Hangit SDK for Android Integration Guide

Table of Contents

Version

Date	Version	Author	Release Notes
3/10/2015	2.0.1	Andrew Peret	• Update battery optimizations. •

Overview

The HangIT SDK provides location based messaging and tracking capabilities for an Android application.

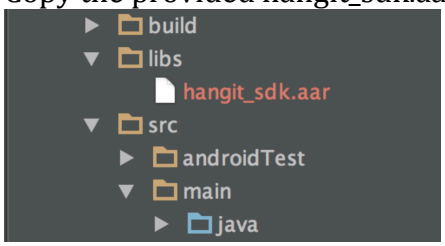
Download

The HangIT SDK library and API key will be provided upon request.

Installing the Hangit SDK

Add Hangit SDK to your Project

Copy the provided hangit_sdk.aar file to the /libs directory in the Android module.



If the /libs directory is not present, switch to “Project” view. The libs directory should be at the same level as the /src directory.

Add a flatDir refenence to the build.gradle file in the Android module.

```
repositories {
  flatDir {
    dirs ‘libs’
  }
}
```

Add a dependency to the hangit_sdk library in the Android module’s build.gradle file

```
dependencies {
  compile(name:'hangit_sdk', ext:'aar')
  compile 'com.google.android.gms:play-services:6.5.87'
}
```

The hangit_sdk requires a google play services be added as a dependency. Without google play services the following AndroidManifest.xml file will throw a compilation error on meta-data com.google.android.gms.version.

Setting Permissions

Set the following permissions within the <manifest> node.

```
AndroidManifest.xml <manifest>
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<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="com.google.android.providers.gsf.permission.READ_GSERVICES"/>
<uses-permission android:name="com.google.android.gms.permission.ACTIVITY_RECOGNITION" />
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
<meta-data android:name="com.google.android.gms.version" android:value="@integer/google_play_servi
ces_version" />
```

Link Hangit SDK to your Project

Add references to the hangit_sdk services within the node.

```
AndroidManifest.xml Application
<service android:name="com.hangit.android.hangit_sdk.ServiceHangITLocation"/>
<service android:name="com.hangit.android.hangit_sdk.ServiceActivityRecognition" />
<receiver android:name="com.hangit.android.hangit_sdk.receiver.BootupReceiver" >
  <intent-filter>
    <action android:name="android.intent.action.BOOT_COMPLETED" />
    <action android:name="android.intent.action.QUICKBOOT_POWERON" />
  </intent-filter>
</receiver>
```

i. HangITLocationService - Interacts with the Android device to obtain locations updates via GPS, wifi, and network.
ii. ActivityRecognitionService - Interacts with the Android device's activity messages for location accuracy.
iii. BootupReceiver - Allows our services to start when the device boots up.

.Add the hangIt SDK key to a reference file. In the provided sample code the SDK key is stored in the strings.xml file. Code reference below.

```
String.xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
<string name="hangit_sdk_key">[KEY OBTAINED FROM HANGIT]</string>
</resources>
```

Integration Options

The Hangit SDK provides flexibility on the level of integration required for your application. The Hangit SDK can support the entire lifecycle. Monitoring a users location, triggering a location event, displaying the alert notification and then presenting the user with a rich message based on the Hangit Portal configuration.

Initialize HangITClient

To run the hangITClient in the background the HangITClient must first be configured. The following code snippet is required.

```
GeneralManager.getHangITClient().addEventListener(this);
```

addEventListener – The HangITClient requires an implementation of HangITClient.HangITClientEventListener.

After the hangItClient is configured the hangItClient.initialize() method is required to start the hangItClient background services. Add the following code snippet after the configuration method calls in the onCreate method of the Activity intended to start HangIT background services.

```
GeneralManager.getHangITClient().initialize(this, getString(R.string.hangit_sdk_key));
```

The initialize method initializes and starts the hangIt services on the device.
For sample code, see the BackgroundOnly.java activity in the PublisherExample application.

Location Tracking

The initialize method will begin location tracking. The HangIT SDK will maintain the locations received from the device along with the geo fences nearby and communicate activities to the HangIT server. No additional code is needed beyond the initialize() method call.

Location Event Trigger and Callback

Once the initialize method is called all events communicated from the HangIT SDK will be communicated via the HangITClientEventListener iterface. The HangITClientEventListener is set during the addEventListener() method call. HangITClientEventListener allows location event interception via the onHangITPlaceEncountered method.

```
….
GeneralManager.getHangITClient().addEventListener(this);
….

@Override
public void onPlaceEncountered(Place place) {
```

```
}
```

Background option
To obtain the information from the onPlaceEncountered event when the app is force closed a broadcast receiver is required.
First create a class that extends BroadcastReceiver. Example below.

```
public class PlaceEncounteredReceiver extends BroadcastReceiver {
    public PlaceEncounteredReceiver() {}
    @Override
    public void onReceive(Context context, Intent intent) {
       String placeData = intent.getStringExtra("placedata");
    }
}
```

Next identify the receiver in the AndroidManifest.xml file.

```
<receiver android:name="PlaceEncounteredReceiver" >
  <intent-filter>
    <action android:name="com.hangit.android.hangit_sdk.BroadcastPlaceEncountered" />
  </intent-filter>
</receiver>
```

Location Event Trigger and Local Notification with Callback

To override the standard callback activity provided by the HangIT SDK, replace the "android:value" parameter of the com.hangit.android.hangit_sdk.notification_activity meta-data node in the AndroidManifest.

```
AndroidManifest.xml Application
<service android:name="com.hangit.android.hangit_sdk.service.HangITLocationService">
…
<meta-data android:name="com.hangit.android.hangit_sdk.notification_activity" android:value="[CALL
 BACK ACTIVITY FULLY QUALIFIED NAME]"/>
</service>
```

This will redirect any down press action of a notification to the specified activity.

Location Event Trigger and Local Notification with Rich Message

The HangIT SDK provides two activities for seamless notification call back integration. In the previously mentioned com.hangit.android.hangit_sdk.notification_activity the following preset activities can be plugged in:
1 – com.hangit.android.hangit_sdk.UIADUnitActivity – This activity will display the standard HangIT ad unit.
2 – com.hangit.android.hangit_sdk.UIWebViewActivity – This activity will display a ful webview loaded with the HTML identified for the location in the HangIT Portal.

Modular – Map & Wallet

The HangIT SDK has activities that can be run in tandem with a hosting android application. To utilize the Map & Wallet, the application must have a google api key identified in the Application node of the AndroidManifest.xml file.

```
<application…>
…
  <meta-data
    android:name="com.google.android.maps.v2.API_KEY"
    android:value="[GOOGLE API KEY]" />
```

[GOOGLE API KEY] – A google API key can be obtained and authorized at https://code.google.com/apis/console.

The MainActivity and AdUnitActivity must be added to the AndroidManifest.xml application node.

```
<application ...>
…
  <activity android:name="com.hangit.android.hangit_sdk.UIMainActivity" android:screenOrientation=
"portrait"/>
  <activity android:name="com.hangit.android.hangit_sdk.UIADUnitActivity" android:screenOrientatio
n="portrait"/>
```

To start the Map & Wallet call start activity utilizing the UIMainActivity.makeIntent method. The code below is from the PushButton.java activity in the PublisherExample application.

```
startActivity(UIMainActivity.makeIntent(PushButton.this, UIMainActivity.class,

                        START_ACTIVITY_SOURCE));
```