INTEGRACION DE LA LIBRERIA ANDROID DE MALCOM

INDICE

Malcom Ventures S.L. 2012

Requerimientos

- La librería de Android requiere una versión mínima 2.2, Froyo (API Level 8) para poder funcionar correctamente.
- El dispositivo ha de tener una cuenta configurada al menos y, en el caso del emulador, en la AVD tener las Google APIs y una cuenta configurada.

Librería Malcom

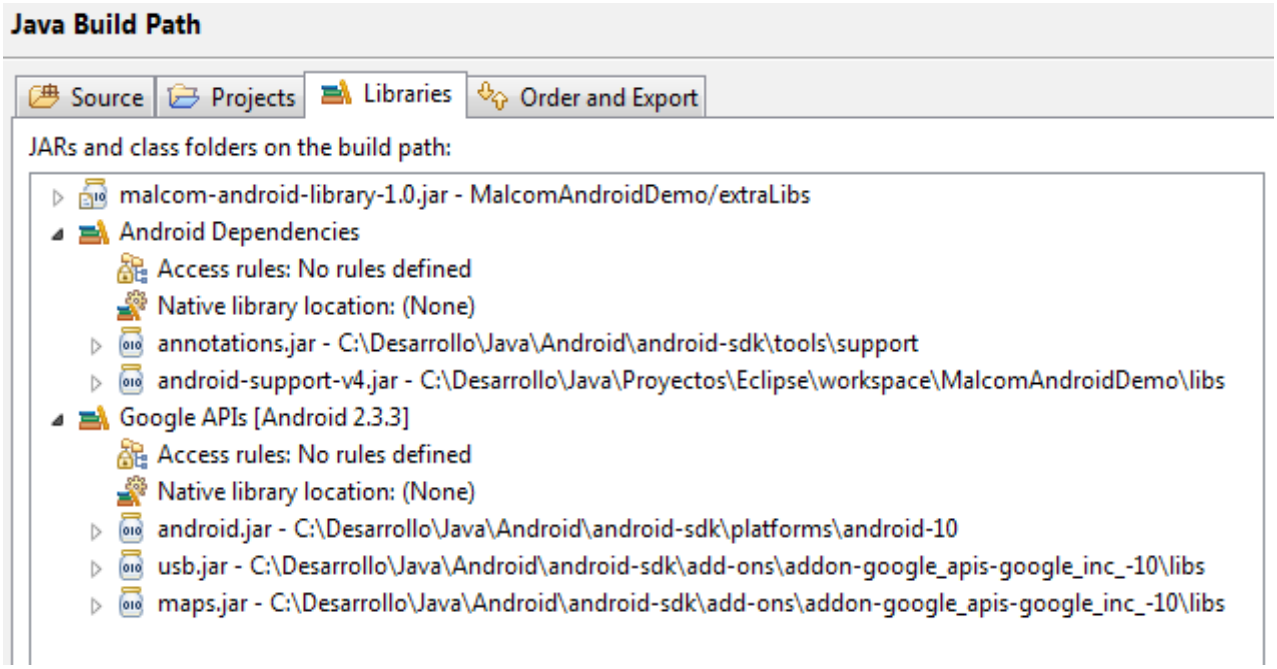
La librería Malcom para Android la conforman tres ficheros jar y una serie de recursos asociados a los diferentes módulos a usar (*se verá mas adelante*) según se usen unos u otros módulos:

- La librería en sí, "**malcom-android-library-<version>.jar**".
- El código fuente que permitirá debuggar, "malcom-android-library-<version>-sources.jar".
- El javadoc, "malcom-android-library-<version>-javadoc.jar"

Para incluirla en el proyecto tenemos dos opciones:

1. Incluir el fichero "malcom-android-library-<version>.jar" en la carpeta "libs".
2. Crear una carpeta llamada por ejemplo "malcomLib" y meter dentro de ella los tres jars de la librería de Malcom.

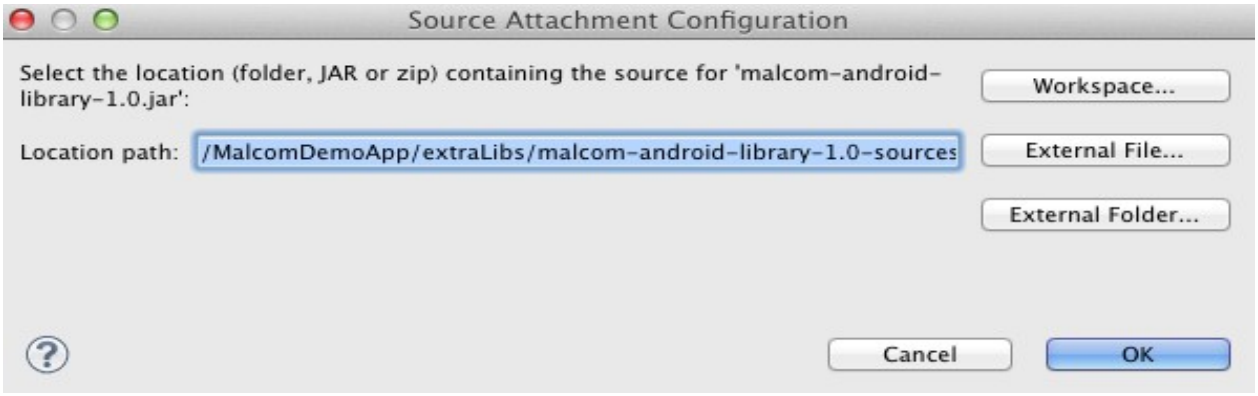
La opción 1 es la perfecta si el desarrollador de la app no desea debuggar por la librería de Malcom. La opción 2 es la recomendable si se desea poder debuggar dentro de la librería de Malcom. En este caso incluiremos manualmente al build-path del proyecto el jar "malcom-android-library-<version>.jar". Una vez hecho esto, tendremos algo tal que:



(*Aquí se ve que estan incluidas las librerías externas gcm.jar, gson-2.2.2-jar además de las propias de Android junto con la librería de Malcom recién añadida*).

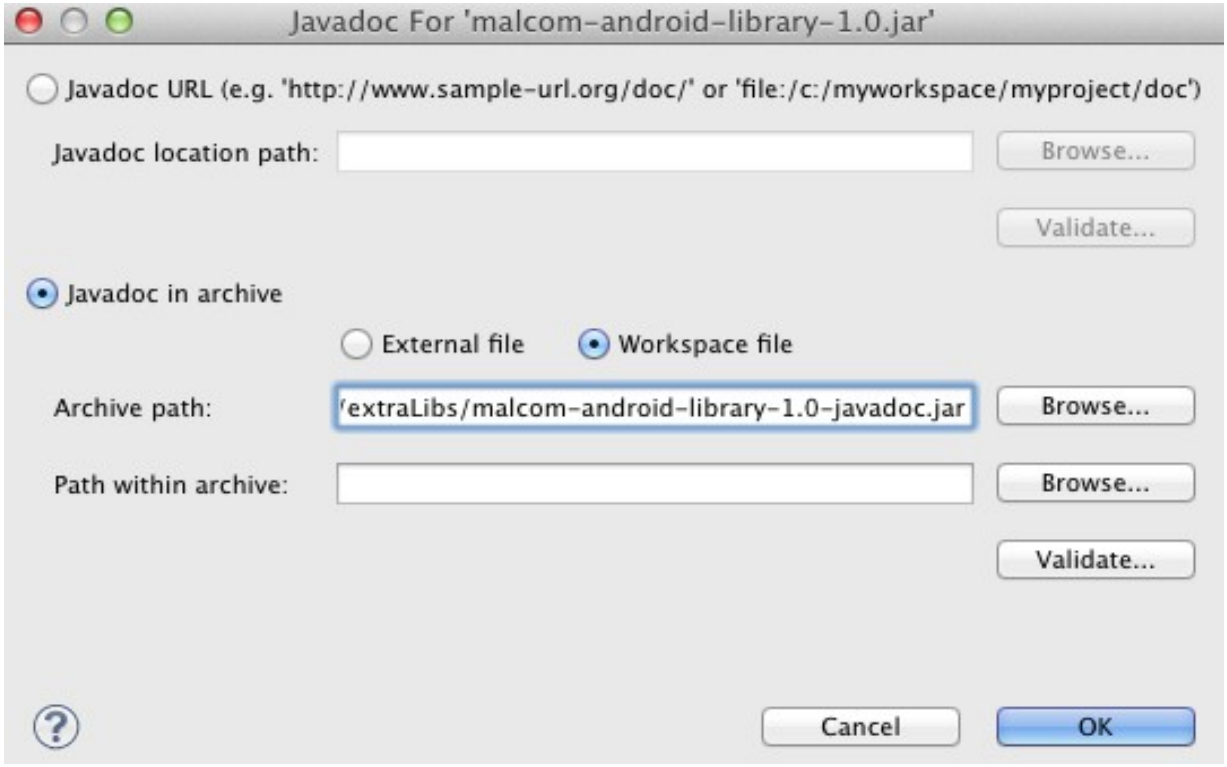
Ahora, le indicamos donde se encuentran los fuentes para asi poder debuggar y, el javadoc. Para

ello, hacemos clic en la librería de Malcom desplegándola y seleccionamos “Source Attachment”



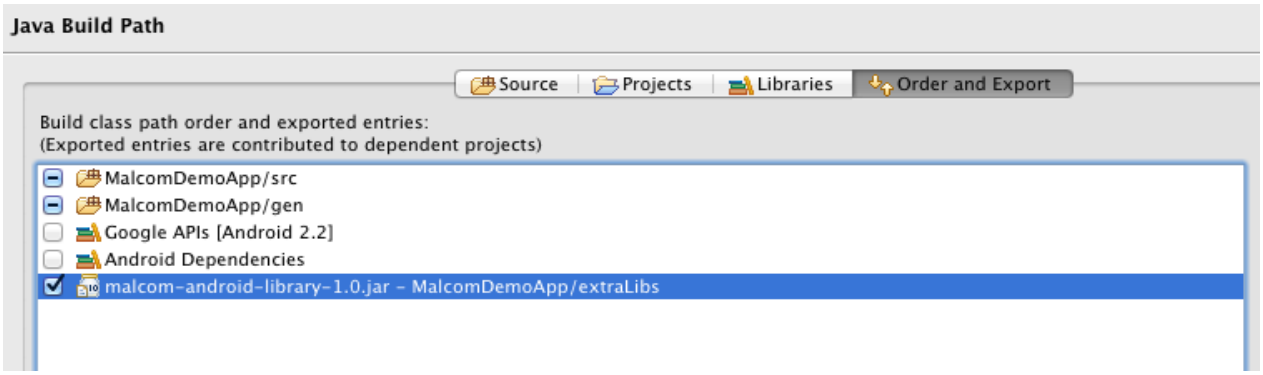
indicamos donde se encuentra el fichero “malcom-android-library-<version>-sources.jar” y damos a “ok”.

Para el javadoc seleccionamos “Javadoc location”:



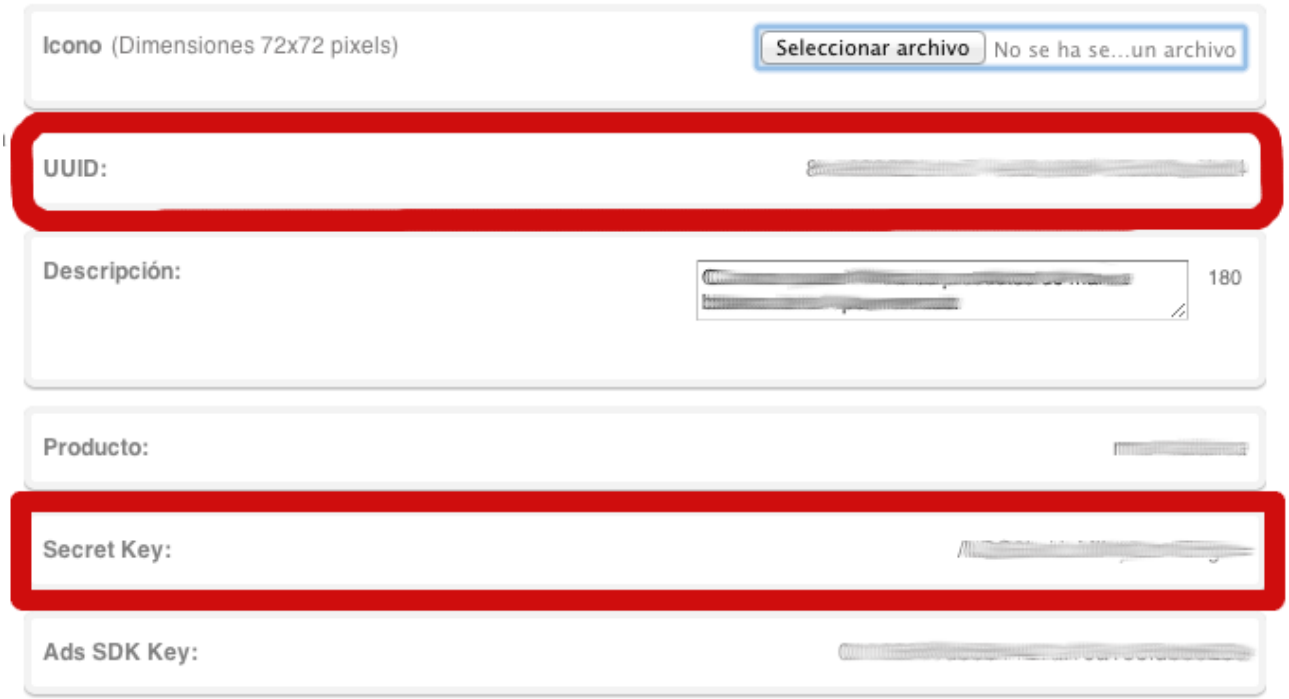
e indicamos la localización en el proyecto Android del fichero “malcom-android-library-<version>-javadoc.jar”. Damos a “Ok”.

Ya solo nos queda decirle al SDK que incluya esta librería en el APK de la aplicación. Para ello, nos vamos a la pestaña “Order and Export” y seleccionamos la librería y damos a “Ok”.



Datos necesarios para la librería

Aquí veremos donde obtener los diferentes datos que necesitamos para inicializar los diferentes módulos de la librería. Los valores los obtenemos desde Malcom (*haciendo clic en el botón de la caja de herramientas*)



Donde el "UUID" es el MalcomAppId y, la "Secret Key" es el valor de MalcomAppSecretKey.

El resto de los demás parámetros a configurar dependen de si usamos unos módulos ú otros.

Concretamente:

- Si usamos el módulo de notificaciones, tendremos que configurar:
  - **gcmSenderId (en esu correspondiente apartado se indica como añadirlo)**. Para usar el módulo de notificaciones se ha de crear un proyecto y activar para dicho proyecto GCM en el portal de Google APIs. Esto nos generará una key, concretamente el valor de

"#project" de la url para la aplicación configurada. Ver http://developer.android.com/guide/google/gcm/gs.html.

AndroidManifest.xml de la aplicación

Para que la librería pueda desarrollar todas sus funciones es necesario incluir unos permisos concretos en el fichero de configuración raiz de la aplicación Android.

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.GET_ACCOUNTS" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.VIBRATE" />
```

Uso de Módulos

Inicialización

En primer lugar hay que inicializar malcom de la siguiente forma:

```
MCMCoreAdapter.getInstance().initMalcom(this, UUID_MALCOM, SECRET KEY);
```

Módulo de Configuración

Este módulo permite ejecutar la configuración establecida en la sección "Configurar" dentro de Malcom.

Para usar este módulo hace falta:

- Copiar el fichero "mcmconfiginfo.json" al directorio "assets" de la aplicación Android.
- Renombrar el fichero de imagen splash que desee para el modo off-line a "splash.img" y ponerlo en la carpeta "assets" de la aplicación Android. *Este paso es opcional, en caso de no indicar alguno simplemente la splash no se mostrará si previamente no se pudo descargar la configuración*.

- En el activity principal, o en aquel que queramos mostrar los diferentes elementos de la configuración, se inicializa el módulo:

```
MCMCoreAdapter.getInstance().moduleConfigurationActivate(this);
```

De esta forma se carga la configuración y se muestran las alertas configuradas.

- Para mostrar la splash secundaria, en el layout correspondiente a ese activity hay que añadir lo siguiente:

```
<LinearLayout
	android:id="@+id/splash_layout"
	android:layout_width="fill_parent"
	android:layout_height="fill_parent"
	android:orientation="vertical"
	android:visibility="gone"
	android:background="@android:color/black"
	android:gravity="center_vertical">


	<ImageView android:id="@+id/image_view"
		android:layout_width="fill_parent"
		android:layout_height="fill_parent"
		android:visibility="gone"/>

	<LinearLayout
		android:id="@+id/splash_progresszone"
		android:layout_width="fill_parent"
		android:layout_height="wrap_content"
		android:orientation="horizontal"
		android:gravity="center_horizontal|center_vertical"
		android:visibility="gone">
		<ProgressBar
			android:id="@+id/splash_progress_bar"
			style="?android:attr/progressBarStyleSmall"
			android:layout_width="wrap_content"
			android:layout_height="wrap_content"
			android:padding="2dip"/>
		<TextView android:layout_height="wrap_content"
			android:layout_width="wrap_content"
			android:textColor="@android:color/white"
			android:text=""
			android:paddingLeft="4dp"/>
	</LinearLayout>

</LinearLayout>
```

- Para mostrar el intersitial:

```
<LinearLayout

	android:id="@+id/webview_layout"
	android:layout_width="fill_parent"
	android:layout_height="fill_parent"
	android:orientation="vertical"
	android:background="@android:color/black"
	android:visibility="gone">
```

```
<WebView
	android:id="@+id/webview"
	android:layout_width="fill_parent"
	android:layout_height="0dp"
	android:layout_weight="1"
/>

<LinearLayout
	android:id="@+id/progresszone"
	android:layout_width="fill_parent"
	android:layout_height="wrap_content"
	android:orientation="horizontal"
	android:gravity="center_horizontal"
	android:visibility="gone">
	<ProgressBar
		android:id="@+id/web_view_progress_bar"
		style="?android:attr/progressBarStyleSmall"
		android:layout_width="wrap_content"
		android:layout_height="wrap_content"
		android:padding="2dip"/>
	<TextView android:layout_height="wrap_content"
		android:layout_width="wrap_content"
		android:textColor="@android:color/white"
		android:text="loading..."
		android:paddingLeft="4dp"/>
</LinearLayout>
<LinearLayout
	android:layout_width="fill_parent"
	android:layout_height="wrap_content"
	android:orientation="horizontal"
>
	<Button android:id="@+id/web_view_close"
		android:layout_width="fill_parent"
		android:layout_height="fill_parent"
		android:text="Close" />
</LinearLayout>

</LinearLayout>
```

Para acceder a una propiedad de la configuración avanzada, ejecutaremos la linea siguiente:

```
MCMCoreAdapter.getInstance().moduleConfigurationGetProperty("<property>")
```

en donde "<property>" es la propiedad que establecimos en Malcom en la sección "Configurar". H*ay que tener en cuenta que previamente tendremos que haber activado el módulo de configuración en caso contrario, la librería lanzará excepción*.

NOTA: Para los que tengáis integrada una versión de la librería anterior a 2013 tenéis que hacer los siguientes cambios:

- Eliminar el archivo .properties.

- Eliminar el activity de malcom.
- Seguir las instrucciones anteriores en el activity principal.

Módulo de Notificaciones

Este módulo permite el registro en Malcom para la recepción de Notificaciones mediante el sistema de notificaciones de Google GCM (Google Cloud Messaging).

Para usar este módulo hay que seguir los siguientes pasos de configuración adicional:

1. Hay que indicar el senderId obtenido en google de la siguiente forma:

```
MCMCoreAdapter.getInstance().setSenderId(SENDER_ID);
```

2. Configuración adicional en la aplicación en el fichero AndroidManifest.xml, donde <PACKAGE> es el paquete de su aplicación:
    1.
```
<permission  android:name="<PACKAGE>.permission.C2D_MESSAGE"
       android:protectionLevel="signature" />
<uses-permission    android:name="<PACKAGE>.permission.C2D_MESSAGE" />
<uses-permission
       android:name="com.google.android.c2dm.permission.RECEIVE" />
<uses-permission android:name="android.permission.WAKE_LOCK" />
```
    2.
```
<receiver
android:name="com.malcom.library.android.module.notifications.gcm.MalcomGCM
BroadcastReceiver"
  android:permission="com.google.android.c2dm.permission.SEND" >
  <intent-filter>
    <action android:name="com.google.android.c2dm.intent.RECEIVE" />
    <action android:name="com.google.android.c2dm.intent.REGISTRATION" />
    <category android:name="<PACKAGE>" />
  </intent-filter>
</receiver>
```
    3.
```
<service
    android:name="com.malcom.library.android.module.notifications.gcm.GCMInt
    entService" />
<service
    android:name="com.malcom.library.android.module.notifications.services.P
    endingAcksDeliveryService" />
```

Una vez configurado correctamente el módulo, para usarlo:

1. Incluimos en el método **onResume()** de la aplicación:

```
MCMCoreAdapter.getInstance().moduleNotificationsRegister(this,
       EnvironmentType.SANDBOX,"<Titulo>",<MainActivity>.class);
```

esta linea se encarga del registro del dispositivo en Malcom y de la activación del módulo. Donde indica "<Titulo>", será el texto que queramos que aparezca al recibir una notificación en la barra superior de Android y, "<MainActivity>" será la activity principal de la aplicación (*esta activity será la que se abra cuando se*

mire la notificación recibida). En la linea vemos también `EnvironmentType.SANDBOX`, esto es el entorno en el que queremos que el dispositivo se registre en Malcom. Tenemos dos opciones; SANDBOX y PRODUCTION.
Tambien es posible ejecutar el registro sin indicar el entorno:

```
MCMCoreAdapter.getInstance().moduleNotificationsRegister(getApplicationConte
xt(), "<Titulo>",<MainActivity>.class);
```

...en este caso, el entono se deduce del valor del flag “debuggable” del fichero “AndroidManifest.xml” de la aplicación donde si esta activo será SANDBOX y en caso contrario, PRODUCTION.

Incluimos la declaración:

```
MalcomNotificationReceiver notReceiver = null;
private Intent intent = null;
```

Añadimos a continuación esta otra linea:

```
MCMCoreAdapter.getInstance()
              .moduleNotificationsCheckForNewNotifications( getAppli
              cationContext(),intent);
```

2. Añadimos lo siguiente en el **onCreate()**:

```
notReceiver = new MalcomNotificationReceiver();
registerReceiver(notReceiver, new
       IntentFilter(MCMNotificationModule.SHOW_NOTIFICATION_ACTION));

intent = getIntent();
```

Sobre-escribimos el método **onNewIntent()** tal que:

```
@Override
protected void onNewIntent(Intent intent) {
       super.onNewIntent(intent);
       this.intent = intent;
}
```

Y sobre-escribimos el método **onDestroy()**:

```
@Override
protected void onDestroy() {
       super.onDestroy();
       unregisterReceiver(notReceiver);
       GCMRegistrar.onDestroy(getApplicationContext());
}
```

A partir de este momento la aplicación será capaz de recibir notificaciones. En el caso de querer desregistrar la aplicación, se dispone de el método:

```
MCMCoreAdapter.getInstance().moduleNotificationsUnregister(getApplicationContext()
);
```

Cuando una notificación se abre, se envia el ACK a Malcom para su procesado. En el caso de que no sea posible dicho envío, el envío del ACK se pospone para cuando sea posible su envío.

Módulo de Estadísticas

Permite enviar estadísticas a Malcom del uso de la aplicación.

Este módulo requiere la inclusión en el AndroidManifest.xml de la aplicación de la linea:

```
<service android:name =
"com.malcom.library.android.module.stats.services.PendingBeaconsDeliveryService" /
>
```

Para iniciar el registro de actividad ejecutaremos la linea siguiente:

```
MCMCoreAdapter.getInstance().moduleStatsStartBeacon(this, true);
```

donde el segundo parámetro le indica si enviar información de geo-localizacion y el tercero permite indicarle una lista de tags.

Cuando queramos enviar las estadísticas recogidas a Malcom, usaremos esta otra linea:

```
MCMCoreAdapter.getInstance().moduleStatsEndBeacon();
```

En el caso de querer usar sub-beacons tendremos respectivamente para iniciar y parar:

```
MCMCoreAdapter.getInstance().moduleStatsStartBeaconWithName("subB1");
MCMCoreAdapter.getInstance().moduleStatsEndBeaconWithName("subB1");
```

Por defecto los sub-beacons son automáticamente procesados al llamar al stop de los beacons asi que no hace falta realizar la llamada de parada de un sub-beacon.

Para añadir o eliminar tags que serán enviados con el beacon (para el uso de segmentos) disponemos de estos dos métodos:

```
MCMCoreAdapter.getInstance().moduleStatsAddTag("tag");
MCMCoreAdapter.getInstance().moduleStatsRemoveTag("tag");
```

Los lugares recomendados para iniciar el proceso de beacons y el envio son:

- **onResume**(). Inicio del procesado.
- **OnPause**(). Stop del procesado (envío del beacon/estadísticas).

Usando los puntos recomendados nos aseguramos que el procesado se inicia cuando la app

esta activa y que los adtos se envian una vez la app. Se deja de usar y pasa a segundo plano y/o se cierra.

Nota: Todo aquel beacon que no pueda ser enviado a Malcom por el motivo que sea (fallo de cobertura por ejemplo), será procesado para su posterior envío en cuanto sea posible.

Modulo de Publicidad

Para usar este módulo, se han de configurar en el *MCMProperties.properties* de la librería los parámetros adAppName, adCompanyName y adWhirlId tal como se explicó en la sección "Assets de la librería". Una vez hecho esto, para activar la publicidad en una activity tenemos dos opciones dependiendo de si dicha publicidad forma parte de las disponibles en Malcom o no.

- Publicidad disponible en Malcom. En este caso basta con añadir al layout de la activity lo siguiente:

```
<LinearLayout
  android:id="@+id/ad_layout"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:orientation="vertical" >
</LinearLayout>
```

Y en el activity, dentro de el onCreate(), poner lo siguiente:

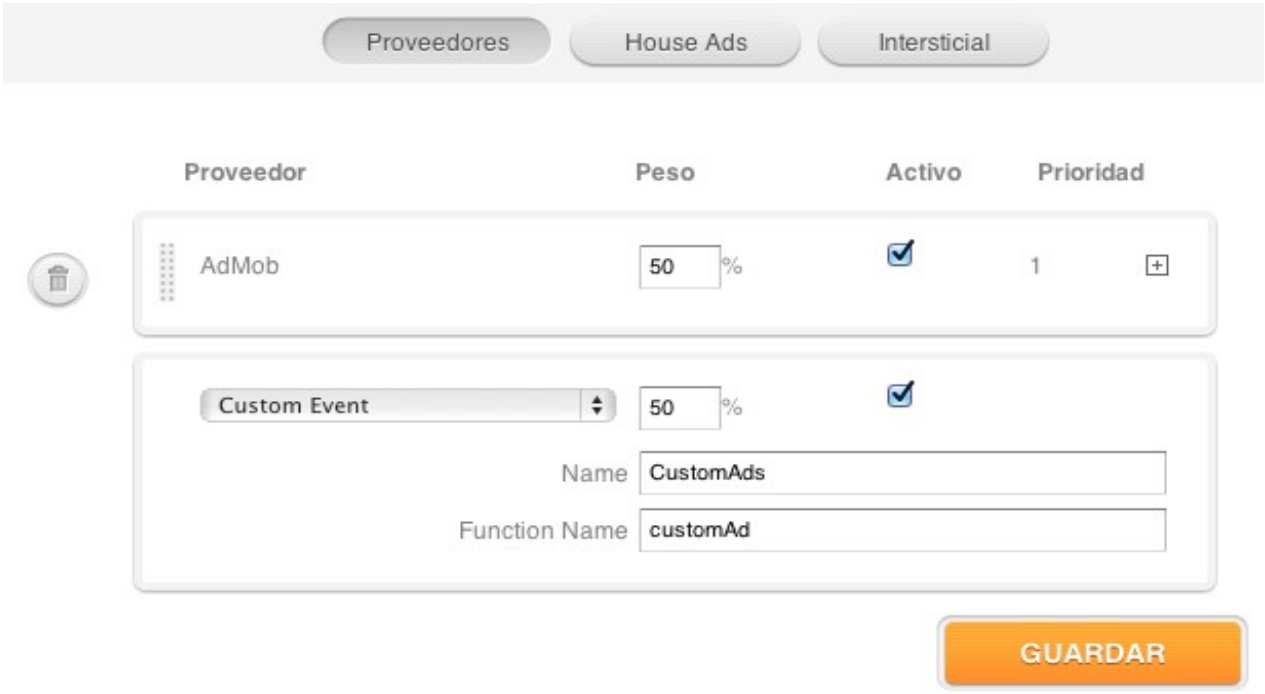
```
LinearLayout layout = (LinearLayout)findViewById(R.id.ad_layout);
//MCMCoreAdapter.getInstance().adWidth = +WIDTH+;
//MCMCoreAdapter.getInstance().adHeight = +HEIGHT+;
MCMCoreAdapter.getInstance().moduleAdsActivate(this, ADS_ID, layout);

layout.setGravity(Gravity.CENTER_HORIZONTAL);
layout.setVerticalGravity(Gravity.BOTTOM);
```

Donde "ADS_ID" es el identificador de la publicidad obtenido en malcom.

Como vemos, hay dos lineas comentadas. Por defecto, el tamaño de la publicidad es de 320x52, en caso de querer otro tamaño se puede indicar mediante estas dos lineas.

- Publicidad no disponible en Malcom. En este caso, en Malcom habremos creado previamente un "Custom Event" dentro de "Monetizar" tal que:



En la aplicación Android, crearemos entonces una clase que extienda de "MCMAdEventhandler" y que será algo similar a:

```
package com.malcom.android.demo.app;

import android.app.Activity;
import com.malcom.library.android.module.ad.MCMAdEventHandler;

public class CustomEvent extends MCMAdEventHandler {

	private Activity context;

	public CustomEvent(Activity context) {
		super();
		this.context = context;
	}

	public void customAd() {
		//TODO
	}
}
```

El nombre del método debajo del constructor "**customAd**()" debe ser exactamente el mismo que el valor que hayamos puesto en Malcom al crear el custom event en la propiedad "Function Name". Dentro de este método implementaremos el sistema particular.

Una vez hecho esto, añadiremos a la activity que queramos que tenga la publicidad el layout , tal como en el caso de una publicidad disponible en Malcom, pero el código de llamada será un tanto diferente:

```
LinearLayout layout = (LinearLayout)findViewById(R.id.ad_layout);
//MCMCoreAdapter.getInstance().adWidth = +WIDTH+;
//MCMCoreAdapter.getInstance().adHeight = +HEIGHT+;
MCMCoreAdapter.getInstance().moduleAdAactivate(this, layout, new
CustomEvent(this));

layout.setGravity(Gravity.CENTER_HORIZONTAL);
layout.setVerticalGravity(Gravity.BOTTOM);
```

Campañas

Una vez configurado en la web de Malcom, para hacer uso de las campañas cross-selling tenemos que añadir al layout donde aparecerá el banner lo siguiente:

```
<RelativeLayout
           android:id="@+id/campaing_banner_layout"
           android:layout_width="fill_parent"
           android:layout_height="fill_parent"
           android:orientation="vertical"
           android:visibility="visible"
           android:gravity="center_vertical">

           <ImageView android:id="@+id/image_campaing_view"
                  android:layout_width="fill_parent"
                  android:layout_height="fill_parent"
                  android:visibility="visible"/>

</RelativeLayout>
```

Y en el activity la siguiente línea:

```
MCMCoreAdapter.getInstance().moduleCampaingAddBanner(this);
```

Recursos Adicionales de la Librería

Son un conjunto de ficheros asociados a los diferentes módulos usados. Estos ficheros serán necesarios según se activen/usen unos u otros módulos.

Son los siguientes:

- **mcmconfiginfo.json**. Es la configuración por defecto usada por el módulo de configuración para el caso de que por ejemplo no este disponible la red.
- **splash.img**. Este es el fichero de imagen splash para el módulo de configuración (*es usado en el modo off-line si previamente no se pudo descargar una configuración. El fichero que deseé como splash se renombra a este nombre y se pone en el directorio splash de la aplicación)*.

Contacto

Malcom Ventures S.L.
http://www.mymalcom.com/