# Utilisation de GCM (Google Cloud Messaging) pour Android

Ce TP est inspiré de :
`http://android.amolgupta.in/2012/07/google-cloud-messaging-gcm-tutorial.html`

On va écrire deux parties.
Une application Android reçoit des messages envoyés par une application Java. Cette application Android doit s'enregistrer (en utilisant le project number du projet GCM) auprès de GCM pour recevoir un regId. Ce regId sera passé à l'application (serveur) Java. Lorsque le smartphone reçoit le message, celui est affiché dans ses notifications. C'est la première partie.
Une application Java (serveur) envoie des messages à des smartphones en utilisant GCM. Pour cela elle utilise l'API GCM, et le(s) regId(s) du (des) smarphone(s) pour l'envoi des messages. Elle aura donc besoin de cet (ces) regId(s) et de l'API Key du projet GCM pour pouvoir utiliser GCM. Ce sera la seconde partie.

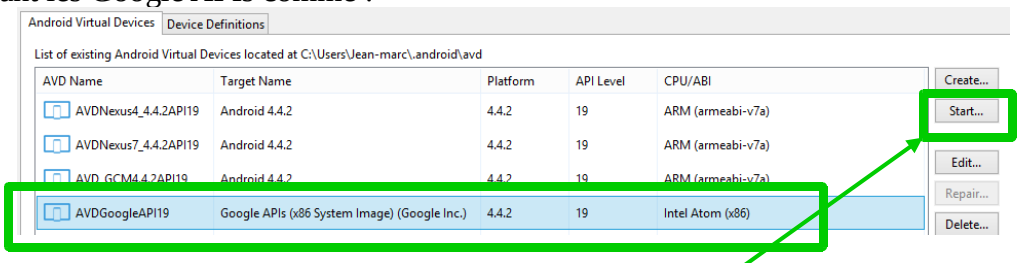
Finalement :

- l'application Android utilise le project number du projet GCM et récupère dynamiquement un regId fabriqué par GCM,
- l'application Java a besoin de l'API Key du projet GCM et du regId du smartphone

Avant de commencer récupérer l'archive `ssoIoT2014.zip` contenant la trame des deux applications à l'URL :
`http://cedric.cnam.fr/~farinone/SSOIoT/ssoIoT2014.zip`

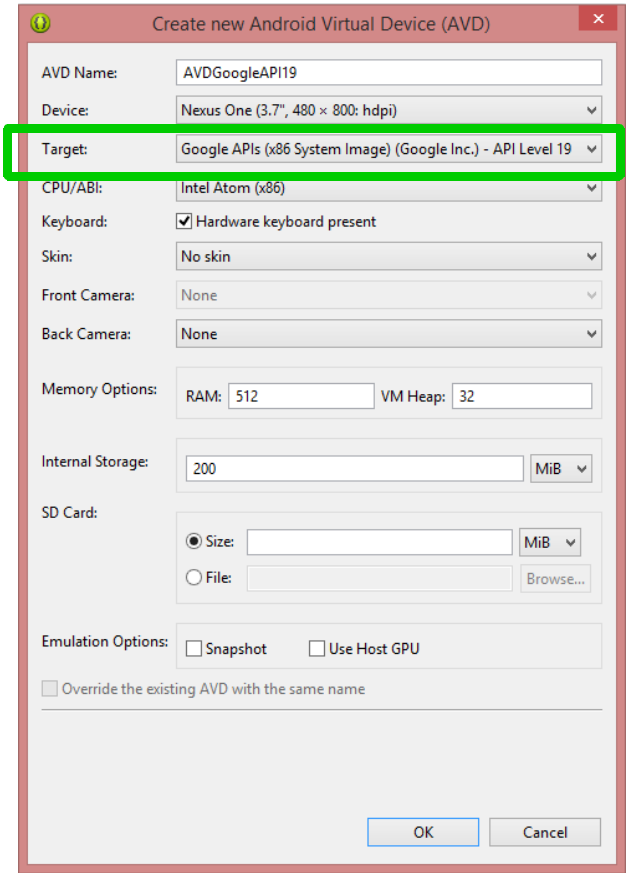
## Partie préliminaire éventuelle : Création d'un AVD lisant GCM

Si vous n'avez pas de véritable smartphone, ce qui se conçoit, l'environnement de développement Android vous en propose et même vous propose d'en créer.
Dans Eclipse Android, cliquer Window | Android Virtual Device Manager. Apparaît alors la liste des émulateurs de smartphones. Ce sont les AVD (Android Virtual Device). Vérifier que vous en avez un possédant les Google APIs comme :

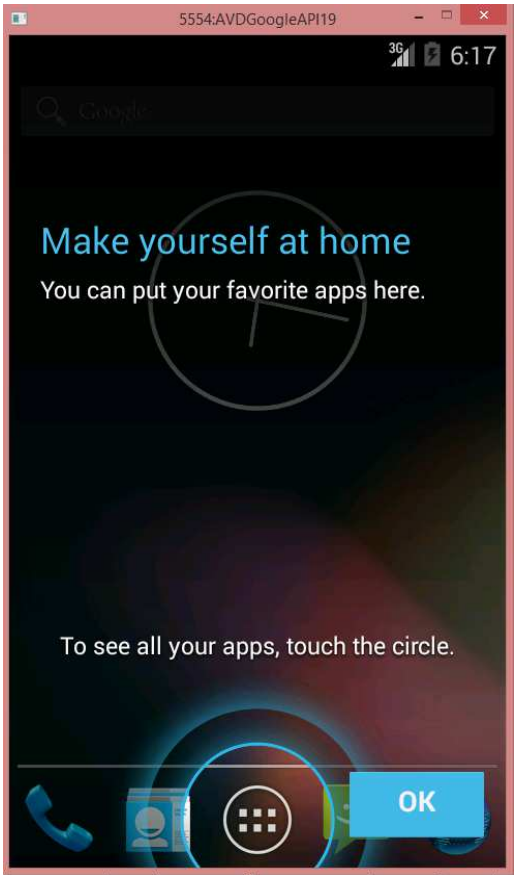



Si c'est le cas, sélectionner le et lancer le : cliquer sur Start...

Si ce n'est pas le cas, il faut en créer un . Cliquer le bouton Create... (au dessus de Start...), puis créer un AVD comme ce dessous. Il est important que votre AVD possède les Google APIs, bibliothèque nécessaire et supplémentaire aux bibliothèques standards Android pour utiliser GCM.




Lancer cet AVD (bouton Start... dans la fenêtre précédente) : euh cela peut prendre du temps (3 ou 4 minutes ?) avant d'obtenir :



# Première partie :
# Création d'une application Android utilisant le GCM

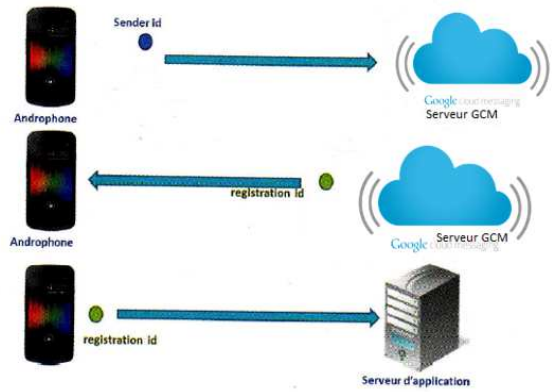
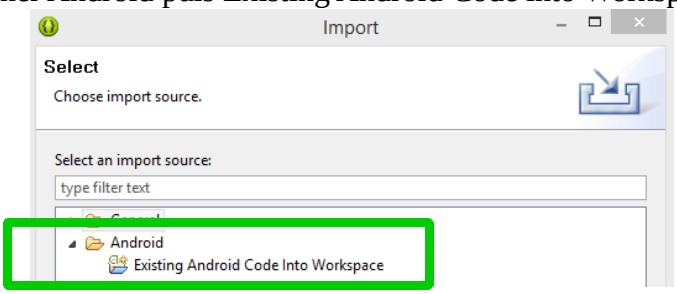



Figure 1 : récupération du regId

On code les 3 étapes indiquées par la figure 1.

### Etape 1 : importer le projet ReceiveFromCloudInAndroidDeviceProject

Ouvrer l'archive `ssoIoT2014.zip`. Il y a deux répertoires correspondant à 2 projets.
Lancer Eclipse for Android sur votre bureau. Importer le projet ReceiveFromCloudInAndroidDeviceProject comme un projet Android par File | Import. Dans la fenêtre Import sélectionner Android puis Existing Android Code into Workspace




Dans la fenêtre Import Projects, sélectionner parcourir (Browse), jusqu'au répertoire où se trouvent les projets à importer (répertoire ADistribuer). Cliquer OK.
Sélectionner le projet à importer, ReceiveFromCloudInAndroidDeviceProject, et COCHER LA CASE "Copy projects into workspace"
Cliquez Finish.

### Etape 2 : compléter le code

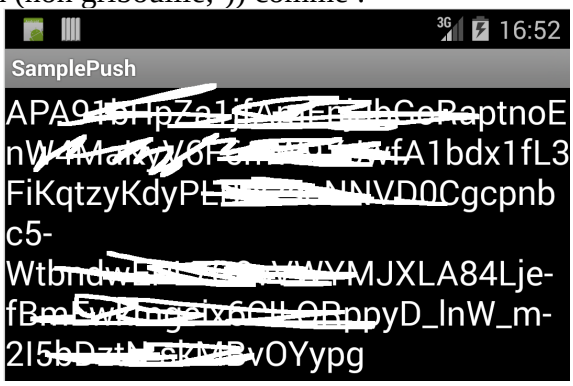
Dans l'activité SamplePushActivity, rechercher la ligne // TODO. La ligne suivante doit être complétée. S'aider du commentaire mis avant la classe.

De même dans la classe GCMIntentService, rechercher la ligne // TODO. La ligne suivante doit être complétée. S'aider du commentaire mis avant la classe.

En résumé, vous avez utiliser (2 fois) pour cette application cliente Android, le project number du projet GCM construit dans la Google Developer Console.

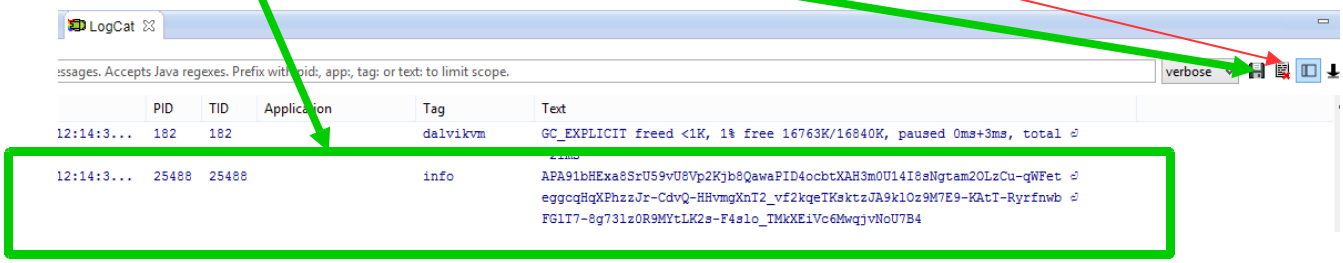
Lancer cette application Android. Pour cela sélectionner le projet dans éclipse, clic droit  et, dans le menu contextuel Run As | 2 Android Application.

Vous devez obtenir un écran (non gribouillé;-)) comme :




C'est le regId obtenu par l'architecture GCM qui doit être passé à l'émetteur (= le serveur Java SE). Il se trouve qu'on l'obtient aussi dans l'onglet LogCat.

Lorsque vous lancer l'application Android, vous devez obtenir le regId dans l'onglet LogCat. Sélectionner la ligne puis cliquer sur la disquette, là, pas là ! (n'est pas Jean-marc !)



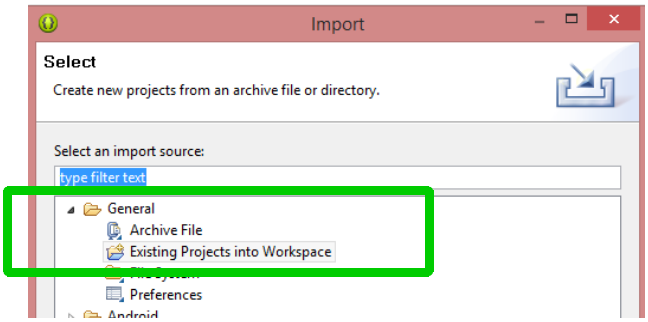

Sauvegarder la ligne dans un fichier texte (`log.txt`). Il est utile pour la seconde partie.

# Seconde partie :
# Une application Java SE émettrice dans le cloud vers des smartphones

### Etape 1 : importer le projet  PushIntoTheCloudProject

Dans votre Eclipse, il faut importer le projet PushIntoTheCloudProject. Attention c'est un projet Java SE et on ne l'importe pas comme un projet Android.

Cliquer File | Import. Dans la fenêtre Import sélectionner General (et pas Android) puis Existing Projects into Workspace




Dans la fenêtre Import Projects, sélectionner parcourir (Browse), jusqu'au répertoire où se trouvent les projets à importer (répertoire ADistribuer). Cliquer OK.
Sélectionner le projet à importer, PushIntoTheCloudProject,  et COCHER LA CASE "Copy projects into workspace"
Cliquez Finish.

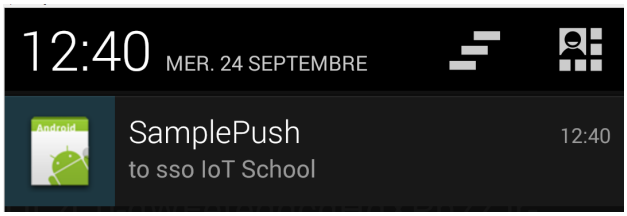
### Etape 2 : compléter le code

Dans la classe `EmetteurDansLeCloud`, rechercher les lignes // TODO. Les lignes suivantes ces TODO doivent être complétée. S'aider du commentaire mis avant la classe.
Rappel : le regId du smartphone à atteindre est dans le fichier log.txt construit dans la partie précédente. Reporter cette valeur dans l'application émettrice.

En résumé, vous avez utiliser pour cette application serveur, l'API Key du projet GCM construit dans la Google Developer Console et le regId des smarphones à atteindre obtenu dynamiquement par le smartphone du GCM.

Lancer l'application émettrice.

On doit obtenir (au bout de une minute à 2 minutes ?), dans le smartphone, une notification :




## Conclusion

Vous avez bien créée :

- une application Java SE envoie un message dans le cloud Google, message à destination de smartphones
- une application Android déployée sur un (ou des) smartphone(s), qui reçoit ces messages et les traitent (les affichent)

## Bonus

Euh, dans notre exemple, si l'application émettrice veut contacter un autre smartphone, il faut ajouter dans son code le regId de ce nouveau smartphone, compiler et exécuter cette application. Tout en ayant récupéré le regId du nouveau smartphone (par copier coller du LogCat, etc.)

Il suffirait d'avoir une application centralisatrice :

- à laquelle l'application Android se connecte lorsqu'elle a son regId pour le lui passer
- à laquelle on peut se connecter pour demander à diffuser un message aux smartphones concernés

Une servlet accessible par l'application Android et un navigateur web, répond à ces besoins. C'est une technologie Java Entreprise Edition (Java EE) facile mais hors sujet pour cette présentation. Mais, en tout cas, un bon bonus;-)