Hello !
LE PUSH NOTIF 2.0

HighCONNEXION

Hello SDK Android 2.0

1.Intégration **2**

2. Configuration du SDK **3**

1. Configuration à faire dans manifest 3
2. Configuration à faire dans le code 3
3. Permettre au serveur de connaître l'existence de ce nouveau device 4

3. Autres fonctions **4**

1. Récupérer les segments 4
2. Configurer les segments 4
3. Autres configurations 5

1.Intégration

Pour l’intégration de notre librairie dans votre projet, nous recommandons l’utilisation d’Android Studio et Gradle.

Ajouter le SDK dans votre”app module's dependencies” dans Android Studio en ajoutant les lignes suivantes dans `dependencies { ... }` configuration:

```
compile 'com.hcnx:hello:2.0'
compile 'com.hcnx:hcnx_base:2.0'
compile 'com.google.firebase:firebase-messaging:10.0.0'
compile 'com.google.code.gson:gson:2.4'
```

Merci de vérifier que la configuration de votre repository contient :

```
repositories {
  maven {
        url "https://bitbucket.org/devmobile/maven-repo-public/raw/master/repository/"
    }
}
```

2. Configuration du SDK

1. Configuration à faire dans manifest

```
<service
      android:name="com.hcnx.hello.reception.HNPushListenerService"
      android:exported="false">
      <intent-filter android:priority="-400">
           <action android:name="com.google.firebase.MESSAGING_EVENT" />
      </intent-filter>
</service>
<service android:name="com.hcnx.hello.reception.HNInstanceIDListenerService">
     <intent-filter>
           <action android:name="com.google.firebase.INSTANCE_ID_EVENT" />
     </intent-filter>
</service>
<meta-data
    android:name="com.google.android.gms.version"
    android:value="@integer/google_play_services_version" />
<meta-data
    android:name="com.hcnx.highnotif.default_ic_notif"
    android:value="ic_notif" />
<meta-data
    android:name="com.hcnx.highnotif.default_title_notif"
    android:value="@string/app_name" />
```

2. Configuration à faire dans le code

Dans votre classe qui étend android.app.Application :

```
 Hello.getInstance(context).configure(apikey, hmackey,senderId);
```

3. Permettre au serveur de connaître l’existence de ce nouveau device

```
Hello.getInstance(context).register(new RegisterCallback() {
    @Override
    public void onRegisterFinished(String subscriberId) {

    }
});
```

3. Autres fonctions

1. Récupérer les segments

- **Récupérer ses segments (Segments) configurés depuis la plateforme Hello ! afin de gérer vos populations Opt'in / Opt'out**

```
Hello.getInstance(getContext()).getSegments(new GetSegmentsCallback() {
    @Override
    void onSegmentsReceived(ArrayList<HNSegment> segments, String subId, Integer
errorCode){

    }
});
```

2. Configurer les segments

- S'abonner / se désabonner à un ou plusieurs segments

```
Hello.getInstance(getContext()).manageSegments(new HelloCallback() {
    @Override
    onRequestFinished(boolean success, String subId, Integer errorCode){

    }
}, subscribeSegments, unSubscribeSegments);
```

subscribeSegments et unSubscribeSegments sont des `ArrayList<HNSegment>.`
La désinscription prend le pas sur l'inscription si deux IDs se retrouvent dans les deux tableaux.
L'un des deux tableaux peut être vide.

3. Autres configurations

- **Récupérer son HN_ID : Identifiant d'abonné à la plateforme Hello !**

```
Hello.getHnId(context);
```

- **Récupérer son Token de Notification Push**

```
Hello.getToken(context);
```

- **Lier son ConsumerId d'application à son Token sur la plateforme Hello !**

```
Hello.getInstance(context).setConsumerId(consumerId)
```

`Consumerid est une String`