Android - Notifications

⊙ Previous Page   Next Page ⊙

A **notification** is a message you can display to the user outside of your application's normal UI. When you tell the system to issue a notification, it first appears as an icon in the notification area. To see the details of the notification, the user opens the notification drawer. Both the notification area and the notification drawer are system-controlled areas that the user can view at any time.

Android **Toast** class provides a handy way to show users alerts but problem is that these alerts are not persistent which means alert flashes on the screen for a few seconds and then disappears.
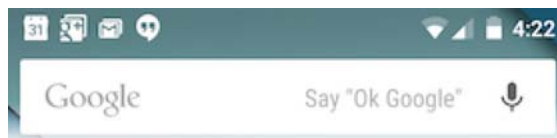
To see the details of the notification, you will have to select the icon which will display notification drawer having detail about the notification. While working with emulator with virtual device, you will have to click and drag down the status bar to expand it which will give you detail as follows. This will be just **64 dp** tall and called normal view.
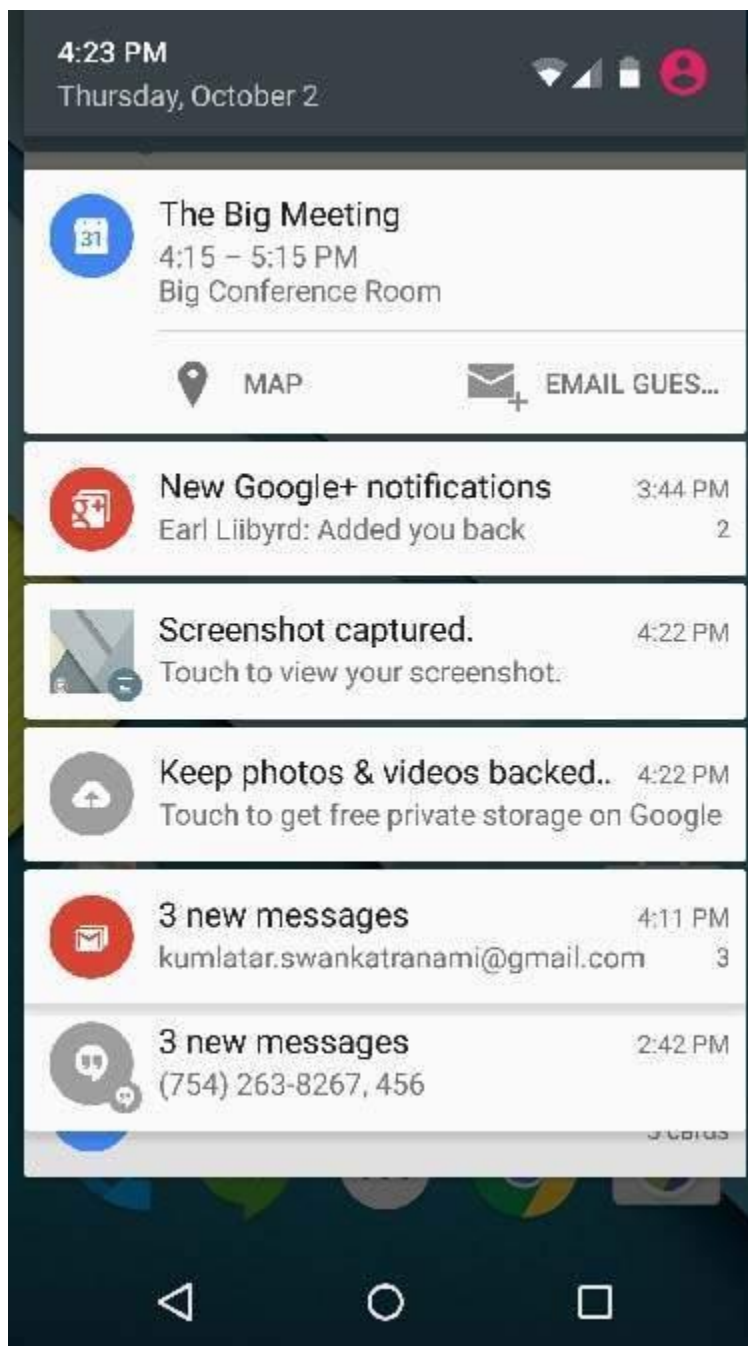
Above expanded form can have a **Big View** which will have additional detail about the notification. You can add upto six additional lines in the notification. The following screen shot shows such notification.

Create and Send Notifications

You have simple way to create a notification. Follow the following steps in your application...

We use cookies to provide and improve our services. By using our site, you consent to our Cookies Policy.

Accept

Learn more

```
NotificationCompat.Builder mBuilder = new NotificationCompat.Builder(this)
```

Step 2 - Setting Notification Properties

Once you have **Builder** object, you can set its Notification properties using Builder object as per your requirement. But this is mandatory to set at least following −

A small icon, set by **setSmallIcon()**

A title, set by **setContentTitle()**

Detail text, set by **setContentText()**

```
mBuilder.setSmallIcon(R.drawable.notification_icon);
mBuilder.setContentTitle("Notification Alert, Click Me!");
mBuilder.setContentText("Hi, This is Android Notification Detail!");
```

You have plenty of optional properties which you can set for your notification. To learn more about them, see the reference documentation for NotificationCompat.Builder.

Step 3 - Attach Actions

This is an optional part and required if you want to attach an action with the notification. An action allows users to go directly from the notification to an **Activity** in your application, where they can look at one or more events or do further work.

The action is defined by a **PendingIntent** containing an **Intent** that starts an Activity in your application. To associate the PendingIntent with a gesture, call the appropriate method of *NotificationCompat.Builder*. For example, if you want to start Activity when the user clicks the notification text in the notification drawer, you add the PendingIntent by calling **setContentIntent()**.

A PendingIntent object helps you to perform an action on your applications behalf, often at a later time, without caring of whether or not your application is running.

We take help of stack builder object which will contain an artificial back stack for the started Activity. This ensures that navigating backward from the Activity leads out of your application to the Home screen.

```
Intent resultIntent = new Intent(this, ResultActivity.class);

TaskStackBuilder stackBuilder = TaskStackBuilder.create(this);

stackBuilder.addParentStack(ResultActivity.class);
```

Finally, you pass the Notification object to the system by calling NotificationManager.notify() to send your notification. Make sure you call **NotificationCompat.Builder.build()** method on builder object before notifying it. This method combines all of the options that have been set and return a new **Notification** object.

```
NotificationManager mNotificationManager = (NotificationManager) getSystemService(Context.NOTIFIC

// notificationID allows you to update the notification later on.
mNotificationManager.notify(notificationID, mBuilder.build());
```

The NotificationCompat.Builder Class

The NotificationCompat.Builder class allows easier control over all the flags, as well as help constructing the typical notification layouts. Following are few important and most frequently used methods available as a part of NotificationCompat.Builder class.

Sr.No.	Constants & Description
1	**Notification build()** Combine all of the options that have been set and return a new Notification object.
2	**NotificationCompat.Builder setAutoCancel (boolean autoCancel)** Setting this flag will make it so the notification is automatically canceled when the user clicks it in the panel.
3	**NotificationCompat.Builder setContent (RemoteViews views)** Supply a custom RemoteViews to use instead of the standard one.
4	**NotificationCompat.Builder setContentInfo (CharSequence info)** Set the large text at the right-hand side of the notification.
5	**NotificationCompat.Builder setContentIntent (PendingIntent intent)**

7	**NotificationCompat.Builder setContentTitle (CharSequence title)**
	Set the text (first row) of the notification, in a standard notification.
8	**NotificationCompat.Builder setDefaults (int defaults)**
	Set the default notification options that will be used.
9	**NotificationCompat.Builder setLargeIcon (Bitmap icon)**
	Set the large icon that is shown in the ticker and notification.
10	**NotificationCompat.Builder setNumber (int number)**
	Set the large number at the right-hand side of the notification.
11	**NotificationCompat.Builder setOngoing (boolean ongoing)**
	Set whether this is an ongoing notification.
12	**NotificationCompat.Builder setSmallIcon (int icon)**
	Set the small icon to use in the notification layouts.
13	**NotificationCompat.Builder setStyle (NotificationCompat.Style style)**
	Add a rich notification style to be applied at build time.
14	**NotificationCompat.Builder setTicker (CharSequence tickerText)**
	Set the text that is displayed in the status bar when the notification first arrives.
15	**NotificationCompat.Builder setVibrate (long[] pattern)**
	Set the vibration pattern to use.
16	**NotificationCompat.Builder setWhen (long when)**
	Set the time that the event occurred. Notifications in the panel are sorted by

Following example shows the functionality of a Android notification using a **NotificationCompat.Builder** Class which has been introduced in Android 4.1.

Step	Description
1	You will use Android studio IDE to create an Android application and name it as *tutorialspoint* under a package *com.example.notificationdemo*.
2	Modify *src/MainActivity.java* file and add the code to notify(""), if user click on the button,it will call android notification service.
3	Create a new Java file *src/NotificationView.java*, which will be used to display new layout as a part of new activity which will be started when user will click any of the notifications
4	Modify layout XML file *res/layout/activity_main.xml* to add Notification button in relative layout.
5	Create a new layout XML file *res/layout/notification.xml*. This will be used as layout file for new activity which will start when user will click any of the notifications.
6	No need to change default string constants. Android studio takes care of default string constants
7	Run the application to launch Android emulator and verify the result of the changes done in the application.

Following is the content of the modified main activity file **src/com.example.notificationdemo/MainActivity.java**. This file can include each of the fundamental lifecycle methods.

```
package com.example.notificationdemo;

import android.app.Activity;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.Context;
import android.content.Intent;
import android.support.v4.app.NotificationCompat;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
```

```java
        public void onClick(View v) {
            addNotification();
        }
    });
}

private void addNotification() {
    NotificationCompat.Builder builder =
        new NotificationCompat.Builder(this)
        .setSmallIcon(R.drawable.abc)
        .setContentTitle("Notifications Example")
        .setContentText("This is a test notification");

    Intent notificationIntent = new Intent(this, MainActivity.class);
    PendingIntent contentIntent = PendingIntent.getActivity(this, 0, notificationIntent,
        PendingIntent.FLAG_UPDATE_CURRENT);
    builder.setContentIntent(contentIntent);

    // Add as notification
    NotificationManager manager = (NotificationManager) getSystemService(Context.NOTIFICATION_S
    manager.notify(0, builder.build());
    }
}
```

Following will be the content of **res/layout/notification.xml** file −

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="400dp"
        android:text="Hi, Your Detailed notification view goes here...." />
</LinearLayout>
```

Following is the content of the modified main activity file **src/com.example.notificationdemo/NotificationView.java**.

```java
package com.example.notificationdemo;

import android.os.Bundle;
import android.app.Activity;

public class NotificationView extends Activity{
    @Override
```

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="MainActivity">

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Notification Example"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:textSize="30dp" />

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Tutorials point "
        android:textColor="#ff87ff09"
        android:textSize="30dp"
        android:layout_below="@+id/textView1"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="48dp" />

    <ImageButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/imageButton"
        android:src="@drawable/abc"
        android:layout_below="@+id/textView2"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="42dp" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Notification"
        android:id="@+id/button"
        android:layout_marginTop="62dp"
        android:layout_below="@+id/imageButton"
        android:layout_centerHorizontal="true" />

</RelativeLayout>
```

Following will be the content of **res/values/strings.xml** to define two new constants.

We use cookies to provide and improve our services. By using our site, you consent to our Cookies Policy.

Accept

Learn more

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.notificationdemo" >

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >

        <activity
            android:name="com.example.notificationdemo.MainActivity"
            android:label="@string/app_name" >

            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>

        </activity>

        <activity android:name=".NotificationView"
            android:label="Details of notification"
            android:parentActivityName=".MainActivity">
            <meta-data
            android:name="android.support.PARENT_ACTIVITY"
            android:value=".MainActivity"/>
        </activity>

    </application>
</manifest>
```

Let's try to run your **tutorialspoint** application. I assume you had created your **AVD** while doing environment set-up. To run the APP from Android Studio, open one of your project's activity files and click Run ⏵ icon from the toolbar. Android Studio installs the app on your AVD and starts it and if everything is fine with your setup and application, it will display following Emulator window —
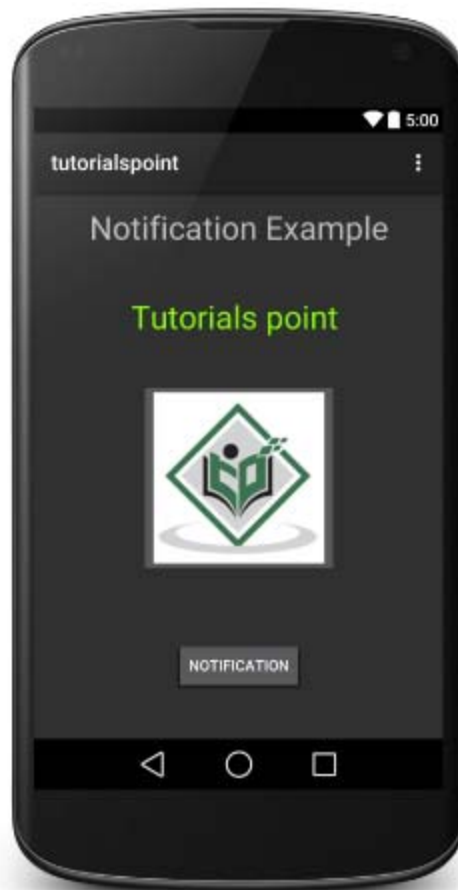
Now click **button**, you will see at the top a message "New Message Alert!" will display momentarily and after that you will have following screen having a small icon at the top left corner.

Now lets expand the view, long click on the small icon, after a second it will display date information and this is the time when you should drag status bar down without releasing mouse. You will see status bar will expand and you will get following screen –
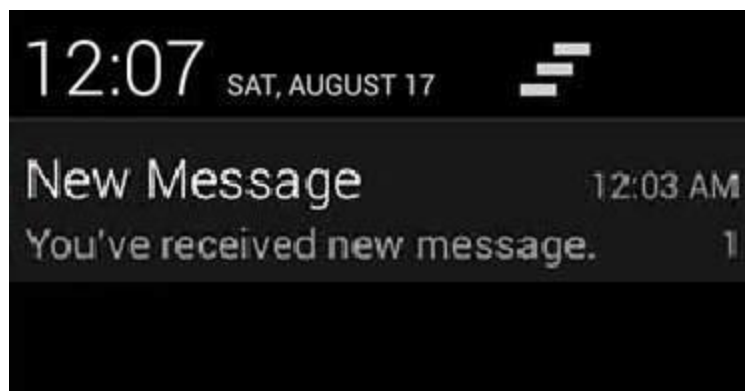
We use cookies to provide and improve our services. By using our site, you consent to our Cookies Policy.

Accept

Learn more

The following code snippet demonstrates how to alter the notification created in the previous snippet to use the Inbox big view style. I'm going to update displayNotification() modification method to show this functionality −

```
protected void displayNotification() {
    Log.i("Start", "notification");

    /* Invoking the default notification service */
    NotificationCompat.Builder  mBuilder = new NotificationCompat.Builder(this);

    mBuilder.setContentTitle("New Message");
    mBuilder.setContentText("You've received new message.");
    mBuilder.setTicker("New Message Alert!");
    mBuilder.setSmallIcon(R.drawable.woman);

    /* Increase notification number every time a new notification arrives */
    mBuilder.setNumber(++numMessages);

    /* Add Big View Specific Configuration */
    NotificationCompat.InboxStyle inboxStyle = new NotificationCompat.InboxStyle();

    String[] events = new String[6];
    events[0] = new String("This is first line....");
    events[1] = new String("This is second line...");
    events[2] = new String("This is third line...");
    events[3] = new String("This is 4th line...");
    events[4] = new String("This is 5th line...");
    events[5] = new String("This is 6th line...");

    // Sets a title for the Inbox style big view
    inboxStyle.setBigContentTitle("Big Title Details:");

    // Moves events into the big view
    for (int i=0; i < events.length; i++) {
        inboxStyle.addLine(events[i]);
    }

    mBuilder.setStyle(inboxStyle);

    /* Creates an explicit intent for an Activity in your app */
    Intent resultIntent = new Intent(this, NotificationView.class);

    TaskStackBuilder stackBuilder = TaskStackBuilder.create(this);
    stackBuilder.addParentStack(NotificationView.class);

    /* Adds the Intent that starts the Activity to the top of the stack */
    stackBuilder.addNextIntent(resultIntent);
    PendingIntent resultPendingIntent =stackBuilder.getPendingIntent(0,PendingIntent.FLAG_UPDATE_C
```
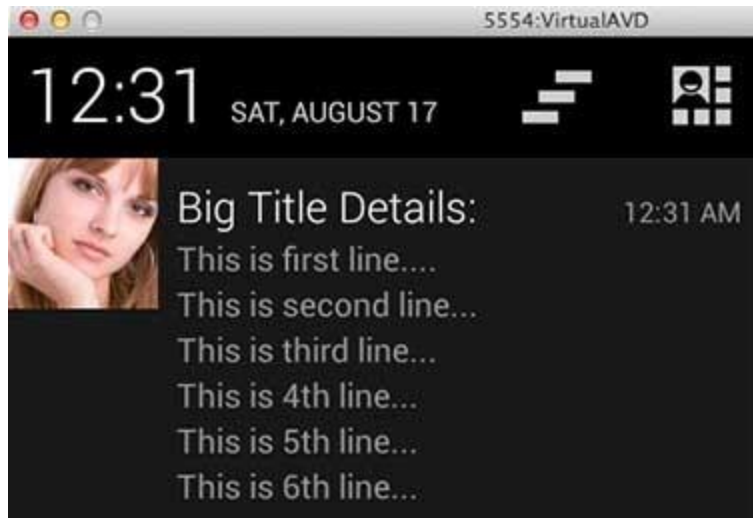
We use cookies to provide and improve our services. By using our site, you consent to our Cookies Policy.

Accept

Learn more

Now if you will try to run your application then you will find following result in expanded form of the view −

Advertisements

We use cookies to provide and improve our services. By using our site, you consent to our Cookies Policy.

Accept

Learn more

Privacy Policy Cookies Policy Contact

| Enter email for newsletter | go |

We use cookies to provide and improve our services. By using our site, you consent to our Cookies Policy.

Accept

Learn more