SDK_HCNX Android 3.0.2

Warning: if you previously have used one of our SDKs (Hello, MGS,...), please remove their dependencies in gradle and replace them with a single HCNX dependence such as

```
compile 'com.hcnx:hcnxframework:3.0.2'
```

You must always go through HCNX to access other modules: e.g. :

```
HCNX.getInstance().getHello()
```

If you have an issue like :
failed to resolve com.android.support...
please make sure that you have *"google()"* or "*maven { url 'https://maven.google.com' }*" (if the google one doesn't work) in the repositories list of your global build.gradle :

```
allprojects {
        repositories {
          google()
        }
}
```

1. Integration

To integrate our library in your project, we recommend to use Android Studio and Gradle.

Add the SDK in your "app module dependencies" in Android Studio adding the following lines in `dependencies { ... }` configuration of the **build.gradle PROJECT** file:

```
compile 'com.hcnx:hcnxframework:3.0.2'

compile 'com.google.firebase:firebase-messaging:10.2.0'
```

In all case, Please add our maven repository to the configuration of your repository to the build.gradle GLOBAL file.

```
repositories {

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

...
}
```

1.1 Integration with firebase config file (google-service.json)

- add this line at the end of the file

```
apply plugin : 'com.google.gms.google-services'
```

- make sure to add the firebase config file in your project (google-service.json) and to add the classpath 'com.google.gms:google-services:3.1.0' in the dependencies of the **Global** build.gradle

1.2 Integration without firebase config file

Remove the line `apply plugin : 'com.google.gms.google-services'` if you have it at the end of the build.gradle

2. SDK's configuration

2.1. Application file

2.1.1 Application file with firebase config file

In the onCreate method of your application file, add the following lines :

```
HCNX
        .getInstance()
        .configure(getApplicationContext(),getString(R.string.highnotif_api_key),
                                                getString(R.string.highnotif_hmac));
```

With the strings property filled in the res/values/strings.xml :

```
<string name="highnotif_api_key">YOUR_API_KEY</string>
<string name="highnotif_hmac">YOUR_HMAC_KEY</string>
```

2.1.2 Application file without firebase config file

In the onCreate method of your application file, add the following lines :

```
HCNX
        .getInstance()
        .configure(getApplicationContext(),getString(R.string.highnotif_api_key),
                                                getString(R.string.highnotif_hmac),
                                                getString(R.string.sender_id));
```

With the strings property filled in the res/values/strings.xml :

```
<string name="highnotif_api_key">YOUR_API_KEY</string>
<string name="highnotif_hmac">YOUR_HMAC_KEY</string>
<string name="sender_id">YOUR_SENDER_ID</string>
```

Don’t use configure the methods without senderId when you are using the firebase config file.

Don’t initialize Firebase by yourself, the SDK or Firebase does it for you.

2.2. AndroidManifest.xml

In the Android Manifest application section, you should add two meta-datas to define the icon name and the title used by the HighConnexion sdk :

```
    <meta-data
           android:name="com.hcnx.hello.default_ic_notif"
           android:value="ic_notif" />
    <meta-data
           android:name="com.hcnx.hello.default_title_notif"
           android:value="@string/app_name" />
```

By default, the sdk use the application icon , title is not mandatory.

3. Media Games solution

The HCNX SDK provides methods to use our Media Games solutions (https://app.mgs.media).

3.1 Get MGS' games

- Implement the interface *GetGamesCallback*

```
public class MyActivity implements GetGamesCallback
```

- Request the configured games from the MGS platform

```
HCNX.getInstance().getMGS().getGames(this);
```

- Get the games thanks to the dedicated callback **onGamesReceived**

```
@Override
public void onGamesReceived(ArrayList<MgsGame> arrayList, String hnid, Integer
errorCode) {
...
}
```

It's possible to get some games in the list with a date in the past. If you don't want to display them, it's your call to hide them.

3.2 Display the rules of a game

Display the rules of a game in a webview

```
HCNX.getInstance().getMGS().showRules(getActivity(), mgsGame);
```

Parameters :

mgsGame : (an object of the mgsGames list) the current game

3.3 Play a game

Launch the flow of a game :

```
HCNX.getInstance().getMGS().play(getActivity(), mgsGame);
```

```
or HCNX.getInstance().getMGS().play(getActivity(), mgsGame,smsContent_answer );
```

Parameters :

mgsGame : (an object of the mgsGames list) the current game

smsContent_answer : may be null if there isn't an answer to the current game otherwise it matches to an answer choose by the user in the list of the answers of sms Content of the current mgsGame.

3.4 Object description

MGSGame :

Type	Name	Description	Visibility on the application	Field filled on MGSGame object
String	pictoAsteriskUrl	url of asterisk image	mandatory	optional
String	pictoPayment	url of picto payment image	mandatory	optional
String	price	price of the game	mandatory	optional
String	pdfGameRules	url of the game cgu, a button have to be displayed to be able to call show rules for display it	mandatory	optional
String	bannerUrl	url of the main image displayed as a banner	optional	optional
String	gameFormat	one of sms, form, audiotel	optional	mandatory
String	title	title of the game	optional	optional
String	startDate	game start date	optional	mandatory
String	endDate	game end date	optional	mandatory
SmsContent	smsContent	content of the sms	optional	optional
String	shortcode	sms number, can be displayed for sms game	optional	optional
String	phoneNumber	phone number, can be displayed for audiotel game	optional	optional

Integer	mgsId	id of the game Not unique. To have a unique ID for each game, combine mgs_id and gameType		mandatory
Integer	position	position in the list		optional
String	pictoGameTypeUrl	not used		
Integer	highwinId	not used		
String	gameType	not used		
String	paymentMethod	not used		
String	subtitle	not used		
String	cgu	not used		
String	formId	not used		
String	url	not used		

SMSContent :

Type	Name	Description	Visibility on the application	Field filled on MGSGame object
String	message	message to send	mandatory	optional
List<String>	answers	If you want to display 2 buttons instead of the “participate” button (for example for a quiz game) you can add this function. It’s adapted for cinematics with 2 SMS sent instead of 3 (in the 1st SMS it will pass the SHORTCODE + the answer corresponding to to the button selected) answers of the game.	optional	optional

3.5 Display sample

3.6 Empty State

For a better user experience, we recommend you to have an empty state in the games list.

4. Hello Push Notification Solution

Segments (or channels) are used to categorize the pushes, and can be used, if you wish, to let your users manage which push they wish to receive or not.

4.1 Get segments

Get your segments (Channels) configured from the Hello! platform to manage your populations Opt'in / Opt'out

- Implement the interface GetSegmentsCallback

```
public class MyActivity implements GetSegmentsCallback
```

- Request the configured segments from the Hello plateform

```
HCNX.getInstance().getHello().getSegments(this);
```

- Get the segments thanks to the dedicated callback **onSegmentsReceived**

```
@Override
public void onSegmentsReceived(ArrayList<HNSegment> segments, String hnid, Integer
errorCode) {
...
}
```

4.2 Manage segments

The manageSegments method is used to let the user enable or disable pushs for each segment.

- Implement the interface **HelloCallback**

```
public class MyActivity implements HelloCallback
```

- Send your configuration with one of this method :
  - `HCNX.getInstance().getHello().manageSegments(this,ArrayList<HNSegment> subCodes,ArrayList<HNSegment> unsubCodes);`
  - `HCNX.getInstance().getHello().manageSegments(this,String subCodesConcat,String unsubCodesConcat); (separator : ';')`
  - `HCNX.getInstance().getHello().manageSegments(this,ArrayList<String> subCodesConcat,ArrayList<String> unsubCodesConcat); (list of the String code)`

- Get the response thanks to the dedicated callback **onRequestFinished**

```
@Override
public void onRequestFinished(boolean success, String hnid, Integer errorCode) {
...
}
```

4.3 Get Campaigns in app

- Implement the interface **GetCampaignsCallback**

```
public class MyActivity implements GetCampaignsCallback
```

- Request the configured campaigns from the Hello plateform

```
HCNX.getInstance().getHello().getCampaignsInApp(this);
```

- Get the campaigns thanks to the dedicated callback **onSegmentsReceived**

```
@Override
public void onCampagnsReceived(ArrayList<HNCampaign> campaigns, String hnid,
Integer errorCode) {
...
}
```

4.4 Object description

HNSegment object :

Type	Name	Description
String	name	name of the segment
String	code	unique code describing the segment
String	category	category of the segments
String	color	segment color, if defined
int	categoryOrder	Order of the category
int	segmentOrder	order of the segment in the category
Boolean	subscribed	true if user is subscribed to this segment
Boolean	isDefault	true if segment is enabled by default
Boolean	isVisible	true if segment should be visible in app

HNCampaign object :

Type	Name	Description
int	id	id of the campaign
String	title	title of the campaign
String	message	message of the campaign

int	actionType	type of action
String	smsTo	sms number, if needed
String	smsText	text for the sms, if needed

4.5 Version

Your SplashscreenActivity (for example) can extend SynchroneUpdateActivity and do a redirection on the startApp method.

5. One Time Password solution

OTP sdk can be used to check the phone number validity or to add a security layer to your app.

5.1 Implement the interface GenerationCallback

```
public class MyActivity implements GenerationCallback, AuthenticationCallback
```

5.2 Generate password

```
HCNX.getInstance().getOTP().generate(String phoneNumber,this);
```

A sms with a password is sent to the user.

5.3 Authenticate

```
HCNX.getInstance().getOTP().authenticate(String password,this);
```

6. Testing Integration

In order to be able to test MGS and HELLO, you must call enableTestEnvironment method after calling the configure method.

By activating the test environment, the getGame method of MGS will return an example of games and Hello will display your last Push, your hn_id, token and hn_code.

```
HCNX.getInstance().enableTestEnvironment();
```

WARNING : these function must be deleted on the build you deliver to High Connexion tests.
WARNING : This method must not be called in a build for playstore publication