|  |
| --- |
| AVNC&IS事业部 |
| Google Cloud Messaging for Android |
| —— Google最新推送技术简介 |

|  |
| --- |
| 高大伟  2012/12/7 |

目录

[1 关于GCM 3](#_Toc342837630)

[1.1 GCM的主要特点 3](#_Toc342837631)

[1.2 GCM的组件 4](#_Toc342837632)

[1.3 GCM的通讯保障 5](#_Toc342837633)

[2 GCM的使用方法 6](#_Toc342837634)

[2.1 程序如何支持GCM 7](#_Toc342837635)

[2.2 GCM服务器如何配置 9](#_Toc342837636)

[2.3 第三方服务器如何配置 10](#_Toc342837637)

# 关于GCM

Google Cloud Messaging for Android (GCM)是一个允许你从服务器端发送数据到你的用户的移动设备客户端上的服务，也就是我们常说的数据推送服务。此服务公布于2012年6月27日。通过GCM服务，可向用户的移动设备客户端发送一个轻量级的消息，告诉它你的服务器端有新的数据要读取，比如你的朋友上传了一个新的电影。发送的数据也可以是附带多达4kb负载数据的消息。因此，像实时通讯这样的程序就可以直接使用收到的消息了。GCM负责消息的队列化，以及将消息发送到目标设备上的目标程序。GCM是完全免费的、没有数据流量限制。

## GCM的主要特点

1. GCM允许第三方的程序服务端发送消息到他们的Android设备。
2. 手机端的程序不需要一直运行来接收消息。当有新的消息到来时，系统会通过Intent广播来唤醒程序，只要程序设置了适当的广播接收器和相应的权限。
3. GCM不提供任何的内置的用户界面或者其他的消息处理。它只是简单地把收到的原始消息传递给程序，由程序决定如何处理数据。比如，程序可能发送一个通知，显示一个自定义的界面或者悄悄地同步数据。
4. GCM要求手机必须运行Android 2.2或者更高版本，并且安装了Google Play Store，或者运行具有Google APIs的Android 2.2模拟器。但是不仅限于通过Google Play Store部署你的程序。
5. GCM使用一个既有的，用于Google services的连接。对于3.0版本以前的设备，用户必须在他们的移动设备上设置他们的Google账户，对于Android 4.0.4或更高的版本就不用设置Google账户了。

## GCM的组件

GCM架构中的几大组件

1. 移动设备

运行着使用GCM的程序的设备

1. 第三方服务器

第三方服务器主要负责业务逻辑。在需要发送通知的时候、通过GCM服务器，推送数据到已经注册的移动设备上的程序。

1. GCM服务器

Google官方的服务器，负责收集从第三方服务器送达的消息、并将消息推送到目标设备上的目标程序。





Google Service Framework

第三方服务器

GCM服务器

应用程序

注册

注册

注册完成，返回RID

注册完成

使用RID注册

向RID代表的程序发送数据

推送数据

数据处理

图1.2.1 交互时序

程序通过Google Service Framework（GSF）向GCM服务器注册，表示程序想使用GCM服务来接收推送数据。注册完成后，GCM服务器返回给程序一个唯一的RID。程序再使用RID向第三方服务器注册，第三方服务器只会向已经注册过的程序推送数据。在有数据需要推送给特定程序时， 第三方服务器向GCM服务器发送要推送的数据，并附带相应的RID。

为了使用GCM的推送服务，除了编写可接收推送数据的程序，我们还要搭建自己的第三方服务器。好在Google提供了非常丰富的在线文档和Sample程序。

## GCM的通讯保障

移动设备客户端与GCM服务器之间采用的是安全的HTTPS协议。除此，还使用了以下凭证，确保数据被正确地发送到目标设备上的目标程序。

1. Sender ID (SID)

从Google的[API Console](https://code.google.com/apis/console" \t "_blank)获得的项目编号。

1. Application ID (AID)

定义在Manifest文件中的程序的Package名。程序的Package名是唯一的，因此确保了数据被发送到正确的程序。

1. Registration ID (RID)

由GCM服务器分配的唯一ID，用于标识特定设备上的特定程序。程序在取得RID后，将它转发到第三方服务器。第三方服务器通常要维护一个RID的列表，这样就可以知道哪些设备上的程序在等待接收推送数据了。

1. Google User Account (GUA)

如果Android的版本低于4.0.4，需要配置一个Google账户。

1. Sender Auth Token (SAT)

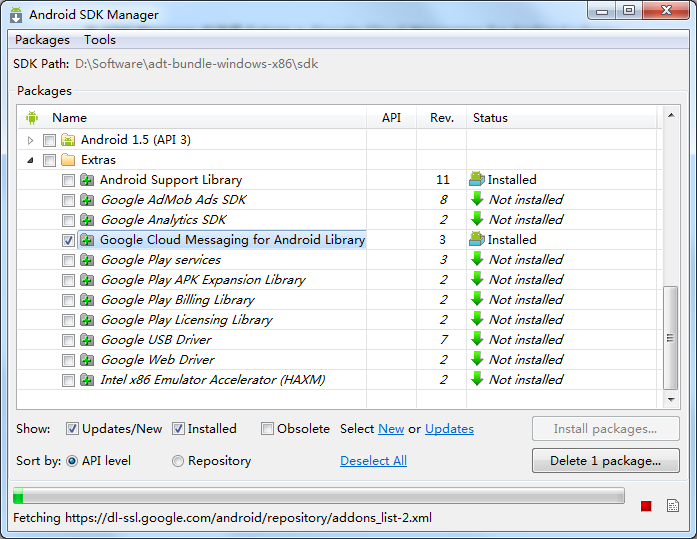
第三方服务器并不可以随便向GCM服务器发送数据，必须向。在向GCM服务器以POST的方式发送数据时，SAT被包含在POST请求的Header中。

# GCM的使用方法

## 安装GCM辅助Lib

为了使用GCM服务，我们并不需要从头一步一步地实现。Google为我们提供了服务器端和客户端的辅助Lib，这些Lib可大大降低使用门槛，而且可提高效率。

在SDK Manager中选择Extras > Google Cloud Messaging for Android Library：



安装后，在*YOUR\_SDK\_ROOT*/extras/google/gcm/目录下可看到以下目录：

1. gcm-client

客户端使用的Lib，提供了如GCMRegistrar这样的辅助类，简化客户端的注册过程。

1. gcm-server

第三方服务器使用的Lib，主要提供了向GCM服务器发送数据的方法。

1. samples

示例程序

## 程序如何支持GCM

首先将*YOUR\_SDK\_ROOT*/extras/google/gcm/gcm-client/dist/gcm.jar复制到工程的lib文件夹，然后：

1. 创建一个manifest文件。这个文件包含程序使用GCM需要使用的权限。



图2.1.1 manifest示例

1. 写java代码。要使用GCM，程序要包括：
   1. 注册和注销GCM服务的代码

客户端Activity的onCreate()方法中添加如下代码：

GCMRegistrar.*checkDevice*(**this**);

GCMRegistrar.*checkManifest*(**this**);

**final** String regId = GCMRegistrar.*getRegistrationId*(**this**);

**if** (regId.equals("")) {

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

} **else** {

Log.v(TAG, "Already registered");

}

在此省略了注销GCM服务的代码，详见Sample代码。

* 1. 处理来自GCM的Intent，Intent包含的数据可能是注册GCM服务的结果，也可能是推送过来的数据。需要注意的是，最好在一个IntentService中处理接收到的Intent。

**public** **class** MyIntentService **extends** IntentService {

**private** **static** PowerManager.WakeLock *sWakeLock*;

**private** **static** **final** Object *LOCK* = MyIntentService.**class**;

**static** **void** runIntentInService(Context context, Intent intent) {

**synchronized** (*LOCK*) {

**if** (*sWakeLock* == **null**) {

PowerManager pm = (PowerManager) context.

getSystemService(Context.POWER\_SERVICE);

*sWakeLock* = pm.newWakeLock(PowerManager.PARTIAL\_WAKE\_LOCK,

"my\_wakelock");

}

}

*sWakeLock*.acquire();

intent.setClassName(context, MyIntentService.**class**.getName());

context.startService(intent);

}

@Override

**public** **final** **void** onHandleIntent(Intent intent) {

**try** {

String action = intent.getAction();

**if**(action.equals("com.google.android.c2dm.intent

.REGISTRATION")) {

**handleRegistration**(intent);

} **else** **if** (action.equals("com.google.android.c2dm

.intent.RECEIVE")) {

**handleMessage**(intent);

}

} **finally** {

**synchronized** (*LOCK*) {

*sWakeLock*.release();

}

}

}

**private** **void** handleRegistration(Intent intent) {

String rid = intent.getStringExtra("registration\_id");

String error = intent.getStringExtra("error");

String unregistered = intent.getStringExtra("unregistered");

// registration succeeded

**if** (rid != **null**) {

// store registration ID on shared preferences

// notify 3rd-party server about the registered ID

}

// unregistration succeeded

**if** (unregistered != **null**) {

// get old registration ID from shared preferences

// notify 3rd-party server about the unregistered ID

}

// last operation (registration or unregistration) returned an error;

**if** (error != **null**) {

**if** ("SERVICE\_NOT\_AVAILABLE".equals(error)) {

// optionally retry using exponential back-off

// (see Advanced Topics)

} **else** {

// Unrecoverable error, log it

Log.i(TAG, "Received error: " + error);

}

}

}

**private** **void** handleMessage(Intent intent) {

// server sent 2 key-value pairs, score and time

String score = intent.getExtra("score");

String time = intent.getExtra("time");

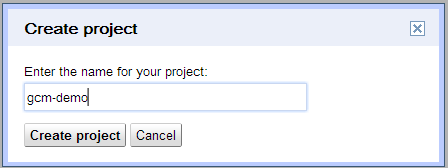
// generates a system notification to display the score and time

}

}

## GCM服务器如何配置

1. 打开[Google APIs Console page](https://code.google.com/apis/console" \t "_blank)，创建新的项目：



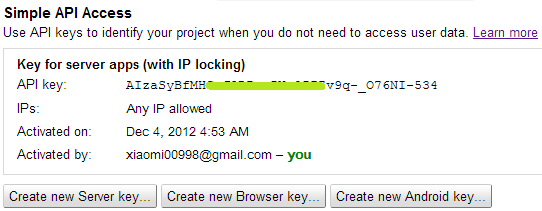
1. 打开新建项目的GCM服务

选择页面左边的Services，将GCM服务打开：



1. 取得API访问密码，即上述的SAT，供第三方服务器使用

选项页面左边的API Access：



## 第三方服务器如何配置

第三方服务器需要支持Servlet API 2.5，因此选择了Tomcat 6.0作为服务器。此外可利用SDK中提供的sample做为base进行开发。第三方服务器的sample的位置是：*YOUR\_SDK\_ROOT*/extras/google/gcm/samples/gcm-demo-server。唯一只需修改的是gcm-demo-server/WebContent/WEB-INF/classes/api.key这个文件，将上面取得的API访问密码粘贴到这个文件中，然后用下面的命令进行编译：

$ ant war