**출처:** **http://www.mimul.com/pebble/default/2012/09/01/1346492216891.html**

**http://developer.android.com/google/gcm/gcm.html**

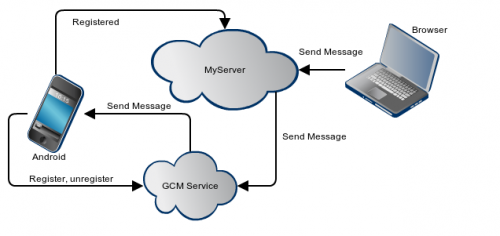
[**Android GCM 활용해서 푸시 메세지 전송기**](http://www.mimul.com:80/pebble/default/2012/09/01/1346492216891.html)

**GCM의 특징**

GCM의 특징은 아래와 같다.

* Android 애플리케이션에 메시지를 보낼 수있는 3'rd Party 애플리케이션 서버를 허용한다.
* GCM은 메시지의 전달과 순서를 보장하지 않는다.
* 메시지를 수신하기 위해 Android 디바이스의 Android 애플리케이션이 실행되고 있을 필요는 없다. 애플리케이션이 적절한 broadcast receiver와 권한을 설정하는 경우, 메시지가 도착했을 때, 시스템은 Intent broadcast가 Android 앱을 깨운다.
* 메시지 데이터를 위해 내장 사용자 인터페이스 및 기타 다른 처리는 제공하지 않고 있다. GCM은 단순히 원시 메시지 데이터를 그대로 Android 앱에 전달하고, 메시지의 처리는 앱이 완벽하게 제어 할 수 있다. 예를 들어 애플리케이션은 노티를 게시하거나, 고객 지정 사용자 인터페이스를 표시하거나, 백엔드로 데이터를 동기화 할 수도 있다.
* Android 2.2 이상을 실행하고, Google Play Store 애플리케이션이 설치되어 있는 디바이스 또는, Google APIs를 가진 Android 2.2을 실행하고 있는 에뮬레이터가 필요하다. 그러나, Google Play Store를 통해 Android 애플리케이션을 배포하는데 제한은 없다.
* Google 서비스에 대한 기존 커넥션을 사용한다. 이것은 3.0 이전 장치에서 사용자가 모바일 디바이스의 Google 계정 설정이 필요하다. Android 4.0.4 이상을 실행하는 장치에서 Google 계정은 필요 없다.

**아키텍처**

  
  
**1. 구성 요소(GCM에 영향을 주는 물리적인 요소)**

* 모바일 디바이스 : GCM을 사용하는 Android 애플리케이션을 실행하는 장치(디바이스). Google Play Store가 설치되고 2.2 이상의 Android 디바이스여야 하며, 해당 디바이스가 Android 4.0.4이하 버전을 실행하는 경우, 하나의 Google 계정이 로그인되어 있어야 한다. 또는 테스트를 위해 Google APIs를 사용한 Android 2.2을 실행하는 에뮬레이터를 사용해야 한다.
* 3'rd Party 애플리케이션 서버 : 개발자가 애플리케이션내에 GCM 기능을 구현한 애플리케이션 서버. 3rd Party 애플리케이션 서버는 GCM 서버를 경유해 디바이스의 Android앱에 데이터를 전송한다.
* GCM 서버 : 3rd Party 애플리케이션 서버에서 메시지 수신 및 디바이스의 메시지 전송에 관련된 Google 서버.

**2. 인증 정보(GCM 처리에 필요한 정보)**

* Sender ID : API 콘솔로부터 획득한 프로젝트 ID. Sender ID는 모바일 디바이스로 메시지 전송이 허용된 Android 앱을 식별하기 위한 등록 처리에 사용된다.
* Application ID : 메시지를 수신하기 위해 등록 할 Android 앱. Android 앱은 manifest가 제공하는 패키지 이름으로 식별된다. 그러면 메시지가 적절한 Android 앱을 타갯팅하는 것이 보장된다.
* Registration ID : 메시지를 수신 할 수 있도록 허용된 Android 앱을 위한, GCM에서 발급한 ID. Android 앱이 Registration ID를 갖고 있으며, 그것을 3rd Party 애플리케이션 서버로 전송한다. 애플리케이션 서버는 그것을 사용하여 Android 애플리케이션이 메시지를 받기 위해 등록된 각 디바이스를 식별하는데 사용한다. 즉, Registration ID는 특정 디바이스에서 실행중인 특정 Android 앱에 연결된다.
* Google 사용자 계정 : 디바이스가 Android 4.0.4 이하의 버전인 경우 GCM을 동작시키기 위해 모바일 디바이스는 적어도 하나의 Google 계정이 있어야 한다.
* Sender 인증 토큰 : Google 서비스에 대한 액세스 권한을 부여하는, 3'rd Party 애플리케이션 서버에 저장되는 API 키. API 키는 메시지를 보내는 POST 요청 헤더에 포함된다.

**GCM 라이프 사이클**

**1. GCM 활성화**  
모바일 디바이스에서 실행되는 앱이 푸시 메시지를 수신하기 위해서 등록하는 과정에서 발생된다. 그 등록 과정은 아래와 같다.

* GCM 서버에 registration Intent를 발송한다. registration Intent(com.google.android.c2dm.intent.REGISTER)는 Sender ID와 안드로이드 애플리케이션 ID를 포함한다. 이 등록 프로세스는 라이프 사이클 메소드가 없기 때문에 미등록 된 경우만 등록을 해야하고, 등록 부분은 onCreate()함수 안에서 처리하면 된다.
* 등록이 성공하면 GCM서버에서 리턴 값으로 안드로이드 애플리케이션에 대한 Registration ID를 부여 받는다. Google은 이 Registration ID를 정기적으로 갱신할 수 있어서 안드로이드 애플리켄이션에서는 이점을 잘 이해해 설계를 해야한다.
* 등록을 완료하기 위해서 안드로이드 애플리케이션은 Registration ID를 애플리케이션 서버로 전송한다. 그리고 Registration ID는 일반적으로 데이터 베이스에 저장된다. 이 Registration ID는 안드로이드 애플리케이션에서 unregisters 하거나 구글이 새로 수정될때까지 애플리케이션 서버와 안드로이드 앱에서 유지된다. 나중에 이 Registration ID로 메세지를 발송할 수 있게 된다.

주의: 사용자가 애플리케이션을 제거할 때 Registration ID는 GCM상에서 자동으로 등록이 해제되지는 않는다. 등록이 해제되는 경우는 GCM 서버가 모바일 디바이스에 메시지 전송을 시도하고 애플리케이션이 제거되었다는 응답을 받을 때 뿐이다. 이때 여러분의 서버는 등록이 해제된 것으로 모바일 디바이스에 기록해야 한다. (그 때 서버는 NotRegistered 오류를 받게 된다.)  
  
GCM 서버에서 Registration ID 삭제를 완료하는 데에는 몇 분이 걸릴 수 있다는 점에 유의하라. 즉, 3'd Party 애플리케이션 서버가 이전에 메시지를 보내면 메시지가 모바일 디바이스로 전달되지 않는 경우에도 3'd Party 애플리케이션 서버는 유효한 메시지 ID를 얻을 수도 있다.  
  
**2. 메세지 전송**  
메세지 전송을 위해서는 Registration ID와 API KEY가 필요하다. API 키 취득 방법은 아래 안드로이드 앱 만들기에서 참조하면 된다.  
애플리케이션 서버가 Android 앱에 메시지를 전송하기 위해 다음을 갖추고 있어야 한다.

* Android 앱은 특정 디바이스에 메시지를 받을 수 있도록 registration ID를 가지고 있어야 한다.
* 3'd Party 애플리케이션 서버가 registration ID를 저장한다.
* API 키. 이것은 개발자가 Android 애플리케이션을 위한 애플리케이션 서버에서 이미 설치가 완료되지 않으면 안된다. (자세한 사항은 3'd Party 애플리케이션 서버의 역할을 참조). API 키는 모바일 디바이스에 메시지를 보내기 위해 사용된다.

다음은 3'rd Party 애플리케이션 서버가 메세지를 보낼 때 발생하는 이벤트의 순서이다.

1. 애플리케이션 서버가 GCM 서버에 메시지를 보낸다.
2. 모바일 디바이스가 오프라인 상태인 경우에 대비해 Google이 메시지를 저장하고 대기열에도 저장한다.
3. 모바일 디바이스가 온라인 상태가 되면, Google이 해당 모바일 디바이스에 메시지를 보낸다.
4. 모바일 디바이스는 적절한 권한을 가진 안드로이드 애플리케이션의 Intent broadcast를 사용해서 메세지를 브로드캐스트함으로써 대상 애플리케이션만 메세지를 받을 수 있다. 그러면 안드로이드 애플리케이션이 깨어나게 된다. 즉, 안드로이드 애플리케이션은 메세지를 수신하기 전에 이미 실행되어 있을 필요는 없다.
5. 안드로이드 애플리케이션은 메세지를 처리한다. 안드로이드 애플리케이션이 중요한 작업을 수행하는 경우, PowerManager.WakeLock을 차단하고 서비스에서 모든 처리를 하는 것이 좋다.

**3. 메세지 수신**  
다음은 모바일 디바이스에 설치된 Android 애플리케이션이 메시지를 받을 때 발생하는 이벤트의 순서이다.

1. 시스템은 들어온 메시지를 수신 메시지의 페이로드에서 원시 키/밸류 쌍을 추출한다.
2. 시스템은 키/밸류 쌍을 com.google.android.c2dm.intent.RECEIVE Intent를 엑스트라의 세트로 대상 Android 애플리케이션에 전달한다.
3. Android 애플리케이션은 com.google.android.c2dm.intent.RECEIVE Intent에서 키의 원시 데이터를 추출하고 데이터를 처리한다.

**Android 애플리케이션 개발**

**1. Sender ID 취득**  
먼저, [Google APIs Console](https://code.google.com/apis/console/)에 방문한다. API Project가 없는 경우는 Create project.. 하면 프로젝트를 만들면 주소표시줄에 다음과 같은 URL이 나타난다.

https://code.google.com/apis/console/#project:243454537157

#project: 다음의 숫자를 기록한다. 이 ID가 C2DM의 Sender ID 에 해당한다.  
  
그리고 GCM 서비스(Services>)를 ON한다.  
  
 - 좌측 메뉴에서 Services를 선택  
 - Google Cloud Messaging for Android 토글을 ON  
  
**2. API Key(= Authorization Key)를 취득**  
C2DM 때 사용하던 Authorization Key 여기서도 만든다.  
  
 - APIs Console 에서 API Access 를 선택하고 Create New Server Key.  
 - IP 제한 필요 없으면 그냥 Create.  
 - Key for server apps(with IP locking)의 API key를 카피.  
  
이제 Sender ID 와 Authorization Key 만들기는 끝났다. 다음으로 안드로이드 앱 개발을 소개한다.  
  
**3. Android App 개발**  
**1) GCM Helper Libraries 설치**  
Android SDK Manager를 클릭하면 Extra>Google Cloud Messaging for Android Library를 설치한다.  
  
**2) GCMIntentService**

public GCMIntentService() {

super(SENDER\_ID);//project ID

if (BuildConfig.DEBUG)

Log.d(LOG\_TAG, "[GCMIntentService] start");

}

protected void onRegistered(Context context, String registrationId) {

Log.v(LOG\_TAG, "onRegistered-registrationId = " + registrationId);

// 디바이스 등록 아이디 등록 처리 로직 개발

// 3rd Party 애플리케이션 서버와 통신함

....

}

@Override

protected void onUnregistered(Context context, String registrationId) {

// 디바이스 토큰 제거 성골의 동작 설명

if (BuildConfig.DEBUG)

Log.d(LOG\_TAG, "onUnregistered-registrationId = " + registrationId);

// 디바이스 등록 아이디 해제 처리 로직 개발

// 3rd Party 애플리케이션 서버와 통신함

....

}

@Override

protected void onMessage(Context context, Intent intent) {

// 메세지를 수신했을 때 동작 설명

if (BuildConfig.DEBUG)

Log.d(LOG\_TAG, "GCMReceiver Message");

try {

String title = intent.getStringExtra(PUSH\_DATA\_TITLE);

String message = intent.getStringExtra(PUSH\_DATA\_MESSAGE);

Vibrator vibrator =

(Vibrator) context.getSystemService(Context.VIBRATOR\_SERVICE);

vibrator.vibrate(1000);

setNotification(context, title, message);

if (BuildConfig.DEBUG)

Log.d(LOG\_TAG, title + ":" + message);

} catch (Exception e) {

Log.e(LOG\_TAG, "[onMessage] Exception : " + e.getMessage());

}

}

@Override

protected void onDeletedMessages(Context context, int total) {

if (BuildConfig.DEBUG)

Log.d(LOG\_TAG, "onDeletedMessages");

// 메세지 삭제

}

@Override

public void onError(Context context, String errorId) {

if (BuildConfig.DEBUG)

Log.d(LOG\_TAG, "onError: " + errorId);

// 오류 발생 시 처리

}

@Override

protected boolean onRecoverableError(Context context, String errorId) {

if (BuildConfig.DEBUG)

Log.d(LOG\_TAG, "onRecoverableError: " + errorId);

return super.onRecoverableError(context, errorId);

}

private void setNotification(Context context, String title, String message) {

NotificationManager notificationManager = null;

Notification notification = null;

try {

notificationManager = (NotificationManager) context

.getSystemService(Context.NOTIFICATION\_SERVICE);

notification = new Notification(R.drawable.ic\_launcher,

message, System.currentTimeMillis());

Intent intent = new Intent(context, NotificationActivity.class);

PendingIntent pi = PendingIntent.getActivity(context, 0, intent, 0);

notification.setLatestEventInfo(context, title, message, pi);

notificationManager.notify(0, notification);

} catch (Exception e) {

Log.e(LOG\_TAG, "[setNotification] Exception : " + e.getMessage());

}

}

**3) Activity**

public class NotificationActivity extends Activity

{

private static final String LOG\_TAG = "GCMsample";

/\*\*

\* 화면 설정

\*/

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.main);

// 디바이스 GCM 사용 가능한지 확인

GCMRegistrar.checkDevice(this);

// 매니페스트 설정이 올바른지 확인

GCMRegistrar.checkManifest(this);

// 등록

registerToken();

}

/\*\*

\* GCM에 디바이스 토큰 등록

\*/

private void registerToken() {

// registration ID（디바이스 토큰) 취득하고 등록되지 않은 경우 GCM에 등록

final String regId = GCMRegistrar.getRegistrationId(this);

if ("".equals(regId)) {

GCMRegistrar.register(this, GCMIntentService.SENDER\_ID);

}

}

/\*\*

\* GCM에 디바이스토큰 삭제

\*/

private void unregisterToken() {

if (GCMRegistrar.isRegistered(this)) {

GCMRegistrar.unregister(this);

}

}

}

**4) Manifest 수정**  
GCM 은 Android 2.2 이상에서만 사용 가능하므로

<uses-sdk android:minSdkVersion="8" android:targetSdkVersion="16"/>

<!-- C2DM 권한 설정 -->

<permission android:name="com.mimul.pushsample.permission.C2D\_MESSAGE"

android:protectionLevel="signature" />

<uses-permission android:name="com.mimul.pushsample.permission.C2D\_MESSAGE" />

<!-- App receives GCM messages. -->

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

<!-- 인터넷 연결 권한 설정 -->

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

<!-- GCM requires a Google account. -->

<uses-permission android:name="android.permission.GET\_ACCOUNTS" />

<!-- Keeps the processor from sleeping when a message is received. -->

<uses-permission android:name="android.permission.WAKE\_LOCK" />

<uses-permission android:name="android.permission.READ\_PHONE\_STATE" />

<!-- application 안쪽 -->

<application

android:icon="@drawable/ic\_launcher"

android:label="@string/app\_name" android:debuggable="true">

<!-- 최초 활동 설정 -->

<activity

android:label="@string/app\_name"

android:name="com.mimul.pushsample.NotificationActivity" >

<intent-filter >

<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />

</intent-filter>

</activity>

<!-- GCM BroadcastReceiver 설정 -->

<receiver

android:name="com.google.android.gcm.GCMBroadcastReceiver"

android:permission="com.google.android.c2dm.permission.SEND" >

<intent-filter>

<!-- Receives the actual messages. -->

<action android:name="com.google.android.c2dm.intent.RECEIVE" />

<!-- Receives the registration id. -->

<action android:name="com.google.android.c2dm.intent.REGISTRATION" />

<category android:name="com.mimul.pushsample" />

</intent-filter>

</receiver>

<service android:name=".GCMIntentService" />

</application>

**3'rd Party 애플리케이션 서버 개발**

안드로이드 앱에서 Registration ID를 서버에서 받아 데이터 베이스에 저장된 registration\_id와 google console에서 등록한 API KEY를 가지고 푸시 메세지를 발송할 수 있다.  
그래서 3'rd Party 애플리케이션 서버는 첫번째로 registration\_id 값을 데이터 베이스에 저장하고 삭제하는 API가 필요하며, 두번째로 푸시 메세지 발송하는 기능이 필요하게 된다.  
  
**1. registration\_id 값 데이터 베이스 저장 API**  
registration\_id 등록하고 해제하는 서버 API를 개발해 구동해주면 된다. 그 내용은 아래와 같다.

* GCMIntentService.onRegistered()함수에서 호출하는 registration\_id 데이터 베이스에 저장하는 API.
* GCMIntentService.onUnRegistered()함수에서 호출하는 registration\_id 데이터 베이스에 삭제하는 API.

**2. GCM 서버로 메세지 전송**

public int sendGCM(String registrationId, String title, String message)

throws Exception

{

int result = 0;

Client client = null;

WebResource r = null;

ClientResponse response = null;

String textEntity = null;

try {

ClientConfig config = new DefaultClientConfig();

client = Client.create(config);

//Set the connection time out (milliseconds)

client.setConnectTimeout(5000);

//Set the read time out (in milliseconds)

client.setReadTimeout(5000);

r = client.resource(UriBuilder.fromUri(gcmUrl).build());

Form form = new Form();

form.add("registration\_id", registrationId);

form.add("collapse\_key", "update");

form.add("data.title", title);

form.add("data.message", message);

response = r.type(MediaType.APPLICATION\_FORM\_URLENCODED)

.header("Authorization", "key=" + gcmApiKey)

.accept(MediaType.APPLICATION\_JSON).post(ClientResponse.class, form);

int status = response.getStatus();

if (status == 200) {

textEntity = response.getEntity(String.class);

//status=200&response=id=0:1346144940651656%996d1f17ce0038c9

} else {

result = -2;

}

} catch (Exception e) {

result = -1;

logger.error("An error occured in {}", e);

} finally {

try {

if (response != null)

response.close();

if (client != null)

client.destroy();

} catch (Exception e) {

e.printStackTrace();

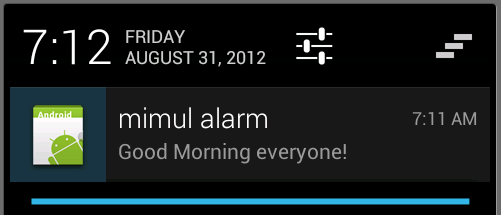
}

}

return result;

}

**푸시 메세지 테스트 결과**

sendGCM 함수를 호출해서 Google 서버로 메세지를 보내면 10초 정도 뒤에 안드로이드 앱에서 메세지를 전송받을 수 있다.  
  
  
  
**[참조 사이트]**

* [GCM Architectural Overview](http://developer.android.com/guide/google/gcm/gcm.html)
* [GCM: Getting Started](http://developer.android.com/guide/google/gcm/gs.html)