Android – Using the SDK

selligent MARKETING CLOUD

Product Manual

Current release: 5/18/2018

1 Table of content

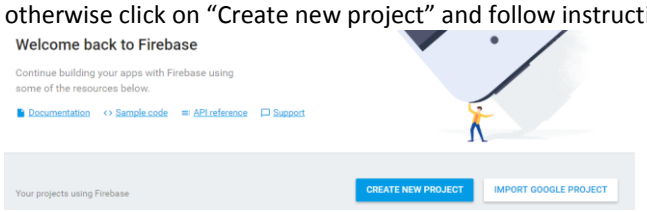
2 Intro

The purpose of this document is to detail how to install the SDK into your app and how to easily start using it.

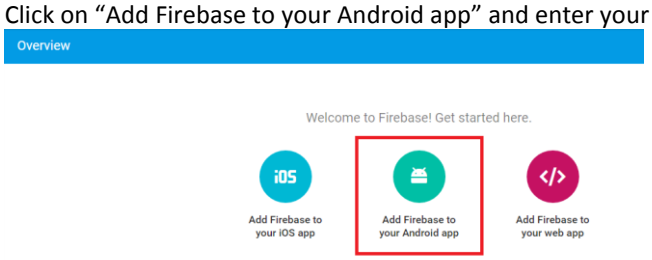
- For more detailed implementation of the SDK please refer to the "Android - MobileSDK Reference.pdf" document.
- For an example of implementation check the "AndroidSDKTemplate" project.

3 Creating a Google application

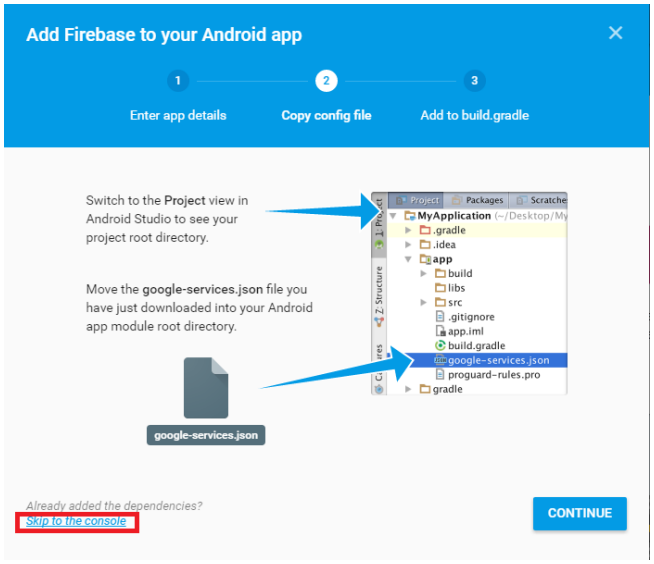
- If you already use push notifications (either yourself or through a third party library), simply re-use the Server Key and Sender ID you already have.
- If you don't have any yet, go to https://console.firebase.google.com/ and sign in with a Google account.
- If you already have a project on Google Developer Console, click on "Import Google Project", otherwise click on "Create new project" and follow instructions.



- Click on "Add Firebase to your Android app" and enter your package name.

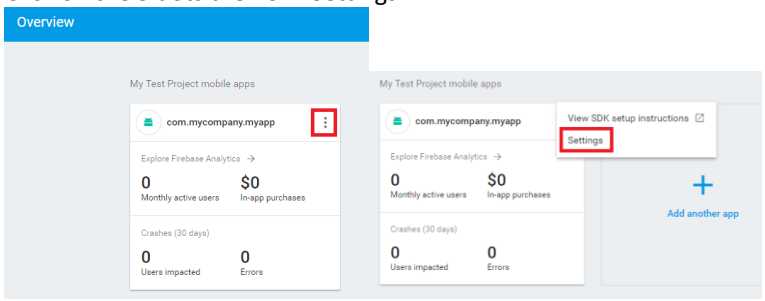


- If your version of Google Play Services is under 9.0 or you don't want to use the new way of registering to cloud messaging, ignore the json file and directly click on "Skip to the console".
  From April 2018, the old GCM way of getting a token has been deprecated and will be removed in April 2019. Therefore, it is highly recommended to use the json file.

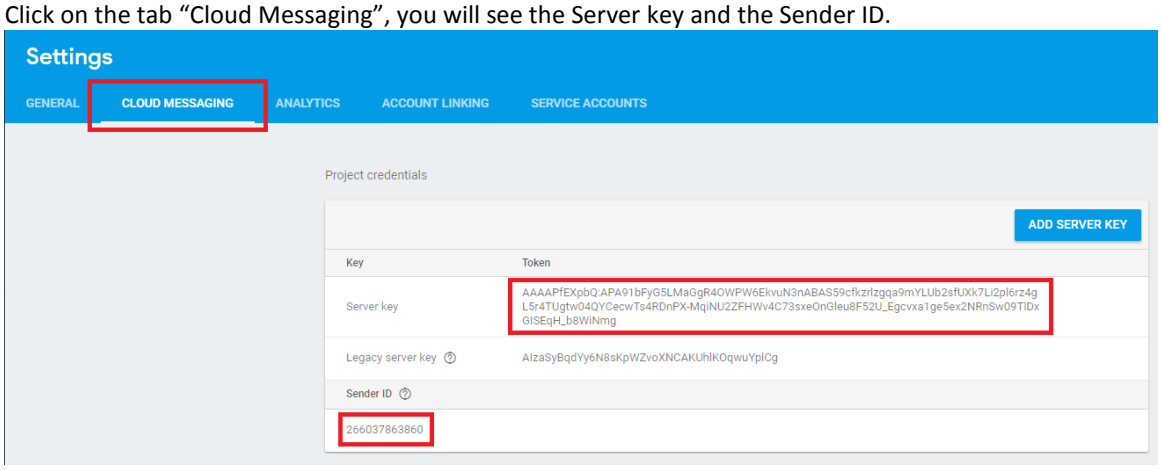


Otherwise, save the JSON file, put it in your project and update your two gradle files following the instructions on the screen.

- Click on the 3 dots then on “Settings”



- Click on the tab “Cloud Messaging”, you will see the Server key and the Sender ID.



- Note the Server key, you will have to give it to the Selligent platform. If you chose not to use the JSON file, note the Sender ID, it will be needed by the SDK when calling the start method.

4 Including the SDK in your project

4.1 SDK library

You can either add a dependency to our library (recommended) or create a new module containing the aar file.

4.1.1 Add the Selligent dependancy from JCenter

First, make sure that you have the following line in the build.gradle file located at the root of your project:

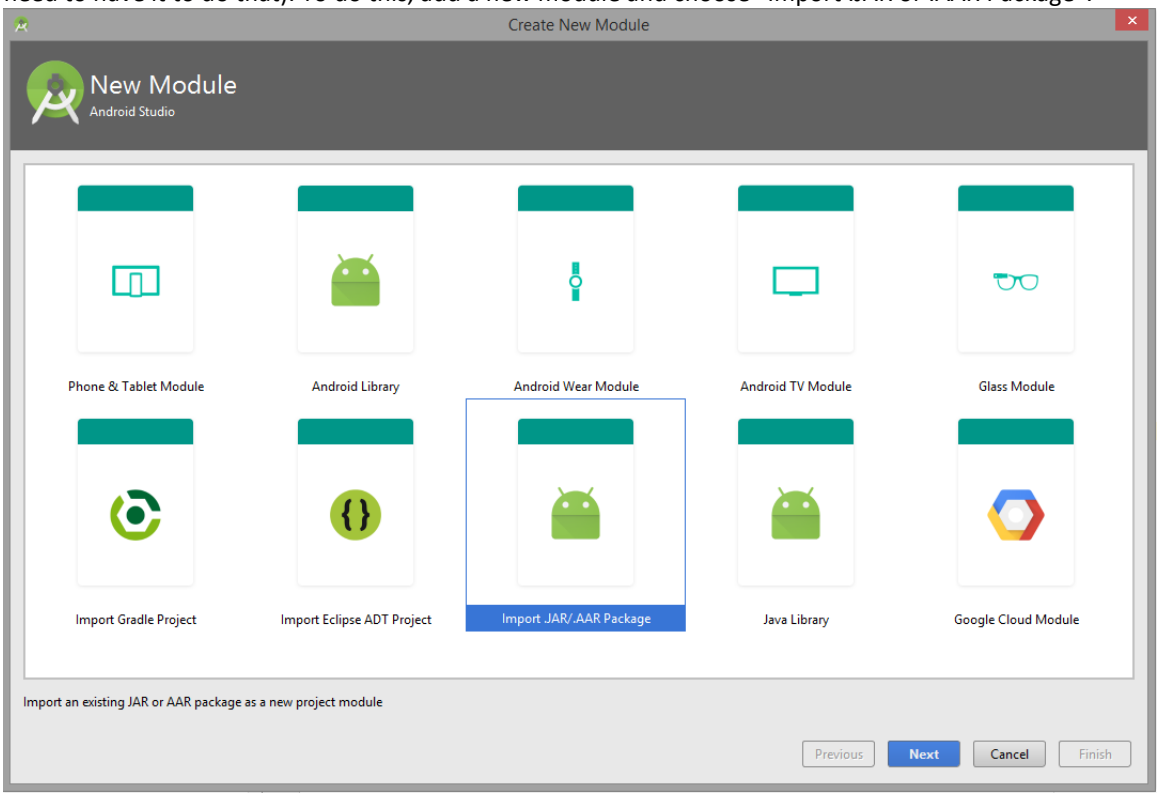
```
repositories {
    jcenter()
}
```

Then, in the build.gradle file under the app module, add the following line in the list of dependencies :

```
compile 'com.selligent.sdk:selligent_mobile_sdk:1.7.0'
```

4.1.2 Import the Selligent library

If you do not want to add the dependency, you can create a new module that will contain the aar file (you need to have it to do that). To do this, add a new module and choose “Import .JAR or .AAR Package”.



And select the file.
Once it is done, add a dependency to this new module in your app, then synchronize and build the project.

4.2 Other libraries

You need to add some external dependencies in your app gradle file.

- Firebase messaging

  ```
  com.google.firebase:firebase-messaging
  ```

- The version of Firebase must be at least 10.2.1 in order to be compatible with Android O. It is recommended to use the latest one available.
- If you choose to use the JSON file given to you by Firebase (cf. Creating a Google application) to retrieve the cloud messaging token and you followed the instructions to modify the gradle files, you should have the following:
  - In the build.gradle file at project level:
    - ```
      dependencies {
              classpath 'com.google.gms:google-services:3.2.0'
      }
      ```
  - In the build.gradle file at app level, at the bottom:
    - ```
      apply plugin: 'com.google.gms.google-services'
      ```
  - In this case, you don't need a dependency to play-services-gcm (you still need play-services-maps if you plan on sending Map type push).

- GSON

  ```
  com.google.code.gson:gson
  ```

- CardView

  ```
  com.android.support:cardview-v7
  ```

- MetadataXTractor
  If you plan on sending Push or In-App messages containing "Response" type buttons, you have to add the following third party library:

  ```
  com.drewnoakes:metadata-extractor
  ```

- PlotProject
  If you want to use geolocation, you will need a dependency to the PlotProject library:

  ```
  com.plotprojects:plot-android:2.4.0-beta
  ```

  For Gradle to find that dependency, you must add a reference to the Maven Plot repository. In the list of repositories in your top build.gradle file, add

  ```
  allprojects {
      repositories {
          …
          maven {
              url 'https://maven-repo.plotprojects.com'
          }
      }
  }
  ```

5 How to use the SDK

5.1 Starting the SDK

5.1.1 Extending Application

The SDK needs to be started in a class extending Application.
If you do not already have one, create a new class, for example “MyApplication” that will extend Application:

```
public class MyApplication extends Application
{
    @Override
    public void onCreate()
    {
        [setup the SDK here]
        super.onCreate();
    }
}
```

On the OnCreate event, setup the SDK (cf. Start).
In the AndroidManifest.xml file, add the following:

```
<application
    android:name=".MyApplication"
    …
```

5.1.2 Start

To start the SDK, in your class extending Application, use the following:

```
SMManager.getInstance().start(settings, this);
```

`this` is of course the instance of the class extending Application.
`settings` is an SMSettings object, proposing the following mandatory members:

WebServiceUrl must be the URL of the Selligent web service that will be called. It is given by Selligent.
GoogleApplicationId If you do not want your app to register to cloud messaging using the JSON file, then you must set this property with the Sender ID that you got on Firebase (cf. 3). Otherwise, you have to leave it null.
From April 2018, the old GCM way of getting a token has been deprecated and will be removed in April 2019. Therefore, it is highly recommended to use the JSON file.
ClientId is the public key allowing the connection to the web service.
PrivateKey is the private key allowing the connection to the web service.

Ex.:

```
SMSettings settings = new SMSettings();
settings.WebServiceUrl = "https://www.some.web.service.com";
settings.GoogleApplicationId = "1111111111"; //or null if you use the JSON file
settings.ClientId = "SomeClientId";
settings.PrivateKey = "SomePrivateKey";
SMManager.getInstance().start(settings, this);
```

5.1.2.1 Optional settings

There are optional settings on SMSettings:

```
public SMClearCache ClearCacheIntervalValue
```

You can set it to change the way the SDK manages the cache. It is recommended to leave it to its default value of Auto.

```
public SMInAppRefreshType InAppMessageRefreshType
```
Setting this value will enable the In-App messages. It tells how often the SDK must retrieve the In-App messages.

```
public SMInAppRefreshType InAppContentRefreshType
```
Setting this value will enable the In-App contents. It tells how often the SDK must retrieve the In-App contents.

```
public SMInAppRefreshType RegionRefreshType
```
This value will define how often the SDK must retrieve the regions.

```
public SMRemoteMessageDisplayType RemoteMessageDisplayType
```
Setting this value will enable/disable the automatic display of remote messages as they are received, when the app is in foreground.

- Automatic: the message will be displayed right away
- Notification: a notification will be created and the message will be displayed after clicking on it.
- None: nothing will be done, the app will have to manage the display (using SMManager.getInstance().displayLastReceivedRemotePushNotification() and SMManager.getInstance().getLastRemotePushNotification()).

```
public boolean ConfigureGeolocation
```
This will tell the SDK to initialize the geolocation capabilities. For it to work, you need to add a dependency to the PlotProject library (cf. Other libraries) and a setting file (cf. Geolocation).
Default value is false.

There are also some optional settings on SMManager:

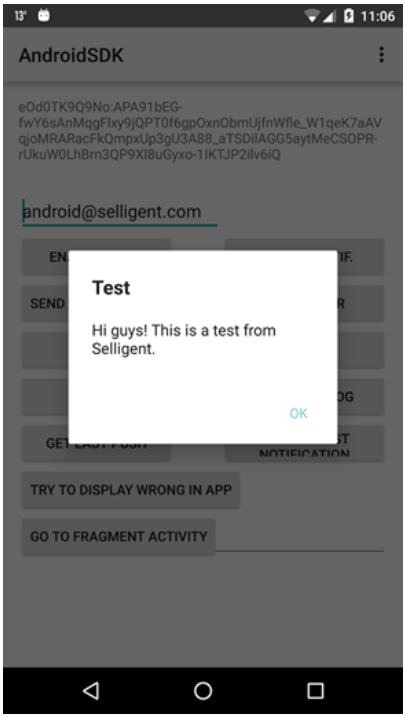
```
SMManager.DEBUG = true;
```
Setting this to true will add the SDK logs to the logcat (it is better to do that before calling the start method in order to see everything logged when the SDK starts).

5.2 Push notifications

Push notifications are messages sent from the server to a device.

When a push is received, if the app is in background or inactive, we create a notification. When you click on it, the app opens and the message is displayed. If the app is in foreground, it will either directly show the message, create a notification or do nothing, depending on the value of the setting RemoteMessageDisplayType (Automatic, Notification or None) given to the SMSettings object when calling the "start" method of SMManager.

Once the message is displayed, it is either in a dialog which, by default, looks like that:



or in a dedicated Activity, depending of the type of the push (Map, Image, HTML and URL open in a dedicated Activity).

5.2.1 Libraries

If you plan on sending push notifications containing "Response" type buttons, you have to add the following third party library in your dependencies:

com.drewnoakes:metadata-extractor:2.9.1

5.2.2 Permissions for Push notification

In order to be able to receive push notifications if you do not use the JSON file, the following permissions must be granted in the AndroidManifest.xml file:

```
<uses-permission android:name="com.google.android.c2dm.permission.RECEIVE" />
<uses-permission android:name="com.mycompany.myapp.permission.C2D_MESSAGE" />

<permission
    android:name="com.mycompany.myapp.permission.C2D_MESSAGE"
    android:protectionLevel="signature" />
```

com.mycompany.myapp must, of course, be replaced by your package.

If you use the JSON file method to register to cloud messaging, you do not need to add the previous lines.

If you plan on sending "Map" type push, you might want to add one of the following permissions in order for the user's location to be displayed on the map:

```
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
```
or
```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

Only one of the two is needed. Coarse location is less precise than fine location. Note that if you do not add any, the map will still be displayed, just not the user's location.

If you plan on sending push that include a "Response with picture" type button, you must add the following permission:

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

This is needed to access the pictures on the device and call the photo app. If it is not present in the manifest, the user will not be able do anything and a message "Not enough permission" will appear instead (this message can be changed, cf. Translations).

Warning: do NOT use "READ_EXTERNAL_STORAGE" instead of "WRITE_EXTERNAL_STORAGE", this would result in the app crashing when the user tries to access the camera or the gallery.

5.2.3 Registering

To register your device to Firebase Cloud Messaging (and therefore allow it to receive push notifications) and you do not want to use the JSON file method, add the following call to the onStart event of your main activity or your base activity if you have one:

```
@Override
protected void onStart()
{
   super.onStart();

   […]

    SMManager.getInstance().registerDevice(this);
}
```

This registration is asynchronous and is executed only once after the application started (it will be executed again the next time you start the app after killing it).

If you use the JSON file method, then this call is useless, everything will be done in the background.

5.2.4 Listening to and displaying the push notifications

In order to check if a notification was received and to display it, you have to add some code inside your activities (or, better, in any base Activity class you have).

First, you will need to add a member to your class with a type SMForegroundGcmBroadcastReceiver.

```
SMForegroundGcmBroadcastReceiver receiver;
```

Then, update the onCreate, onStart, onStop and onNewIntent events like this:

```
@Override
protected void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);

    […]

    SMManager.getInstance().checkAndDisplayMessage(getIntent(), this);
}

@Override
protected void onStart()
{
    super.onStart();

    […]
```

```
    if (receiver == null)
    {
        receiver = new SMForegroundGcmBroadcastReceiver(this);
    }
    registerReceiver(receiver, receiver.getIntentFilter());

    //If you have the following line in your Activity, it needs to be after registering
    //the receiver
    SMManager.getInstance().registerDevice(this);
}

@Override
protected void onStop()
{
    super.onStop();

    […]
    unregisterReceiver(receiver);
}

@Override
protected void onNewIntent(Intent intent)
{
    super.onNewIntent(intent);

    SMManager.getInstance().checkAndDisplayMessage(intent, this);
}
```

5.2.4.1 Extending SMBaseActivity

There is a class SMBaseActivity in the SDK that already does everything described in Registering and Listening and displaying the push notifications. You can make your activities extend it to avoid writing the code described in those points. Be aware though that SMBaseActivity extends AppCompatActivity, so it might not work for your project.

```
public class MainActivity extends SMBaseActivity
{
}
```

If you extend SMBaseActivity, nothing else needs to be done.

5.2.5 Customization

If the app is in background when a push is received, an icon will appear in the status bar and a notification will be added to the Notification drawer. Clicking on it will call a specific Activity which will display the message. Both can be customized.

5.2.5.1 Setting a specific icon

To customize that icon, call these methods after starting the SDK in your Application class:

```
SMManager.getInstance().setNotificationSmallIcon(R.drawable.some_icon);
```

This sets the small icon that will be used for the notifications in the notification bar. If not set, the default icon of the SDK will be used (the head of the Android robot).

```
SMManager.getInstance().setNotificationLargeIcon(R.drawable.some_large_icon);
```

This sets the large icon that will be used for the notifications. If not set, no large icon will be specified to Android, so the small one will be used.

5.2.5.2 Setting a specific Activity

By default, the Activity called to display the message of a push is NotificationActivity. If you want to keep it, in order to be able to go back to your application from it, it must be declared in the manifest as a child of an activity from your app (in the example, MainActivity).

```
<application
    …>
    …
    <activity
        android:name="com.selligent.sdk.NotificationActivity"
        android:parentActivityName=".MainActivity">
        <meta-data android:name="android.support.PARENT_ACTIVITY"
                   android:value=".MainActivity"></meta-data>
    </activity>
</application>
```

You can also set any Activity of your app to be called instead (in which case, no need to add the previous code to your manifest).

In your Application class, after starting the SDK, do this:

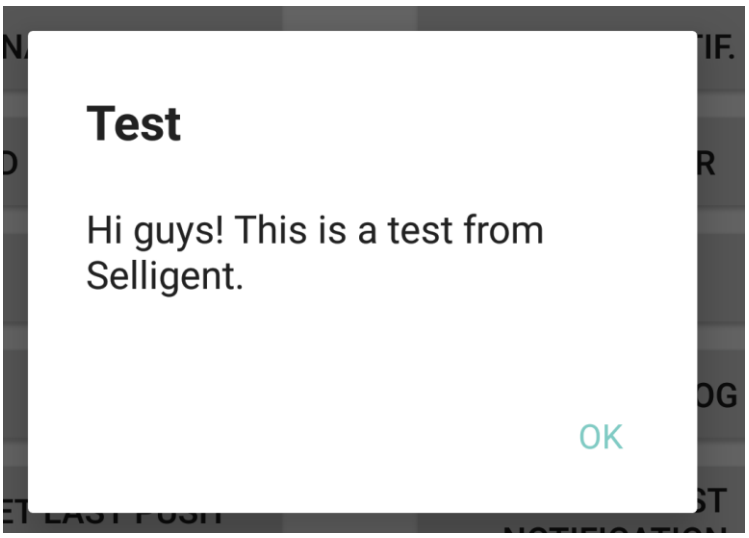
```
SMManager.NOTIFICATION_ACTIVITY = MyActivity.class;
```

If you used SMRemoteMessageDisplayType.Notification as value for RemoteMessageDisplayType (cf. Optional settings), you can also set this property in your base activity with the value returned by getClass(), that way the current activity will be the one called when clicking on the notification when the application is in foreground.

5.2.6 Design customization

5.2.6.1 Dialog

Some push messages are displayed as a dialog box which, by default, looks like this:



PS: the background and text colors will be the one defined in your theme.

This is a default layout made to have a "Material" look. It is entirely customizable. Its definition is in the file "styles.xml". Here is its content:

```
<style name="Selligent.Dialog.Container">
  <item name="android:paddingLeft">0dp</item>
  <item name="android:paddingRight">0dp</item>
  <item name="android:paddingTop">0dp</item>
  <item name="android:paddingBottom">0dp</item>
</style>
<style name="Selligent.Dialog.Title">
  <item name="android:singleLine">true</item>
  <item name="android:ellipsize">end</item>
  <item name="android:layout_marginLeft">24dp</item>
  <item name="android:layout_marginRight">24dp</item>
```

```
    <item name="android:layout_marginTop">24dp</item>
    <item name="android:layout_marginBottom">0dp</item>
    <item name="android:textSize">20sp</item>
    <item name="android:textColor">?android:attr/textColorPrimary</item>
    <item name="android:typeface">sans</item>
    <item name="android:textStyle">bold</item>
    <item name="android:shadowRadius">0</item>
    <item name="android:gravity">start</item>
</style>
<style name="Selligent.Dialog.UpperDivider">
    <item name="android:layout_height">0dp</item>
    <item name="android:visibility">gone</item>
</style>
<style name="Selligent.Dialog.BodyScrollView">
    <item name="android:clipToPadding">false</item>
    <item name="android:layout_marginTop">20dp</item>
    <item name="android:layout_marginLeft">24dp</item>
    <item name="android:layout_marginRight">24dp</item>
    <item name="android:layout_marginBottom">24dp</item>
</style>
<style name="Selligent.Dialog.Body">
    <item name="android:textSize">16sp</item>
    <item name="android:typeface">sans</item>
    <item name="android:textColor">?android:attr/textColorPrimary</item>
    <item name="android:maxLines">10</item>
</style>
<style name="Selligent.Dialog.LowerDivider">
    <item name="android:layout_height">0dp</item>
</style>
<style name="Selligent.Dialog.ButtonScrollView">
</style>
<style name="Selligent.Dialog.ButtonContainer">
    <item name="android:gravity">end</item>
    <item name="android:paddingLeft">8dp</item>
    <item name="android:paddingRight">8dp</item>
    <item name="android:paddingTop">8dp</item>
    <item name="android:paddingBottom">8dp</item>
</style>
<style name="Selligent.Dialog.ButtonRow">
    <item name="android:layout_width">wrap_content</item>
</style>
<style name="Selligent.Dialog.Button">
    <item name="android:layout_height">36dp</item>
    <item name="android:minWidth">64dp</item>
    <item name="android:paddingLeft">8dp</item>
    <item name="android:paddingRight">8dp</item>
    <item name="android:radius">2dp</item>
    <item name="android:focusable">true</item>
    <item name="android:clickable">true</item>
    <item name="android:gravity">center_vertical|center_horizontal</item>
    <item name="android:textSize">14sp</item>
    <item name="android:typeface">sans</item>
    <item name="android:textAllCaps">true</item>
    <item name="android:textColor">#ff80cbc4</item>
    <item name="android:background">?android:attr/selectableItemBackground</item>
</style>
```

To customize it, in your own file “styles.xml”, override the styles you want to modify. Note that you have to copy the whole content of those styles, not only the items you want to change, otherwise the others will be lost.

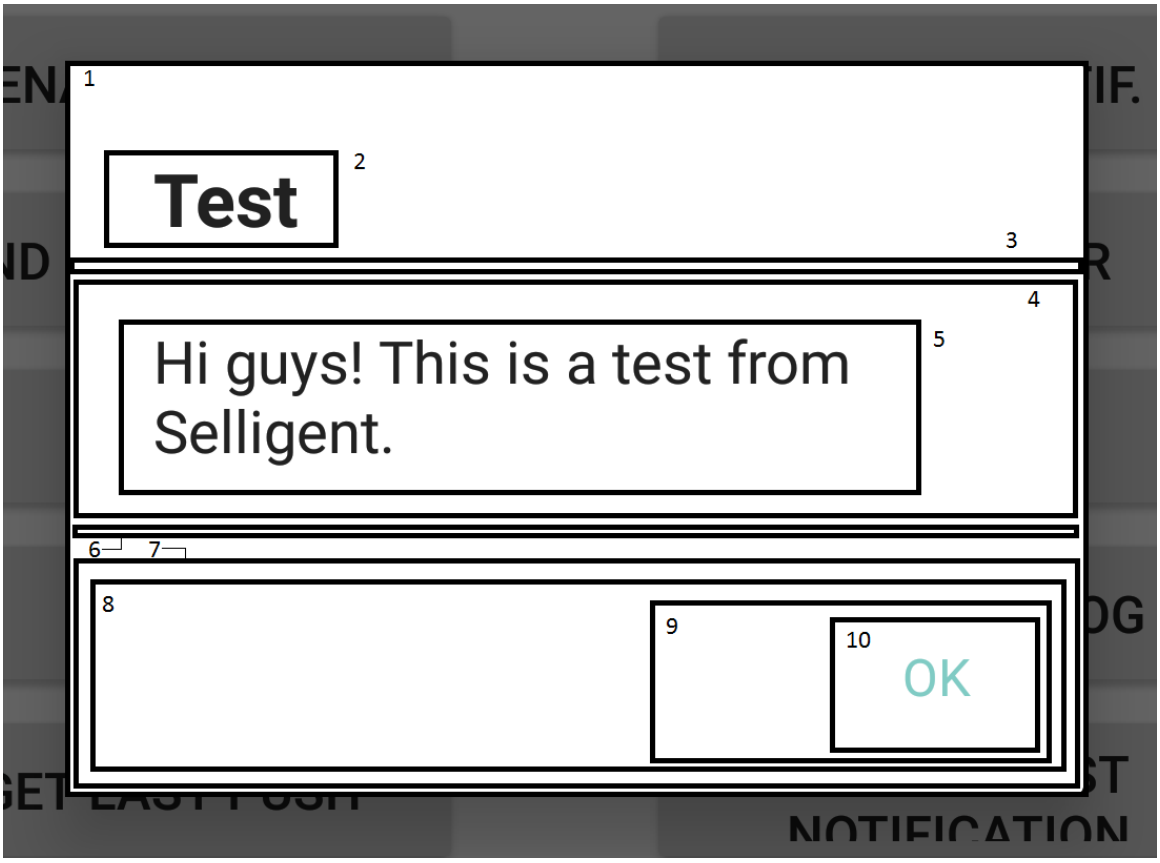
For example, if you simply want to change the text color of a button to red, you still have to add in your file the whole style:

```
<style name="Selligent.Dialog.Button">
    <item name="android:layout_height">36dp</item>
    <item name="android:minWidth">64dp</item>
    <item name="android:paddingLeft">8dp</item>
    <item name="android:paddingRight">8dp</item>
    <item name="android:radius">2dp</item>
```

```
    <item name="android:focusable">true</item>
    <item name="android:clickable">true</item>
    <item name="android:gravity">center_vertical|center_horizontal</item>
    <item name="android:textSize">14sp</item>
    <item name="android:typeface">sans</item>
    <item name="android:textAllCaps">true</item>
    <item name="android:textColor">#ff0000</item>
    <item name="android:background">?android:attr/selectableItemBackground</item>
</style>
```

The different styles are applied like this:

```
1  <style name="Selligent.Dialog.Container">
2  <style name="Selligent.Dialog.Title">
3  <style name="Selligent.Dialog.UpperDivider">
4  <style name="Selligent.Dialog.BodyScrollView">
5  <style name="Selligent.Dialog.Body">
6  <style name="Selligent.Dialog.LowerDivider">
7  <style name="Selligent.Dialog.ButtonScrollView">
8  <style name="Selligent.Dialog.ButtonContainer">
9  <style name="Selligent.Dialog.ButtonRow">
10 <style name="Selligent.Dialog.Button">
```



5.2.6.2 Activities

Some other type of messages (like Map, HTML, Response, etc.) are displayed in their own activity, not in a dialog. Those activities extend AppCompatActivity and, therefore, need an AppCompat theme. So, in order to avoid any crash when displaying them, we force our own AppCompat theme: Theme.SMTheme. It has for parent Theme.AppCompat.Light. You might want to override it to reflect your own layout.
To do so, simply define that theme in your app and use as parent the appropriate AppCompat theme.
Examples:

- If you use Theme.Holo.Light, define Theme.SMTheme like this (in styles.xml):

```
<style name="Theme.SMTheme" parent="Theme.AppCompat.Light"></style>
```

- If you use a customized Holo theme whose parent is Theme.Holo.Light, do this

```
<style name="Theme.SMTheme" parent="Theme.AppCompat.Light">
    <item name="colorPrimaryDark">@color/yourPrimaryDarkColor</item>
    <item name="colorPrimary">@color/yourPrimaryColor</item>
    <item name="android:textColorPrimary">@color/yourTextColor</item>
</style>
```

(cf. https://developer.android.com/training/material/theme.html)

- If you already use an AppCompat theme, then simply use it as parent:

```
<style name="Theme.SMTheme" parent="YourTheme"></style>
```

Note: avoid using a "NoActionBar" theme as parent because it would hide the toolbar, rendering the "Response push/In-App message" functionality useless (the icon used to send the response would not be displayed).

5.2.7 Retrieving the Firebase Cloud Messaging (FCM) token

There are two ways to retrieve the FCM token: listening to a broadcast and calling a method.

NB: you will see "GCM" instead of "FCM" in the broadcast and method names, that is because "GCM" stands for "Google Cloud Messaging", which was the previous name of FCM, before Google moved the functionality to Firebase.

5.2.7.1 Broadcast

You can listen to SMManager.BROADCAST_EVENT_RECEIVED_GCM_TOKEN (its value is "SMReceivedGCMToken"). This broadcast is sent after reception of the token from FCM and only if it is different from the one already stored.

It's a local broadcast and, therefore, must be listened to using a LocalBroadcastManager.

The value of the token can be retrieved from the intent received by using SMManager.BROADCAST_DATA_GCM_TOKEN (its value is "SMDataGCMToken").

For example, considering you have a class EventReceiver:

```
public class EventReceiver extends BroadcastReceiver
{
    public void onReceive(Context context, Intent intent)
    {
        String action = intent.getAction();

        switch (action)
        {
            case SMManager.BROADCAST_EVENT_RECEIVED_GCM_TOKEN:
                String gcmToken =
intent.getStringExtra(SMManager.BROADCAST_DATA_GCM_TOKEN);
                //Do some stuff
                break;
        }
    }
}
```

5.2.7.2 SMManager.getInstance().getGCMToken

This method will return the token stored by the SDK.

Note that, as the processing to register to FCM is asynchronous, it is possible that the value returned is either empty or not up-to-date if the registration is not finished yet when the call is made.

5.2.8 Enabling/disabling the notifications

By default, the notifications are enabled. They can be disabled at any time using the following method:

```
SMManager.getInstance().disableNotifications();
```

They are enabled again by doing:

```
SMManager.getInstance().enableNotifications();
```

5.2.9 Setup for special push

5.2.9.1 Map

If you expect to receive Map type notifications, you will need to specify a Google Map key in your manifest. This key needs to be generated with the google developer console.

To create it, please follow the steps described in this page:
https://developers.google.com/maps/documentation/android-api/signup#key-biz

At the end of this procedure you need to add this generated key under the APPLICATION xml tag in the AndroidManifest.xml like this:

```
<meta-data
    android:name="com.google.android.geo.API_KEY"
    android:value="xxxxxxxxxxxxxxx"/>
```

5.2.9.2 Event

When displayed, the message of a push can contain buttons. One type of button can send a broadcast containing a specific value, in order for you to execute some code when you receive it. This broadcast is sent locally using LocalBroadcastManager. In order to listen to it, you need to add a BroadcastReceiver to your app, specify the action (the aforementioned value) and register it using LocalBroadcastManager.

The action needs to be the name of the event specified at the creation of the push.

For example, if the value of the broadcast is "CustomEvent" and considering you have a class EventReceiver:

```
public class EventReceiver extends BroadcastReceiver
{
    public void onReceive(Context context, Intent intent)
    {
        String action = intent.getAction();

        switch (action)
        {
            case "CustomEvent":
                //Perform the actions requested by your app
                break;
        }
    }
}
```

And in your activities/base activity:

```
EventReceiver localReceiver;
…

@Override
protected void onStart()
{
    super.onStart();

    […]
```

```
    if (localReceiver == null)
    {
        localReceiver = new EventReceiver();
    }
    IntentFilter filter = new IntentFilter();
    filter.addAction("CustomEvent");
    LocalBroadcastManager.getInstance(this).registerReceiver(localReceiver, filter);
}
```

5.2.10 Broadcasts

Some specific broadcasts are sent during the management of the push notifications (reception, display and interaction):

BROADCAST_EVENT_RECEIVED_REMOTE_NOTIFICATION : When a push is received, it contains its id and title.
This broadcast is only useful if RemoteMessageDisplayType is set to None, so you can decide when to display the push message. In all other cases, the SDK manages everything itself so it is not needed.

BROADCAST_EVENT_BUTTON_CLICKED : When a button is clicked, it contains an SMNotificationButton object

BROADCAST_EVENT_WILL_DISPLAY_NOTIFICATION : When a message is about to be displayed

BROADCAST_EVENT_WILL_DISMISS_NOTIFICATION : When a message is about to be dismissed

BROADCAST_EVENT_RECEIVED_GCM_TOKEN : When the token is received, it contains the token (cf. Retrieving the Firebase Cloud Messaging (FCM) token)

For more info regarding the broadcasts, go to Broadcasts

5.2.11 Manual display of a push notification

If you set RemoteMessageDisplayType to None and listen to the broadcast BROADCAST_EVENT_RECEIVED_REMOTE_NOTIFICATION, you will want to use the following method to display the push:

```
SMManager.getInstance().displayLastReceivedRemotePushNotification(activity);
```

This method will display the last push received (the SDK only stores the last one), using a dialog or a dedicated Activity, depending on its type.

```
SMManager.getInstance().getLastRemotePushNotification();
```

This method will return a HashMap containing the id and title of the push (the keys are “id” and “title”).

5.3 In-App messages

In-App messages are messages retrieved periodically by the SDK.

They are retrieved when the app becomes active (ie. at start, when going from background to foreground and when the orientation changes) ONLY if the last refresh is older than the value set for InAppMessageRefreshType.

5.3.1 Libraries

If you plan on sending In-App messages containing “Response” type buttons, you have to add the following third party library in your dependencies:

com.drewnoakes:metadata-extractor:2.9.1

5.3.2 Permissions

There isn't any mandatory permission required to use the In-App messages in general. However, like for push notifications, some In-App messages will require special permissions in order to be displayed properly.

If you plan on sending "Map" type messages, you might want to add one of the following permissions in order for the user location to be displayed on the map:

```
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
or
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

Only one of the two is needed. Coarse location is less precise than fine location. Note that if you do not add any, the map will still be displayed, just not the user location.

If you plan on sending messages that include a "Response with picture" type button, you need to add the following permission:

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

This is needed to access the pictures on the device and call the photo app. If it is not present in the manifest, the user will not be able do anything and a message "Not enough permission" will appear instead.

Warning: do NOT use "READ_EXTERNAL_STORAGE" instead of "WRITE_EXTERNAL_STORAGE", this would result in the app crashing when the user tries to access the camera or the gallery.

5.3.3 Enabling/disabling the In App-messages

In app messages are disabled by default, unless you set InAppMessageRefreshType on SMSettings.
It is highly recommended to avoid setting the value to Minutely for production. It is there for testing purpose only.

If you want to disable them at some point (or if you want to give the user the ability to do it), use the following method:

```
SMManager.getInstance().disableInAppMessages();
```

They are enabled again by doing (this can also be used to change the refresh type):

```
SMManager.getInstance().enableInAppMessages(SMInAppRefreshType.Daily);
```

Those two methods can be called anywhere in your app.

5.3.4 Reception of the messages

The SDK retrieved the In-App messages automatically, according to the setting InAppMessageRefreshType set when starting the SDK. When messages are received, a broadcast is sent:

BROADCAST_EVENT_RECEIVED_IN_APP_MESSAGE.

Use `SMManager.BROADCAST_DATA_IN_APP_MESSAGES` to retrieve them from the intent.
(cf. Local broadcasts for more information on how to use it)

Only the title and id of each message are sent. They can be used to display some kind of inbox (a list of the title of the messages).

The list received by this broadcast contains all the In-App messages that haven't been read by the user yet. Therefore, it is possible for you to receive some messages that you already got earlier.

5.3.5 Display of an In-App message

Once you have the In-App messages, you can display one using the following method:
`SMManager.getInstance().displayMessage(messageId, activity);`

`messageId` is the id of the In-App message to display (received by listening to the broadcast as discussed in Reception of the messages) and `activity` the Activity that will display it.
It will be displayed the way the push notifications are.

5.3.6 Broadcasts

Some specific broadcasts are sent during the management of the In-App messages (reception, display and interaction):

BROADCAST_EVENT_RECEIVED_IN_APP_MESSAGE : When In-App messages are received, it contains an array of SMInAppMessages.

BROADCAST_EVENT_BUTTON_CLICKED : When a button is clicked, it contains an SMNotificationButton object

BROADCAST_EVENT_WILL_DISPLAY_NOTIFICATION : When a message is about to be displayed

BROADCAST_EVENT_WILL_DISMISS_NOTIFICATION : When a message is about to be dismissed

For more info regarding the broadcasts, go to Broadcasts

5.4 In-App contents

In-App contents are retrieved periodically by the SDK. They are messages that will be displayed inside your activities, either through our fragments or your own views, contrary to In-App and push messages that are displayed in a dialog or a dedicated Activity.

New content is retrieved when the app becomes active (ie. at start, when going from background to foreground and when the orientation changes) ONLY if the last refresh is older than the value set for InAppContentRefreshType.

5.4.1 Libraries

In order for the project to build, you must have a dependency to the following library:

- com.android.support:cardview-v7

5.4.2 Enabling the In-App contents

In-App contents are disabled by default, unless you set InAppContentRefreshType on SMSettings.
It is highly recommended to avoid setting the value to Minutely for production. It is there for testing purpose only.

5.4.3 Implementing the In-App content using Selligent tools

There are 3 types of In-App contents: Image, URL and HTML. Each has its own Fragment, respectively: SMInAppContentImageFragment, SMInAppContentUrlFragment and SMInAppContentHtmlFragment.

Each one can either be displayed as a full screen dialog using the “show” method (they all extend DialogFragment) or be used as a standard Fragment (cf. official Android documentation on Fragments).

Note that you cannot reference them directly in a layout using a “fragment” tag because they need arguments. You have to instantiate them using the static method “newInstance” and add them programmatically to the layout.

Once instantiated, you can call the method hasContent() to know if there is any content available for the given category, so you can display something else instead of the fragment.

Example:

Considering the following layout:

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">

    […]

    <FrameLayout
        android:layout_width="match_parent"
        android:layout_height="300dp"
        android:id="@+id/urlFragmentContainer"/>

    </LinearLayout>
```

You will have this kind of code:

```
SMInAppContentUrlFragment urlFragment = SMInAppContentUrlFragment.newInstance("testUrl");

if (urlFragment.hasContent())
{
    FragmentManager fragmentManager = getFragmentManager();
    FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
    fragmentTransaction.replace(R.id.urlFragmentContainer, urlFragment, "URL");
    fragmentTransaction.commit();
}
else
{
     //do some other stuff
}
```

In-App contents are sorted by categories (it is one of their property at creation). Therefore, all our Fragments require to be given that category at instantiation. Each one will only display the In-App content corresponding to it.

SMInAppContentImageFragment and SMInAppContentUrlFragment both display only one content at a time. It is not the case for the HTML type, so SMInAppContentHtmlFragment needs a second argument which is the number of In-App contents to display. Setting it to -1 means that all available contents for the given category will be used.

5.4.3.1 Customization

The layout of the In App contents is entirely customizable. We have a default one that is defined in the file "styles.xml". Here is its content:

```
<style name="Selligent.InAppContents.Image">
    <item name="android:background">#FFFFFF</item>
</style>
<style name="Selligent.InAppContents.Html.Container">
    <item name="android:background">#EEEEEE</item>
    <item name="android:layout_margin">0dp</item>
</style>
<style name="Selligent.InAppContents.Html.Card">
    <item name="cardBackgroundColor">#FFFFFF</item>
    <item name="cardCornerRadius">2dp</item>
    <item name="cardElevation">2dp</item>
    <item name="cardMaxElevation">2dp</item>
    <item name="cardPreventCornerOverlap">true</item>
    <item name="cardUseCompatPadding">true</item>
    <item name="contentPadding">5dp</item>
    <item name="contentPaddingBottom">5dp</item>
    <item name="contentPaddingLeft">5dp</item>
    <item name="contentPaddingRight">5dp</item>
    <item name="contentPaddingTop">5dp</item>
</style>
<style name="Selligent.InAppContents.Html.CardContainer"/>
<style name="Selligent.InAppContents.Html.CardTitle">
    <item name="android:textSize">24sp</item>
```

```
    <item name="android:textColor">#000000</item>
    <item name="android:paddingLeft">16dp</item>
    <item name="android:paddingRight">16dp</item>
    <item name="android:paddingTop">24dp</item>
    <item name="android:paddingBottom">16dp</item>
</style>
<style name="Selligent.InAppContents.Html.UpperDivider">
    <item name="android:layout_height">0dp</item>
    <item name="android:visibility">gone</item>
</style>
<style name="Selligent.InAppContents.Html.CardBody">
    <item name="android:textSize">14sp</item>
    <item name="android:textColor">#555555</item>
    <item name="android:paddingLeft">16dp</item>
    <item name="android:paddingRight">16dp</item>
    <item name="android:paddingTop">0dp</item>
    <item name="android:paddingBottom">16dp</item>
</style>
<style name="Selligent.InAppContents.Html.LowerDivider">
    <item name="android:layout_height">0dp</item>
    <item name="android:visibility">gone</item>
</style>
<style name="Selligent.InAppContents.Html.CardLinkContainer">
    <item name="android:paddingLeft">8dp</item>
    <item name="android:paddingRight">8dp</item>
    <item name="android:paddingTop">8dp</item>
    <item name="android:paddingBottom">8dp</item>
    <item name="android:gravity">start</item>
</style>
<style name="Selligent.InAppContents.Html.CardLink">
    <item name="android:padding">8dp</item>
    <item name="android:layout_marginRight">8dp</item>
    <item name="android:textSize">14sp</item>
    <item name="android:textColor">#333333</item>
    <item name="android:textAllCaps">true</item>
    <item name="android:background">?android:attr/selectableItemBackground</item>
</style>
<style name="Selligent.InAppContents.CloseButton">
    <item name="android:layout_width">30dp</item>
    <item name="android:layout_height">30dp</item>
    <item name="android:layout_margin">5dp</item>
    <item name="android:padding">10dp</item>
    <item name="android:background">@drawable/sm_ic_close_image</item>
</style>
```

To customize it, in your own file “styles.xml”, override the styles you want to modify. Note that you have to copy the whole content of those styles, not only the items you want to change, otherwise the others will be lost.

For example, if you simply want to change the background color of a Card to red, you still have to add in your file the whole style:

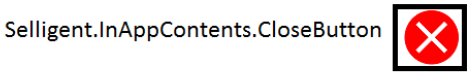
```
<style name="Selligent.InAppContents.Html.Card">
    <item name="cardBackgroundColor">#FF0000</item>
    <item name="cardCornerRadius">2dp</item>
    <item name="cardElevation">2dp</item>
    <item name="cardMaxElevation">2dp</item>
    <item name="cardPreventCornerOverlap">true</item>
    <item name="cardUseCompatPadding">true</item>
    <item name="contentPadding">5dp</item>
    <item name="contentPaddingBottom">5dp</item>
    <item name="contentPaddingLeft">5dp</item>
    <item name="contentPaddingRight">5dp</item>
    <item name="contentPaddingTop">5dp</item>
</style>
```

The visual elements corresponding to those styles are defined below.

5.4.3.1.1 Shared style

The button used to close the In App Content when displayed full screen is available for all contents.

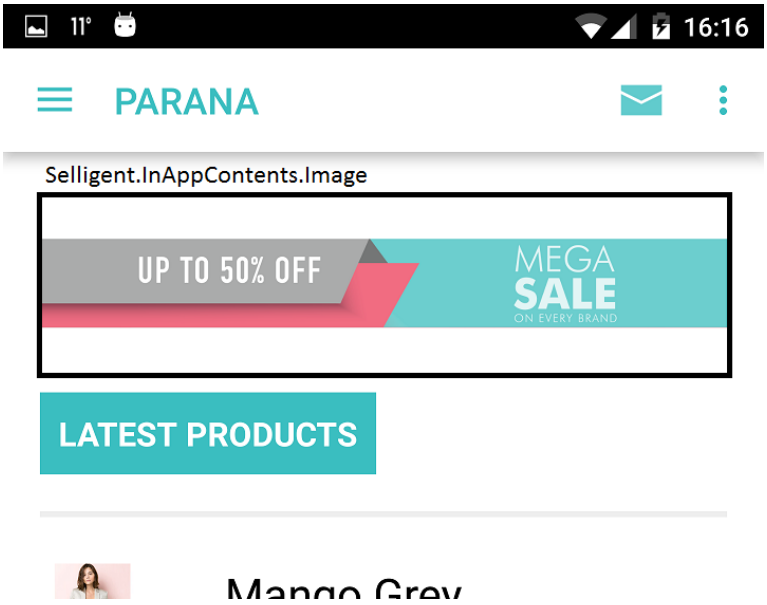
<style name="Selligent.InAppContents.CloseButton">



5.4.3.1.2 SMInAppContentImageFragment

<style name="Selligent.InAppContents.Image">

This style can be used, for example, to set a default background color when the image does not completely fill the space reserved for the fragment.

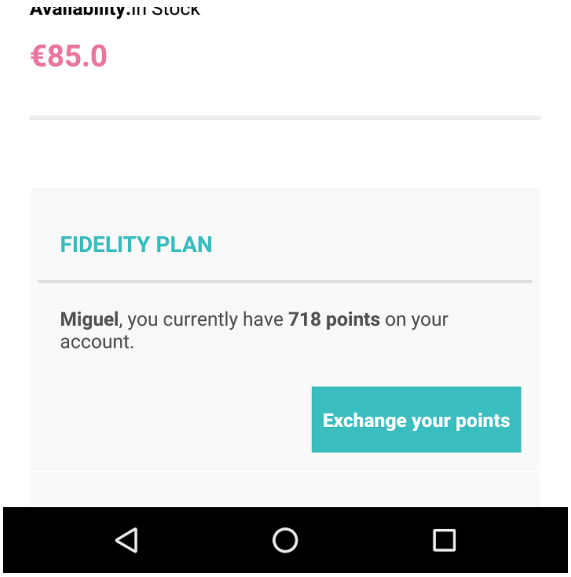


5.4.3.1.3 SMInAppContentUrlFragment

The url is displayed using the full size of the Fragment, so there is no customization available.

5.4.3.1.4 SMInAppContentHtmlFragment

Consider the following HTML type In App Content:



The following styles can be used to customize it:

1 <style name="Selligent.InAppContents.Html.Container">

2 <style name="Selligent.InAppContents.Html.Card"> (This one is used on the CardView element)

3 <style name="Selligent.InAppContents.Html.CardContainer"> (This one is used on the RelativeLayout element, first child of the CardView)

4 <style name="Selligent.InAppContents.Html.CardTitle">

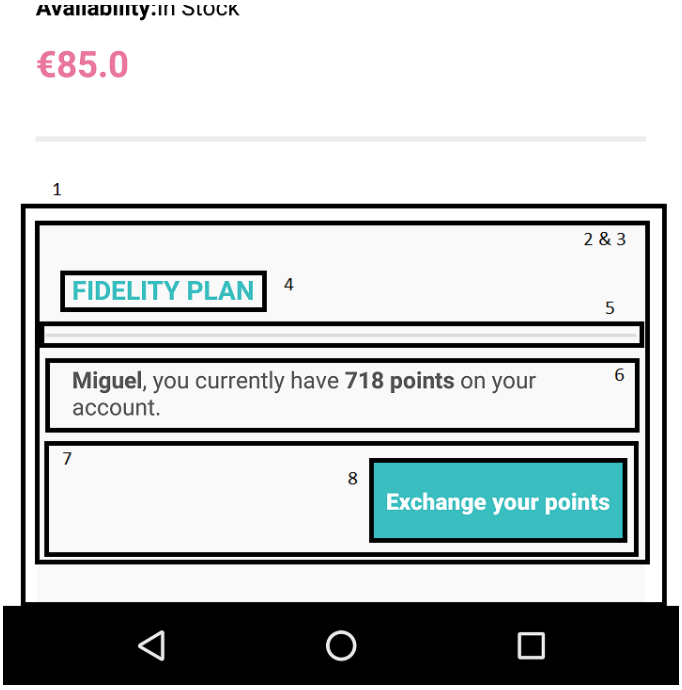
5 <style name="Selligent.InAppContents.Html.UpperDivider">

6 <style name="Selligent.InAppContents.Html.CardBody">

<style name="Selligent.InAppContents.Html.LowerDivider"> (It is not used in this screenshot. It is the equivalent of 5 but situated between the body and the links).

7 <style name="Selligent.InAppContents.Html.CardLinkContainer">

8 <style name="Selligent.InAppContents.Html.CardLink">



5.4.3.2 Refresh

If you ever want to refresh the content of one of our Fragments in case new content is available, you can call the "refresh" method. If the fragment is not displayed, it will just get the (possible) new content from the cache. If it is displayed, the layout will be refreshed too.

5.4.4 Implementing the In-App content using your own controllers

If you prefer to use your own controls to display the In-App contents, we give you a broadcast and a few methods to get and manage them. They are all available on SMManager.

First, to know when In-App contents are available, you have to listen to the broadcast ***BROADCAST_EVENT_RECEIVED_IN_APP_CONTENTS.*** It contains a HashMap with the number of content per category (the key is the category; the value is the count), not the actual contents. If a category is not present, it means there is no content for it.

Then, you can use the following methods:

- `public ArrayList<SMInAppContent> getInAppContents(String category, SMContentType type, int max)`
  This will return all the valid In-App contents for a given category and a given type. If you want a certain amount of contents, just pass it to the method, otherwise pass -1 to get all the available ones.
  The actual content is in the body property of each SMInAppContent and its value, retrieved using the method "getBody()", depends on the In-App content type:
  - Image: the URL of the image
  - HTML: a text with some HTML formatting.
  - URL: a URL

  If an Image type In-App content is marked as downloadable at creation, when the SDK retrieves it, it will automatically start downloading the bitmap asynchronously. That bitmap can later be obtained using the "getImage" method.

- `public void setInAppContentAsSeen(SMInAppContent inAppContent)`
  This will mark the In-App content as seen and send an "open" event to the platform.

- `public void executeLinkAction(Context context, SMLink link, SMInAppContent content)`
  This will execute the action behind the link (open browser, open phone app, etc.) and send a "click" event to the platform.

5.4.5 Broadcasts

BROADCAST_EVENT_RECEIVED_IN_APP_CONTENTS: When In-App contents are received.
BROADCAST_EVENT_BUTTON_CLICKED : When a link is clicked, it contains an SMNotificationButton object.
(cf. Local broadcasts)

5.5 Events

The following method can be used to send specific messages to the web service.

```
SMManager.getInstance().sendSMEvent(SMEvent event);
```

The kind of event message sent to the user will depend of the class of object given to the method. All the different classes extend SMEvent. They are described in the following points.

NB: Since 1.3, the data passed to the SMEvent is Hashtable<String, String> (In earlier versions it was Hashtable<String, Object>).

5.5.1 Registration/Unregistration

5.5.1.1 SMEventUserRegister

This object is used to send a “register” event to the server with the e-mail of the user, potential data and a callback.

Ex:

```
Hashtable<String, String> hash = new Hashtable<>();
hash.put("Key1", "Registration value 1");
hash.put("Key2", "Registration value 2");
hash.put("Key3", "Registration value 3");
SMEvent event = new SMEventUserRegister("user@company.com", hash,
        new SMCallback()
        {
            @Override
            public void onSuccess(String result)
            {
                [Do something here]
            }

            @Override
            public void onError(int responseCode, Exception exception)
            {
                [Do something here]
            }
        });
)
SMManager.getInstance().sendSMEvent(event);
```

5.5.1.2 SMEventUserUnregister

This object is used to send an “unregister” event to the server with the e-mail of the user, potential data and a callback.

Ex:

```
Hashtable<String, String> hash = new Hashtable<>();
hash.put("Key1", "Unregistration value 1");
hash.put("Key2", "Unregistration value 2");
hash.put("Key3", "Unregistration value 3");
SMEvent event = new SMEventUserUnregister("user@company.com", hash,
        new SMCallback()
        {
            @Override
            public void onSuccess(String result)
            {
                [Do something here]
            }

            @Override
            public void onError(int responseCode, Exception exception)
            {
                [Do something here]
            }
        });
SMManager.getInstance().sendSMEvent(event);
```

5.5.2 Login/Logout

5.5.2.1 SMEventUserLogin

This object is used to send a “login” event to the server with the e-mail of the user, potential data and a callback.

Ex:

```
Hashtable<String, String> hash = new Hashtable<>();
hash.put("Key1", "Login value 1");
```

```
hash.put("Key2", "Login value 2");
hash.put("Key3", "Login value 3");
SMEvent event = new SMEventUserLogin("user@company.com", hash,
        new SMCallback()
        {
            @Override
            public void onSuccess(String result)
            {
                [Do something here]
            }

            @Override
            public void onError(int responseCode, Exception exception)
            {
                [Do something here]
            }
        });
SMManager.getInstance().sendSMEvent(event);
```

5.5.2.2 SMEventUserLogout

This object is used to send a “logout” event to the server with the e-mail of the user, potential data and a callback.

Ex:

```
Hashtable<String, String> hash = new Hashtable<>();
hash.put("Key1", "Logout value 1");
hash.put("Key2", "Logout value 2");
hash.put("Key3", "Logout value 3");
SMEvent event = new SMEventUserLogout("user@company.com", hash,
        new SMCallback()
        {
            @Override
            public void onSuccess(String result)
            {
                [Do something here]
            }

            @Override
            public void onError(int responseCode, Exception exception)
            {
                [Do something here]
            }
        });
SMManager.getInstance().sendSMEvent(event);
```

5.5.3 Custom

5.5.3.1 SMEvent

This object is used to send a custom event to the server with some data and a callback.

Ex:

```
Hashtable<String, String> hash = new Hashtable<>();
hash.put("Key1", "Custom value 1");
hash.put("Key2", "Custom value 2");
hash.put("Key3", "Custom value 3");
SMEvent event = new SMEvent(hash,
        new SMCallback()
        {
            @Override
            public void onSuccess(String result)
            {
                [Do something here]
            }

            @Override
            public void onError(int responseCode, Exception exception)
            {
                [Do something here]
            }
        });
SMManager.getInstance().sendSMEvent(event);
```

5.6 Geolocation

Geolocation is managed through a 3rd party library by PlotProjects, so you need to add a dependency to it (cf. Other libraries).
To configure that library, you must add a plotconfig.json file in the asset folder of your app. There, you can set a few properties but only one is mandatory: the Plot public token. For more information, check PlotProjects documentation.
Example:

```
{
  "publicToken": "YOUR_PUBLIC_TOKEN",
  "debug": true
}
```

To tell the SDK to use the Geolocation, set the property ConfigureGeolocation to true on the SMSettings object (cf. Optional settings).

If you set "enableOnFirstRun" to false in the plotconfig.json file, geolocation will be initialized but NOT enabled. So the permission will not be asked to the user and you will have to enable it manually.
To do so, you can call the method SMManager.getInstance().enableGeolocation(). Note that this will enable the geolocation but not ask the permission, you must do it yourself in that case.
This method, along with SMManager.getInstance().disableGeolocation() and SMManager.getInstance().isGeolocationEnabled() can also be used to provide the user with an opt-in/opt-out functionality.

NB:

- Don't initialize or call any method of the PlotProjects API in your app, everything is managed by our SDK.
- Our SDK sets the icon used by plot for the geolocation notifications, reusing the one that you already gave us, so no need to specify it a second time in the config file.
- Default value for "enableOnFirstRun" is true, so if you specify only the token in the config file, you don't have to call any method.

5.7 Broadcasts

A certain number of broadcasts are sent from the SDK at different moments. You can listen to them to be able to execute some code related to those events.
Refer to the template project for examples of what to do with the data sent with those broadcasts.

5.7.1 Generic broadcasts

Due to limitations to what can be done in background starting with Android O, this broadcast is now deprecated with SDK 1.6.0. It is still sent but you won't be able to listen to it if your app targets android O (targetSdk=26) and runs on an Android O device.

This does not affect the local broadcasts which continue to work normally.

These broadcasts are sent using Context.sendBroadcast(Intent). In order to listen to them, you have to register a BroadcastReceiver, either in your AndroidManifest.xml file or dynamically in your activity. Each of these require a category filter with the package name of your app.

```
BROADCAST_EVENT_RECEIVED_REMOTE_NOTIFICATION = "SMReceivedRemoteNotification"
```

Example (considering your package name is com.mycompany.myapp):

```
<receiver android:name=".EventReceiver">
    <intent-filter>
        <action android:name="SMReceivedRemoteNotification"/>

        <category android:name="com.mycompany.myapp"/>
```

```
    </intent-filter>
</receiver>
```

And in your class EventReceiver:

```
public class EventReceiver extends BroadcastReceiver
{
    public void onReceive(Context context, Intent intent)
    {
        String action = intent.getAction();

        switch (action)
        {
            case SMManager.BROADCAST_EVENT_RECEIVED_REMOTE_NOTIFICATION:
                 String id = intent.getStringExtra("id");
                String title = intent.getStringExtra("title");
                break;
        }
    }
 }
```

If you want to register your receiver dynamically:

```
EventReceiver receiver;
...
@Override
protected void onStart()
{
    super.onStart();

    if (receiver == null)
    {
        receiver = new EventReceiver();
    }
    IntentFilter filter = new IntentFilter();
    filter.addAction(SMManager.BROADCAST_EVENT_RECEIVED_REMOTE_NOTIFICATION);
    registerReceiver(receiver, filter);
}
```

5.7.2 Local broadcasts

These broadcast are sent using LocalBroadcastManager, they are local to the app. In order to listen to them, you have to register a BroadcastReceiver with LocalBroadcastManager dynamically in your activity. As they are local to your app, they do not require a category filter.

```
BROADCAST_EVENT_RECEIVED_IN_APP_MESSAGE = "SMReceivedInAppMessage"
BROADCAST_EVENT_RECEIVED_IN_APP_CONTENTS = "SMReceivedInAppContent"
BROADCAST_EVENT_BUTTON_CLICKED = "SMEventButtonClicked"
BROADCAST_EVENT_WILL_DISPLAY_NOTIFICATION = "SMEventWillDisplayNotification"
BROADCAST_EVENT_WILL_DISMISS_NOTIFICATION = "SMEventWillDismissNotification"
BROADCAST_EVENT_RECEIVED_GCM_TOKEN = "SMReceivedGCMToken";
```

For example, consider you have a class EventReceiver:

```
public class EventReceiver extends BroadcastReceiver
{
    String action = intent.getAction();

    switch (action)
```

```
    {
        case SMManager.BROADCAST_EVENT_RECEIVED_IN_APP_MESSAGE:
            SMInAppMessage[] messages =
(SMInAppMessage[])intent.getSerializableExtra(SMManager.BROADCAST_DATA_IN_APP_MESSAGES);
            //Do some stuff
            break;

        case SMManager.BROADCAST_EVENT_RECEIVED_IN_APP_CONTENTS:
            HashMap<String, Integer> categories = (HashMap<String, Integer>)
        intent.getSerializableExtra(SMManager.BROADCAST_DATA_IN_APP_CONTENTS);
            //Do some stuff
        Break;


        case SMManager.BROADCAST_EVENT_BUTTON_CLICKED:
            SMNotificationButton button =
(SMNotificationButton)intent.getSerializableExtra(SMManager.BROADCAST_DATA_BUTTON);
            //Do some stuff
            break;

        case SMManager.BROADCAST_EVENT_WILL_DISPLAY_NOTIFICATION:
            //Do some stuff
            break;

        case SMManager.BROADCAST_EVENT_WILL_DISMISS_NOTIFICATION:
            //Do some stuff
            break;

        case SMManager.BROADCAST_EVENT_RECEIVED_GCM_TOKEN:
            String gcmToken = intent.getStringExtra(SMManager.BROADCAST_DATA_GCM_TOKEN);
            //Do some stuff
            break;
    }
}
```

And in your activities:

```
EventReceiver localReceiver;
...
@Override
protected void onStart()
{
    super.onStart();

    if (localReceiver == null)
    {
        localReceiver = new EventReceiver();
    }
    IntentFilter filter = new IntentFilter();
    filter.addAction(SMManager.BROADCAST_EVENT_BUTTON_CLICKED);
    filter.addAction(SMManager.BROADCAST_EVENT_WILL_DISMISS_NOTIFICATION);
    filter.addAction(SMManager.BROADCAST_EVENT_WILL_DISPLAY_NOTIFICATION);
    filter.addAction(SMManager.BROADCAST_EVENT_RECEIVED_IN_APP_MESSAGE);
    LocalBroadcastManager.getInstance(this).registerReceiver(localReceiver, filter);
}
```

5.8 Translations

When asking for a permission, a text is displayed explaining why we need it. By default, it is in English but a translation for few languages is provided: Dutch, Spanish, French and German. If you want to add another language (or change the message), add under "res" a folder named "value-[language code]" (example: to add Russian, you would name it "value-ru"), create a "strings.xml" file and, under a <resources> tag, add:

```
<string name="sm_permission_explanation_location">Write here the message you want</string>
<string name="sm_permission_explanation_write_external_storage">Write here the message you
want</string>
<string name="sm_permissions_not_enough">Write here the message you want</string>
```

6 Proguard

If you are using Proguard to minify the code of your app, add the following lines to the file proguard-rules.pro:

```
-dontwarn com.selligent.sdk.**
-keep class com.selligent.sdk.* {
    public private *;
}
```

7 Changelog

Version 1.9

- Added management of buttons inside the notification.

Version 1.8

- Added management of push without In-App messages and with an action triggered when clicking on the notification (like a deep link).

Version 1.7.2

- Corrected a bug preventing the push to be correctly handled when received while on a webview opened by a previous notification.

Version 1.7.1

- Corrected a bug preventing the image to be correctly displayed in an InApp-Content fragment when the image was larger than the screen.

Version 1.7.0

- Added geolocation functionality
- Android 8.1 compatible

Version 1.6.1

- Bug correction

Version 1.6.0

- Adaptations for Android O.
- Removal of the service com.selligent.sdk.GcmIntentService. If you reference it in your AndroidManifest.xml file, remove that entry.
- The broadcast ***BROADCAST_EVENT_RECEIVED_REMOTE_NOTIFICATION*** is now deprecated due to limitations with Android O. It is still sent but will not be received by an app targeting Android O and running on an Android O device.

Version 1.5.0

- Image type In App contents can now be marked at creation as downloadable. In that case, after retrieving the In App contents, the SDK will (asynchronously) download the image for each content marked as such and store it on the device.
- Due to changes in the Android SDK, the inclusion of the third party library **com.drewnoakes:metadata-extractor** is now required when using “Response” type buttons within a push or In-App messages.
- “Dangerous” permissions (those requiring to be explicitly granted by the user under Android 6.0 and above) are not included in the AndroidManifest.xml by the SDK anymore, it will have to be done in the app manifest. This will allow to restrict the permissions to the functionality actually used by the app. See each functionality to know which permission they require.
- Added support for Android SDK 24 and 25.
- Added management of “Passbook” type buttons with push, In-App message and In-App content. Clicking on such a button will open a passbook app if the device has one or the browser instead.

Version 1.4.3

- sendSMEvent, when used with a custom event, will only send it if the data passed is new. If all entries in the hashtable have the same values as the last time it was sent, then we won't do anything. If you want to log when a specific action happened and the values do not change, add a date in the data.

Version 1.4.2

- Selligent SDK now available from JCenter

Version 1.4

- SMSettings.Theme is now deprecated. This value is not used anymore as the layout of the dialog in now completely customizable (cf. Dialog).
- The design of a dialog does not try to adapt to the theme and the version of Android anymore, instead there is a default material layout that is entirely customizable (cf. Dialog).
- The SetInfo event is now sent only when some of its info changed, not systematically at each start of the SDK anymore.
- Added support for Android SDK 23 and the new way permissions are managed.
- Added In-App contents management.

8 Troubleshooting

- **Q.: When I look at the devices in Campaign, why don't they have a token?**
  A.: If there is no token, that means the SDK did not send it. There are a few reasons why this could happen:
  - If you don't use the json file (cf. Creating a Google application), then you have to call the method registerDevice(Context context) on SMManager in the onStart method of your base Activity. Also, check if the senderId is correctly passed to the SDK (SMSettings.GoogleApplicationId).
  - If you use the json file, make sure that it is correctly placed in your app and that you updated your build.gradle files as described on the Firebase website when adding cloud messaging to your app.

- **Q.: Why don't I receive the push on my device?**
  A.: You have to check a few things. First, look at the push status in Campaign. Here are some errors you might encounter:
  - **Device subscription expired**. This means the token used to contact the device is not valid anymore. It may be because the user hasn't used the app for a while and the token expired or the user uninstalled the app. You can't do anything here except wait for a new one to be received.
  - **Mismatch sender id**. This means the token used to contact the device was created with a sender id that does not correspond to the server key stored on the Selligent platform. Check both values on Firebase and then change the sender id given to the SDK and/or send us the correct server key.
  - **Authentication failed**. The server key that you gave us is not correct. Check it on Firebase and send us the new one.
  - If the push optout is 1, then the push will not be sent to the device. Maybe it was not able to get a token or there was a call to the method disableNotifications().

  If the status in Campaign is ok, then there might be something wrong in your app.

- If you don't use the JSON file, check that you added the correct permissions in your manifest (cf. Permissions for Push notifications)
- Do you have another BroadcastReceiver listening to GCM/FCM push? If yes, then it might be trying to interpret our JSON payload and crash before our receiver has time to finish its work. You can recognize a Selligent push at its "sm" property at the root of the JSON.
- When starting the SDK, did you set the property RemoteMessageDisplayType to None? This will prevent it from creating a notification and displaying the message when receiving a push while the app is in foreground.
- Did you call the method disableNotifications()? This will set the push optout to 1 in Campaign, preventing it to send push to that device.

- **Q.: My app crashes when opening, with the following trace**

```
at com.google.android.gms.iid.zzd.zzdo(Unknown Source)
at com.google.android.gms.iid.zzd.<init>(Unknown Source)
at com.google.android.gms.iid.zzd.<init>(Unknown Source)
at com.google.android.gms.iid.InstanceID.zza(Unknown Source)
at com.google.android.gms.iid.InstanceID.getInstance(Unknown Source)
at
com.selligent.sdk.SMRegistrationIntentService.getInstanceID(SMRegistrationIntentSe
rvice.java:19).
```

A.: This can happen if the targetSdk of your app is 23 or above and your version of Google Play Services is 8.+. This is a known bug in Google Play Services that was corrected in later versions. Try using 9.2 or above.

Q.: I see "The SDK did not start correctly" in the logs, what should I do?

A: Check your code starting the SDK. Did you correctly set the 3 following values: webServiceUrl, clientId and privateKey? If any is missing, that message will appear.

- **Q.: I try to send an event but nothing happens, the callback methods are not called**

A: The SDK did not start correctly (cf. previous question). You can see in the logs a message telling you that by setting
SMManager.DEBUG = true
before calling the start method. If you did but still don't see anything, then it means your code starting the SDK is not called. In this case, you need to check your AndroidManifest.xml file and make sure that your class is correctly referenced in it. If your class is called "MyCustomApplication" and you package is "com.mycompany.myapp", then you must have in the manifest:
<application android:name="com.mycompany.myapp.MyCustomApplication"
Note that the package name is not mandatory, you can simply specify the name of the class preceded by a ".".