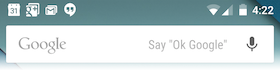
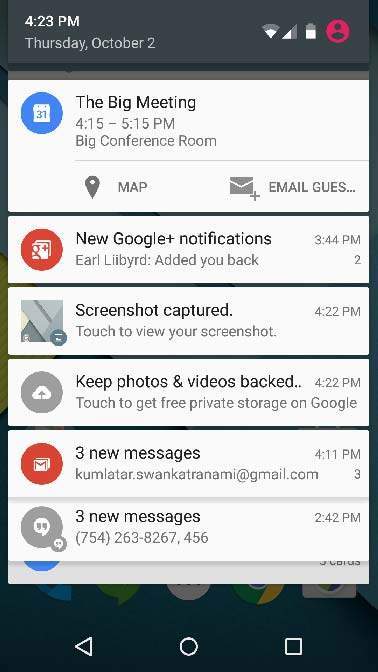
**Android Notification**

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 to create a notification −

**Step 1** - Create Notification Builder

As a first step is to create a notification builder using *NotificationCompat.Builder.build()*. You will use Notification Builder to set various Notification properties like its small and large icons, title, priority etc.

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);

// Adds the Intent that starts the Activity to the top of the stack

stackBuilder.addNextIntent(resultIntent);

PendingIntent resultPendingIntent = stackBuilder.getPendingIntent(0,PendingIntent.FLAG\_UPDATE\_CURRENT);

mBuilder.setContentIntent(resultPendingIntent);

**Step 4** - Issue the notification

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.NOTIFICATION\_SERVICE);

// 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)**  Supply a PendingIntent to send when the notification is clicked. |
| 6 | **NotificationCompat.Builder setContentText (CharSequence text)**  Set the text (second row) of the notification, in a standard notification. |
| 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 this time. |