain.android.Jump$* {*; }
-keep class com.philips.cdp.registration.User$* {*; }
-keep class com.janrain.android.capture.Capture$* {*; }

-keep public class com.philips.cdp.security.SecureStorage {
    public static void init(android.content.Context);
}

-keep public class com.philips.cdp.security.SecureStorage {
    public static java.lang.String objectToString(java.io.Serializable);
}

-keep public class com.philips.cdp.security.SecureStorage {
    public static java.lang.Object stringToObject(java.lang.String);
}

-keep public class com.philips.cdp.security.SecureStorage {
    public static void migrateUserData(java.lang.String);
}

-keep public class com.philips.cdp.security.SecureStorage {
    public static byte[] encrypt(java.lang.String);
}
```

```

-keep public class com.philips.cdp.security.SecureStorage {
    public static byte[] decrypt(byte[]);
}

-keep public class com.philips.cdp.security.SecureStorage {
    public static void generateSecretKey();
}

-keepclasseswithmembernames public class com.janrain.android.** {*; }
-keepclasseswithmembernames public class com.janrain.android.Jump {*; }
-keepclasseswithmembernames public class com.janrain.android.JumpConfig
{*; }
-keepclasseswithmembernames public class com.janrain.android.TradSignInUi
{*; }

#Locale match
-keep public class com.philips.cdp.localematch.** {*; }

#Registration API
-keep class com.philips.cdp.registration.** {*; }
-dontwarn com.philips.cdp.registration.**

#HSDP Lib
-keep class com.philips.dhpcclient.** {*; }
-keep class com.fasterxml.jackson.annotation.** {*; }
-keep class com.fasterxml.jackson.core.** {*; }
-keep class com.fasterxml.jackson.databind.** {*; }

#GSM
-keep class com.google.android.gms.* { public *; }
-dontwarn com.google.android.gms.**
-dontwarn org.w3c.dom.bootstrap.DOMImplementationRegistry

#webkit
-keep class android.net.http.SslError
-keep class android.webkit.WebViewClient

-dontwarn android.webkit.WebView
-dontwarn android.net.http.SslError
-dontwarn android.webkit.WebViewClient

#notification
-dontwarn android.app.Notification

```

14. Demo apps

DemoApps are available with package

