Easy Local Android Notifications

User Guide v1.12

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.

**** NEW SYSTEM ****

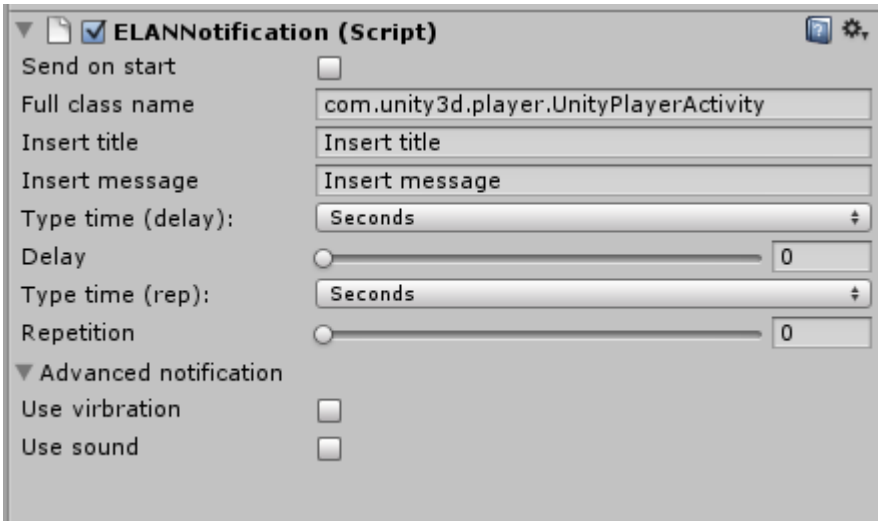
From version 1.1 we have redesigned the way ELAN creates, prepares and sends notifications. We have moved from using the ELANManager static class to send notifications (via SendNotification() and ScheduleRepeatingNotification()) to a new *modular* component-based system. You can still use the old methods to send notifications, but they are now deprecated and won't be supported anymore, so we recommend you to adapt your projects to the new system as soon as possible.

Migrating from ELAN 1.04 or older

If you are using ELAN 1.04 or older and you want to use the new system, fear not! In its core, ELAN is still the same good little helpful plugin.

The intended way of sending notifications is now using the ELANNotification object. You can create notifications in two different ways:

1) Attach it to any of your game objects and configure it via the inspector



1) Create an instance of that object via code.

```
parametrizedNotification5 = new ELANNotification();
parametrizedNotification5.message = "Hello from parametrized notification 5";
parametrizedNotification5.title = "parametrized notification 5";
parametrizedNotification5.repetition = 5;
parametrizedNotification5.delay = 3;
parametrizedNotification5.useSound = true;
parametrizedNotification5.useVibration = true;
parametrizedNotification5.send();
```

Before moving forward, let's see in more detail what options you can set up for your notifications. Since you can set a notification via code and via the inspector, we will specify both for reference:

- *Send on Start* (Inspector only): whether the notification will be sent when the Start() function of the object is called (once the object is instantiated)
- *Full clas name* (ELANNotification.fullClassName): please section *When Your Project extends the UnityPlayerActivity*. In normal circumstances, this should be left as the default "com.unity3d.player.UnityPlayerActivity"
- *Title* (ELANNotification.title): Title you want your notification to display
- *Message* (ELANNotification.message): message you want your notification to display.
- *Type time* for delay and repetition (ELANNotification.delayTypeTime and ELANNotification.repetitionTypeTime): When setting delays and repetitions to your notifications, you can specify the unit of time; there are 4 available: seconds, minutes, hours and days.

- *Delay* and *repetition* (ELANNotification.delay and ELANNotification.repetition): time for your delays / repetitions
- *Use vibration* (ELANNotification.useVibration): whether to use vibration when the notification is received
- *Use sound* (ELANNotification.useSound) : whether to use sound when the notification is received

Independently of what method you choose to create nofitications, you may want to send it at some point -no good to have a bunch of notifications if they don't make it to your device! To do so, you should call the class method *send()*, included on all ELANNotification objects. If you attached it to a game object, you have the option of sending the message automatically on Start; just check the *Send on Start* checkbox and you are good to go.

When your project extends the UnityPlayerActivity

There are certain android plugins that require your project to extend the UnityPlayerActivity. Don't worry about what that means –certain android activities need some feedback from the system or a remote service, and to do that they have to be linked to the main activity of your android app.

Unfortunately, this has some side effects when you want to start or resume your app from an external source, such as a notification. In a nutshell, because the apps no longer run on the default UnityPlayerActivity –which is what ELAN calls to start your apps- there is a missing link and the app does not always resumes correctly.

To fix this, in the past you have to go and hack the java code included in ELAN.jar, recompile and run it again. Because that was a pain, we have updated the workflow and now you can specify your main activity, should your app need it. Every ELANNotification has a field called *fullClassName* which you can populate either via code or through the inspector. This should include the full qualified name of your main class: the full name of the main activity as it is declared in the androidmanifest file.

How do you know if any of your plugins extend the UnityPlayerActivity

To know whether any other plugin extends the UnityPlayerActivity, open your AndroidManifest.xml and see if any activity has the tags MAIN and LAUNCHER (literally, whether you see those words under any activity). If that activity is not one of the Unity ones (UnityPlayerActivity, UnityNativePlayerActivity or UnityProxyActivity), then your app is extending the basic activity.

Check the name of that activity (should be something like com.companyName.ProjectName.Activity), copy it and use it in your ELANNotifications.fullClassName.

ELANManager.cs functions

ELANManager has 5 functions you can call to set up different local notifications:

1) ***DEPRECATED –****if you want to send notifications, please see the How to use ELAN section.*

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) ***DEPRECATED -*** *if you want to send repeating notifications, please see the How to use ELAN section.*
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) ***DEPRECATED –*** *If you want to cancel single notifications, please use the method available to all ELANNotification objects cancel().*
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().

<u>Importing ELAN to your existing project</u>

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.
- ***NEW***: Now you can specify when you want your notification to be displayed using a System.DateTime object with the ELANNotification.setFireDate(System.DateTime date) method.
- 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
- You can add sound and vibration to any of your 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!

BLACK SCREEN ISSUE

In older versions of ELAN we shipped with a manifest that used to work with Unity 4.2. The folks at Unity seemed to change something in the pipeline on version 4.3 and the manifest was causing issues when restarting apps –black screen bug.

After much help from users and time debugging we managed to detect the error and now we have modified our manifest, so if you are using it, you should be A-OK. If you are using ELAN 1.04 or older and you experience this issue, please modify your manifest as follows:

1) Go to your manifest and remove all activity declarations for UnityPlayerActivity, UnityPlayerProxyActivity and UnityPlayerNativeActivity.

2) Add the following activity:

```
      <activity
      android:name="com.unity3d.player.UnityPlayerActivity
      " android:label="@string/app_name"

         android:configChanges="fontScale|keyboard|keyboar
dHidden|locale|mnc|mcc|navigation|orientation|screenLayou
t|screenSize|smallestScreenSize|uiMode|touchscreen">

       <intent-filter>

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

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

    </intent-filter>

    <meta-data
android:name="unityplayer.ForwardNativeEventsToDalvik"
android:value="true" />

</activity>
```

Version control

Changes to version 1.12

- Added a convenience method to ELANNotification to use a date as opposed to specifying the lapse of time –setFireDate(System.DateTime date), where *date* is the moment you want your notification to be displayed on the notification bar

Changes to version 1.11

- Fixed an issue with scheduling repeating notifications (ELAN.jar updated)

Changes to version 1.1

- Completely redone system to send notifications: now you can create ELANNotification objects via scripts and send them autonomously, or attach the ELANNotification as a component to a game object and have it sent automatically on Start()
- ELANManager.SendNotification() and ELANManager.ScheduleRepeatingNotification() are now deprecated (use ELANNotification instead)
- AndroidManifest changes to fix an issue with Unity 4 and reopening apps when tapping on the notification
- ELAN.Jar has been updated

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