Easy Local Android Notifications

User Guide v1.03

Carlos Fernandez Musoles

Support: carlos.fernandez.musoles@gmail.com

Local notifications

Local notifications are messages delivered from your application to users –all within the same device; these messages appear within the notifications bar in android (usually on the draggable area on top of your device), and are guaranteed to reach the user regardless of whether they are using the app that generate them or not.

Easy Local Android Notifications (ELAN) packs all the Java code in one plugin and allows users to set timed notifications –messages to appear in the user's notification board after a certain period of time. Of course, this delay can be 0 and you will get the messages immediately.

Local notifications are well suited for any kind of app-to-user communication, such as: reminders to get back to your game, periodic events, even advertising!

Requirements to use ELAN

You will need the following to use ELAN within your project:

- Unity Android
- Any android device (notifications only work on the actual device and not in the Unity Editor)

How to use ELAN?

Easy Local Android Notifications package is, well, easy to use! The package includes a demo scene (within Plugins>Android>ELAN folder) ready to run on your android device.

Just import all the files within the package to an empty project. Then, open and build the scene called "demo" on any android device. Now you can send local notifications from within Unity!

To best utilize the demo, set the build to show in landscape on your device.

ELAN functions

ELAN has three functions you can call to set up different local:

1) ELANManager.SendNotification(string title, string message, int delay). It is pretty self explanatory, but here is a description of each of the parameters:

- title: string that will act as the notification title (in bold)
- message: string that will be used as the notification message
- delay: how many seconds to wait before firing the notification. Note that the notification will be fired whether the user is still within the app or no. The current version only accepts integers.

- **Returns** an integer (notification ID, you can use it to cancel the notification)

2) ELANManager.ScheduleRepeatingNotification(string title, string message, int delay, int repeat). Similar to SendNotification(..) above, but with an extra parameter that makes the notification show up repeatedly:
- repeat: time in between repetitions (the notification will be fired every 'repeat' seconds). It must be an integer.
Currently the plugin only offers supports for 1 repeating notification per app – you can cancel and set another one, though.

3) To cancel a scheduled repeating notification call ELANManager.CancelRepeatingNotification().

4) To cancel a single notification (not repeating) call ELANManager.CancelNotification(int id). You will need to pass the ID with which the notification was created –SendNotification() now returns an integer, which is the notification ID you need here.

5) To cancel all notifications (single, delayed and repeating), call ELANManager.CancelAllNotifications(). **NOTE** that form version 1.03 this call will only remove notifications set on the current session (and all repeating notifications of any session). If you want to cancel previously set notifications, you are responsible of tracking the notification IDs (which are returned by SendNotification().

Importing ELAN to your existing project

If you want to have the ability to send local notifications in your project, all you have to do is import all the files within the package to your existing project. If you didn't have an AndroidManifest.xml file, one is provided –though some alterations are required; see below. If you did, you only have to include the following lines AS THEY ARE WRITTEN HERE anywhere within your <application> tag:

```
<service android:enabled="true"
android:name="com.CFM.ELAN.ELANAlarmCreator" />
<receiver android:name="com.CFM.ELAN.TimedAlarm" />
```

If you didn't have an AndroidManifest, you can use the one included in the package. Just remember to change the package name to your own:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
android:installLocation="preferExternal"
      package="com.CFM.ELAN"
      android:versionName="1.00"
      android:versionCode="1">
```

That's it! Now you can call any of the static methods within ELANManager –see ELAN functions section above- from anywhere in your project to set up your own

notifications.

What can you do with ELAN?

- You can send notifications from your game to the user device (locally), with or without delays.
- The messages are guaranteed to reach the user, whether he is on your app or not.
- You can fire multiple notifications with different delays –all of them will reach at their time, without interferences.
- When the user clicks on the notifications he is brought back to your app.
- You can also schedule repeating notifications, which will show periodically on the notifications bar.
- At any time you can cancel any type of notification –the built in function will only cancel notifications set on the current session, the user is responsible of tracking notification IDs if he wants to cancel previous session notifications

Room for improvement

You are well equipped now to deal with all sorts of local notifications. However, for the curious out there, I've included the java files within the ELAN.jar package. From studying the code to improving it, you own it! Go ahead and make us proud.

Here is a list of some of the improvements you may be interested on:

Change notification icon

If you want another icon to be shown in your android devices, just replace the png file under the folder Plugins>Android>res>drawable. Place your new icon there and name it 'ic_launcher.png'. Done!

Set up multiple recurrent notifications

This is a tough one, but digging into the ELANAlarmCreator.java, more specifically, within the setRepeatingAlarm(…) method, you could twitch it to accept multiple pending intents. Hints: you will need to change the current FLAG_CANCEL_CURRENT and use a custom known ID (not 0 as it is used now for all notifications).

Of course, you will need to twitch the cancelRepeatingNotirication(…) method to accept custom IDs –otherwise your repeating alarms would run forever!

Add spice to your notifications

Want to add sound or vibrations to your notifications? Add the appropriate flags to them within the onReceive method in TimedAlarm class:

```
notif.defaults |= Notification.DEFAULT_SOUND;
notif.defaults |= Notification.DEFAULT_VIBRATE;
```

You could also change the type of sound played! Just remember that for vibration you need to set an extra permission on your AndroidManifest:

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

Version control

Changes to version 1.04

- Fixed the time limit on 'delay' and 'repeat' parameters for notifications (not it is 9223372036854774, roughly 300k years)
    - ELANManager.cs: changes to functions signature (SendNotification(…) and ScheduleRepeatingNotification(…))
    - ELAN.Jar files changed to accommodate the fix:
        - ELANAlarmCreator.java and ELANManager.java (not 'time' and 'repeat' are long type, not integer).

Changes to version 1.03

- Changes to ELANManager.cs:
    - CancelAllNotifications() now only cancels current sessions notifications

Changes to version 1.02

- Changes to ELANManager.cs:
    - All functions are now wrapped in compiler directives so you won't experience errors while using ELAN in a cross-platform project
    - Two new public functions added: CancelAllNotifications() and CancelNotification(id).
    - SendNotification() changed –now returns notification ID (int)
- Changes to ELANManager.java and ELANAlarmCreator.java to accommodate cancellations.

Changes to version 1.01

- Changes to AndroidManifest.xml to switch from activity to service (to fix the screen blackout issue)
    - Substitute the <activity> tag previously used for the <service> tag, as specified in the guide
- Changes to the ELAN jar plugin, specifically to the classes:
    - ELANManager.java
    - ELANAlarmCreator.java