[Skip to content](https://developer.android.com/guide/topics/ui/notifiers/notifications.html" \l "top)

[[Android](https://developer.android.com/index.html) Developers](https://developer.android.com/index.html)

* [DESIGN](https://developer.android.com/design/index.html)

* [DEVELOP](https://developer.android.com/develop/index.html)

* [DISTRIBUTE](https://developer.android.com/distribute/index.html)

[DEVELOPER CONSOLE](https://play.google.com/apps/publish/)

Top of Form



Bottom of Form

navigation

 API Guides

* [**Introduction**](https://developer.android.com/guide/index.html)
* [**App Components**](https://developer.android.com/guide/components/index.html)
* [**App Resources**](https://developer.android.com/guide/topics/resources/index.html)
* [**App Manifest**](https://developer.android.com/guide/topics/manifest/manifest-intro.html)
* [**User Interface**](https://developer.android.com/guide/topics/ui/index.html)
  + [**Overview**](https://developer.android.com/guide/topics/ui/overview.html)
  + [**Layouts**](https://developer.android.com/guide/topics/ui/declaring-layout.html)
  + [**Input Controls**](https://developer.android.com/guide/topics/ui/controls.html)
  + [**Input Events**](https://developer.android.com/guide/topics/ui/ui-events.html)
  + [**Menus**](https://developer.android.com/guide/topics/ui/menus.html)
  + [**Settings**](https://developer.android.com/guide/topics/ui/settings.html)
  + [**Dialogs**](https://developer.android.com/guide/topics/ui/dialogs.html)
  + [**Notifications**](https://developer.android.com/guide/topics/ui/notifiers/notifications.html)
  + [**Toasts**](https://developer.android.com/guide/topics/ui/notifiers/toasts.html)
  + [**Search**](https://developer.android.com/guide/topics/search/index.html)
  + [**Drag and Drop**](https://developer.android.com/guide/topics/ui/drag-drop.html)
  + [**Accessibility**](https://developer.android.com/guide/topics/ui/accessibility/index.html)
  + [**Styles and Themes**](https://developer.android.com/guide/topics/ui/themes.html)
  + [**Custom Components**](https://developer.android.com/guide/topics/ui/custom-components.html)
* [**Animation and Graphics**](https://developer.android.com/guide/topics/graphics/index.html)
* [**Computation**](https://developer.android.com/guide/topics/renderscript/index.html)
* [**Media and Camera**](https://developer.android.com/guide/topics/media/index.html)
* [**Location and Sensors**](https://developer.android.com/guide/topics/sensors/index.html)
* [**Connectivity**](https://developer.android.com/guide/topics/connectivity/index.html)
* [**Text and Input**](https://developer.android.com/guide/topics/text/index.html)
* [**Data Storage**](https://developer.android.com/guide/topics/data/index.html)
* [**Libraries**](https://developer.android.com/topic/libraries/index.html)
* [**Administration**](https://developer.android.com/guide/topics/admin/index.html)
* [**Web Apps**](https://developer.android.com/guide/webapps/index.html)
* [**Best Practices**](https://developer.android.com/guide/practices/index.html)
* [HOME](https://developer.android.com/index.html)
* [DESIGN](https://developer.android.com/design/index.html)
* [DEVELOP](https://developer.android.com/develop/index.html)
  + [Training](https://developer.android.com/training/index.html)
  + [API Guides](https://developer.android.com/guide/index.html)
  + [Reference](https://developer.android.com/reference/packages.html)
  + [Samples](https://developer.android.com/samples/index.html)
  + [Android Studio](https://developer.android.com/studio/index.html)
  + [Android NDK](https://developer.android.com/ndk/index.html)
  + [Google Services](https://developer.android.com/google/index.html)
* [DISTRIBUTE](https://developer.android.com/distribute/index.html)
* [PREVIEW](https://developer.android.com/preview/index.html)

* [Develop](https://developer.android.com/develop/index.html)
* [API Guides](https://developer.android.com/guide/index.html)
* [User Interface](https://developer.android.com/guide/topics/ui/index.html)

Notifications

In this document

1. [Design Considerations](https://developer.android.com/guide/topics/ui/notifiers/notifications.html#Design)
2. [Creating a Notification](https://developer.android.com/guide/topics/ui/notifiers/notifications.html#CreateNotification)
   1. [Required notification contents](https://developer.android.com/guide/topics/ui/notifiers/notifications.html#Required)
   2. [Optional notification contents and settings](https://developer.android.com/guide/topics/ui/notifiers/notifications.html#Optional)
   3. [Notification actions](https://developer.android.com/guide/topics/ui/notifiers/notifications.html#Actions)
   4. [Notification priority](https://developer.android.com/guide/topics/ui/notifiers/notifications.html#Priority)
   5. [Creating a simple notification](https://developer.android.com/guide/topics/ui/notifiers/notifications.html#SimpleNotification)
   6. [Applying an expanded layout to a notification](https://developer.android.com/guide/topics/ui/notifiers/notifications.html#ApplyStyle)
   7. [Handling compatibility](https://developer.android.com/guide/topics/ui/notifiers/notifications.html#Compatibility)
3. [Managing Notifications](https://developer.android.com/guide/topics/ui/notifiers/notifications.html#Managing)
   1. [Updating notifications](https://developer.android.com/guide/topics/ui/notifiers/notifications.html#Updating)
   2. [Removing notifications](https://developer.android.com/guide/topics/ui/notifiers/notifications.html#Removing)
4. [Preserving Navigation when Starting an Activity](https://developer.android.com/guide/topics/ui/notifiers/notifications.html#NotificationResponse)
   1. [Setting up a regular activity PendingIntent](https://developer.android.com/guide/topics/ui/notifiers/notifications.html#DirectEntry)
   2. [Setting up a special activity PendingIntent](https://developer.android.com/guide/topics/ui/notifiers/notifications.html#ExtendedNotification)
5. [Displaying Progress in a Notification](https://developer.android.com/guide/topics/ui/notifiers/notifications.html#Progress)
   1. [Displaying a fixed-duration progress indicator](https://developer.android.com/guide/topics/ui/notifiers/notifications.html#FixedProgress)
   2. [Displaying a continuing activity indicator](https://developer.android.com/guide/topics/ui/notifiers/notifications.html#ActivityIndicator)
6. [Notification Metadata](https://developer.android.com/guide/topics/ui/notifiers/notifications.html#metadata)
7. [Heads-up Notifications](https://developer.android.com/guide/topics/ui/notifiers/notifications.html#Heads-up)
8. [Lock Screen Notifications](https://developer.android.com/guide/topics/ui/notifiers/notifications.html#lockscreenNotification)
   1. [Setting Visibility](https://developer.android.com/guide/topics/ui/notifiers/notifications.html#visibility)
   2. [Controlling Media Playback on the Lock Screen](https://developer.android.com/guide/topics/ui/notifiers/notifications.html#controllingMedia)
9. [Custom Notification Layouts](https://developer.android.com/guide/topics/ui/notifiers/notifications.html#CustomNotification)

Key classes

1. [NotificationManager](https://developer.android.com/reference/android/app/NotificationManager.html)
2. [NotificationCompat](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.html)

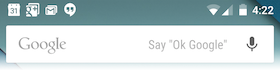
Videos

1. [Notifications in 4.1](http://www.youtube.com/watch?v=Yc8YrVc47TI&feature=player_detailpage#t=1672s)

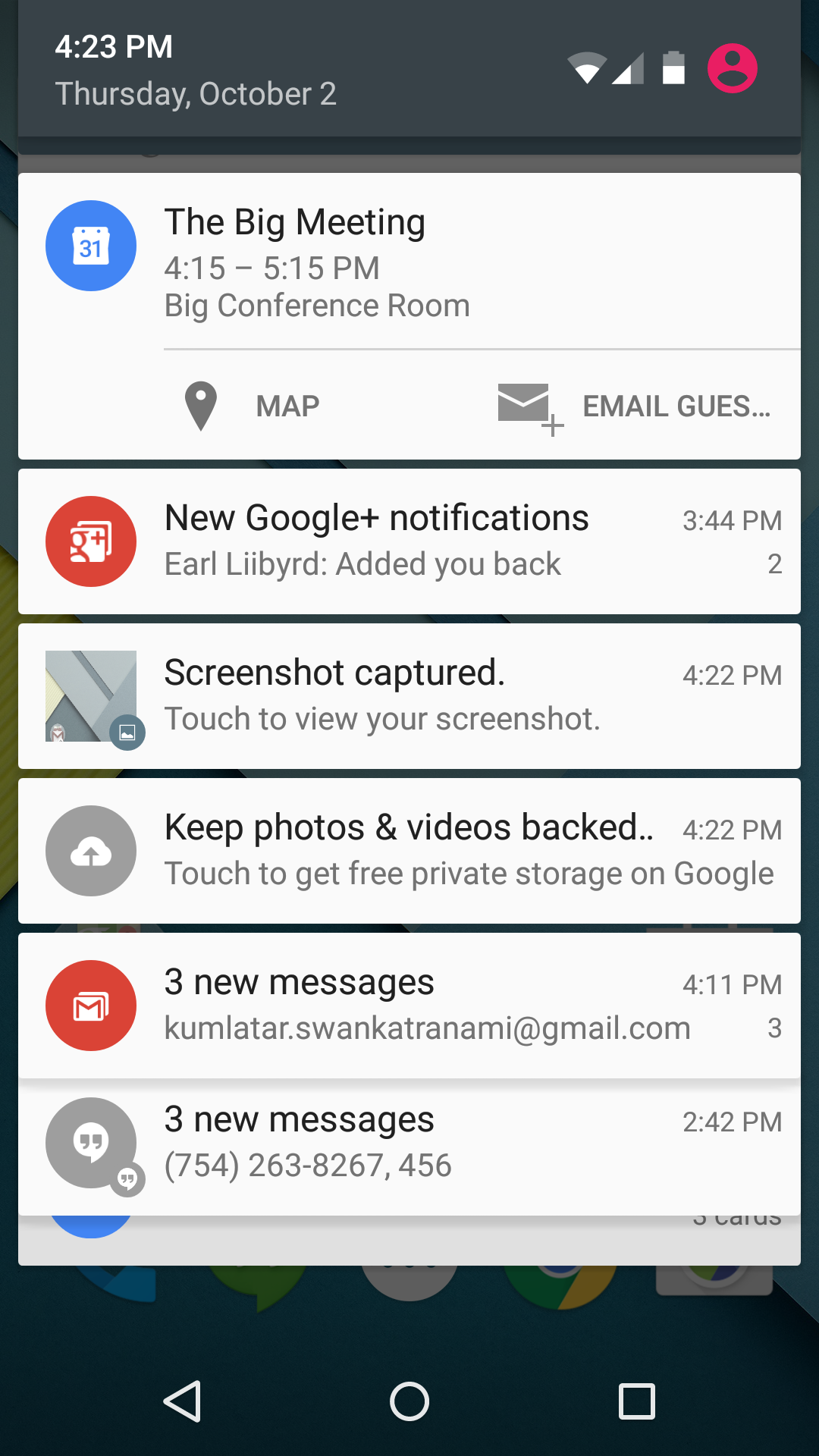
See also

1. [Android Design: Notifications](https://developer.android.com/design/patterns/notifications.html)

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.



**Figure 1.** Notifications in the notification area.



**Figure 2.** Notifications in the notification drawer.

**Note:** Except where noted, this guide refers to the [NotificationCompat.Builder](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html) class in the version 4[Support Library](https://developer.android.com/tools/support-library/index.html). The class [Notification.Builder](https://developer.android.com/reference/android/app/Notification.Builder.html) was added in Android 3.0 (API level 11).

Design Considerations

Notifications, as an important part of the Android user interface, have their own design guidelines. The material design changes introduced in Android 5.0 (API level 21) are of particular importance, and you should review the [Material Design](https://developer.android.com/training/material/index.html) training for more information. To learn how to design notifications and their interactions, read the [Notifications](https://developer.android.com/design/patterns/notifications.html) design guide.

Creating a Notification

You specify the UI information and actions for a notification in a [NotificationCompat.Builder](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html) object. To create the notification itself, you call [NotificationCompat.Builder.build()](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html#build()), which returns a [Notification](https://developer.android.com/reference/android/app/Notification.html) object containing your specifications. To issue the notification, you pass the [Notification](https://developer.android.com/reference/android/app/Notification.html) object to the system by calling [NotificationManager.notify()](https://developer.android.com/reference/android/app/NotificationManager.html#notify(int, android.app.Notification)).

Required notification contents

A [Notification](https://developer.android.com/reference/android/app/Notification.html) object *must* contain the following:

* A small icon, set by [setSmallIcon()](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html#setSmallIcon(int))
* A title, set by [setContentTitle()](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html#setContentTitle(java.lang.CharSequence))
* Detail text, set by [setContentText()](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html#setContentText(java.lang.CharSequence))

Optional notification contents and settings

All other notification settings and contents are optional. To learn more about them, see the reference documentation for [NotificationCompat.Builder](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html).

Notification actions

Although they're optional, you should add at least one action to your notification. An action allows users to go directly from the notification to an[Activity](https://developer.android.com/reference/android/app/Activity.html) in your application, where they can look at one or more events or do further work.

A notification can provide multiple actions. You should always define the action that's triggered when the user clicks the notification; usually this action opens an [Activity](https://developer.android.com/reference/android/app/Activity.html) in your application. You can also add buttons to the notification that perform additional actions such as snoozing an alarm or responding immediately to a text message; this feature is available as of Android 4.1. If you use additional action buttons, you must also make their functionality available in an [Activity](https://developer.android.com/reference/android/app/Activity.html) in your app; see the section [Handling compatibility](https://developer.android.com/guide/topics/ui/notifiers/notifications.html#Compatibility) for more details.

Inside a [Notification](https://developer.android.com/reference/android/app/Notification.html), the action itself is defined by a [PendingIntent](https://developer.android.com/reference/android/app/PendingIntent.html) containing an [Intent](https://developer.android.com/reference/android/content/Intent.html) that starts an [Activity](https://developer.android.com/reference/android/app/Activity.html) in your application. To associate the[PendingIntent](https://developer.android.com/reference/android/app/PendingIntent.html) with a gesture, call the appropriate method of [NotificationCompat.Builder](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html). For example, if you want to start [Activity](https://developer.android.com/reference/android/app/Activity.html) when the user clicks the notification text in the notification drawer, you add the [PendingIntent](https://developer.android.com/reference/android/app/PendingIntent.html) by calling [setContentIntent()](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html#setContentIntent(android.app.PendingIntent)).

Starting an [Activity](https://developer.android.com/reference/android/app/Activity.html) when the user clicks the notification is the most common action scenario. You can also start an [Activity](https://developer.android.com/reference/android/app/Activity.html) when the user dismisses a notification. In Android 4.1 and later, you can start an [Activity](https://developer.android.com/reference/android/app/Activity.html) from an action button. To learn more, read the reference guide for[NotificationCompat.Builder](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html).

Notification priority

If you wish, you can set the priority of a notification. The priority acts as a hint to the device UI about how the notification should be displayed. To set a notification's priority, call [NotificationCompat.Builder.setPriority()](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html#setPriority(int)) and pass in one of the [NotificationCompat](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.html) priority constants. There are five priority levels, ranging from [PRIORITY\_MIN](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.html#PRIORITY_MIN) (-2) to [PRIORITY\_MAX](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.html#PRIORITY_MAX) (2); if not set, the priority defaults to [PRIORITY\_DEFAULT](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.html#PRIORITY_DEFAULT) (0).

For information about setting an appropriate priority level, see "Correctly set and manage notification priority" in the [Notifications](https://developer.android.com/design/patterns/notifications.html) Design guide.

Creating a simple notification

The following snippet illustrates a simple notification that specifies an activity to open when the user clicks the notification. Notice that the code creates a [TaskStackBuilder](https://developer.android.com/reference/android/support/v4/app/TaskStackBuilder.html) object and uses it to create the [PendingIntent](https://developer.android.com/reference/android/app/PendingIntent.html) for the action. This pattern is explained in more detail in the section[Preserving Navigation when Starting an Activity](https://developer.android.com/guide/topics/ui/notifiers/notifications.html#NotificationResponse):

NotificationCompat.Builder mBuilder =  
        new NotificationCompat.Builder(this)  
        .setSmallIcon(R.drawable.notification\_icon)  
        .setContentTitle("My notification")  
        .setContentText("Hello World!");  
// Creates an explicit intent for an Activity in your app  
Intent resultIntent = new Intent(this, ResultActivity.class);  
  
// The stack builder object 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.  
TaskStackBuilder stackBuilder = TaskStackBuilder.create(this);  
// Adds the back stack for the Intent (but not the Intent itself)  
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);  
NotificationManager mNotificationManager =  
    (NotificationManager) getSystemService(Context.NOTIFICATION\_SERVICE);  
// mId allows you to update the notification later on.  
mNotificationManager.notify(mId, mBuilder.build());

That's it. Your user has now been notified.

Applying an expanded layout to a notification

To have a notification appear in an expanded view, first create a [NotificationCompat.Builder](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html) object with the normal view options you want. Next, call[Builder.setStyle()](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html#setStyle(android.support.v4.app.NotificationCompat.Style)) with an expanded layout object as its argument.

Remember that expanded notifications are not available on platforms prior to Android 4.1. To learn how to handle notifications for Android 4.1 and for earlier platforms, read the section [Handling compatibility](https://developer.android.com/guide/topics/ui/notifiers/notifications.html#Compatibility).

For example, the following code snippet demonstrates how to alter the notification created in the previous snippet to use the expanded layout:

NotificationCompat.Builder mBuilder = new NotificationCompat.Builder(this)  
    .setSmallIcon(R.drawable.notification\_icon)  
    .setContentTitle("Event tracker")  
    .setContentText("Events received")  
NotificationCompat.InboxStyle inboxStyle =  
        new NotificationCompat.InboxStyle();  
String[] events = new String[6];  
// Sets a title for the Inbox in expanded layout  
inboxStyle.setBigContentTitle("Event tracker details:");  
...  
// Moves events into the expanded layout  
for (int i=0; i < events.length; i++) {  
  
    inboxStyle.addLine(events[i]);  
}  
// Moves the expanded layout object into the notification object.  
mBuilder.setStyle(inBoxStyle);  
...  
// Issue the notification here.

Handling compatibility

Not all notification features are available for a particular version, even though the methods to set them are in the support library class[NotificationCompat.Builder](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html). For example, action buttons, which depend on expanded notifications, only appear on Android 4.1 and higher, because expanded notifications themselves are only available on Android 4.1 and higher.

To ensure the best compatibility, create notifications with [NotificationCompat](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.html) and its subclasses, particularly [NotificationCompat.Builder](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html). In addition, follow this process when you implement a notification:

1. Provide all of the notification's functionality to all users, regardless of the version they're using. To do this, verify that all of the functionality is available from an [Activity](https://developer.android.com/reference/android/app/Activity.html) in your app. You may want to add a new [Activity](https://developer.android.com/reference/android/app/Activity.html) to do this.

For example, if you want to use [addAction()](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html#addAction(android.support.v4.app.NotificationCompat.Action)) to provide a control that stops and starts media playback, first implement this control in an [Activity](https://developer.android.com/reference/android/app/Activity.html) in your app.

1. Ensure that all users can get to the functionality in the [Activity](https://developer.android.com/reference/android/app/Activity.html), by having it start when users click the notification. To do this, create a[PendingIntent](https://developer.android.com/reference/android/app/PendingIntent.html) for the [Activity](https://developer.android.com/reference/android/app/Activity.html). Call [setContentIntent()](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html#setContentIntent(android.app.PendingIntent)) to add the [PendingIntent](https://developer.android.com/reference/android/app/PendingIntent.html) to the notification.
2. Now add the expanded notification features you want to use to the notification. Remember that any functionality you add also has to be available in the [Activity](https://developer.android.com/reference/android/app/Activity.html) that starts when users click the notification.

Managing Notifications

When you need to issue a notification multiple times for the same type of event, you should avoid making a completely new notification. Instead, you should consider updating a previous notification, either by changing some of its values or by adding to it, or both.

For example, Gmail notifies the user that new emails have arrived by increasing its count of unread messages and by adding a summary of each email to the notification. This is called "stacking" the notification; it's described in more detail in the [Notifications](https://developer.android.com/design/patterns/notifications.html) Design guide.

**Note:** This Gmail feature requires the "inbox" expanded layout, which is part of the expanded notification feature.

The following section describes how to update notifications and also how to remove them.

Updating notifications

To set up a notification so it can be updated, issue it with a notification ID by calling [NotificationManager.notify()](https://developer.android.com/reference/android/app/NotificationManager.html#notify(int, android.app.Notification)). To update this notification once you've issued it, update or create a [NotificationCompat.Builder](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html) object, build a [Notification](https://developer.android.com/reference/android/app/Notification.html) object from it, and issue the [Notification](https://developer.android.com/reference/android/app/Notification.html) with the same ID you used previously. If the previous notification is still visible, the system updates it from the contents of the [Notification](https://developer.android.com/reference/android/app/Notification.html) object. If the previous notification has been dismissed, a new notification is created instead.

The following snippet demonstrates a notification that is updated to reflect the number of events that have occurred. It stacks the notification, showing a summary:

mNotificationManager =  
        (NotificationManager) getSystemService(Context.NOTIFICATION\_SERVICE);  
// Sets an ID for the notification, so it can be updated  
int notifyID = 1;  
mNotifyBuilder = new NotificationCompat.Builder(this)  
    .setContentTitle("New Message")  
    .setContentText("You've received new messages.")  
    .setSmallIcon(R.drawable.ic\_notify\_status)  
numMessages = 0;  
// Start of a loop that processes data and then notifies the user  
...  
    mNotifyBuilder.setContentText(currentText)  
        .setNumber(++numMessages);  
    // Because the ID remains unchanged, the existing notification is  
    // updated.  
    mNotificationManager.notify(  
            notifyID,  
            mNotifyBuilder.build());  
...

Removing notifications

Notifications remain visible until one of the following happens:

* The user dismisses the notification either individually or by using "Clear All" (if the notification can be cleared).
* The user clicks the notification, and you called [setAutoCancel()](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html#setAutoCancel(boolean)) when you created the notification.
* You call [cancel()](https://developer.android.com/reference/android/app/NotificationManager.html#cancel(int)) for a specific notification ID. This method also deletes ongoing notifications.
* You call [cancelAll()](https://developer.android.com/reference/android/app/NotificationManager.html#cancelAll()), which removes all of the notifications you previously issued.

Preserving Navigation when Starting an Activity

When you start an [Activity](https://developer.android.com/reference/android/app/Activity.html) from a notification, you must preserve the user's expected navigation experience. Clicking *Back* should take the user back through the application's normal work flow to the Home screen, and clicking *Recents* should show the [Activity](https://developer.android.com/reference/android/app/Activity.html) as a separate task. To preserve the navigation experience, you should start the [Activity](https://developer.android.com/reference/android/app/Activity.html) in a fresh task. How you set up the [PendingIntent](https://developer.android.com/reference/android/app/PendingIntent.html) to give you a fresh task depends on the nature of the [Activity](https://developer.android.com/reference/android/app/Activity.html) you're starting. There are two general situations:

Regular activity

You're starting an [Activity](https://developer.android.com/reference/android/app/Activity.html) that's part of the application's normal workflow. In this situation, set up the [PendingIntent](https://developer.android.com/reference/android/app/PendingIntent.html) to start a fresh task, and provide the [PendingIntent](https://developer.android.com/reference/android/app/PendingIntent.html) with a back stack that reproduces the application's normal *Back* behavior.

Notifications from the Gmail app demonstrate this. When you click a notification for a single email message, you see the message itself. Touching **Back** takes you backwards through Gmail to the Home screen, just as if you had entered Gmail from the Home screen rather than entering it from a notification.

This happens regardless of the application you were in when you touched the notification. For example, if you're in Gmail composing a message, and you click a notification for a single email, you go immediately to that email. Touching *Back* takes you to the inbox and then the Home screen, rather than taking you to the message you were composing.

Special activity

The user only sees this [Activity](https://developer.android.com/reference/android/app/Activity.html) if it's started from a notification. In a sense, the [Activity](https://developer.android.com/reference/android/app/Activity.html) extends the notification by providing information that would be hard to display in the notification itself. For this situation, set up the [PendingIntent](https://developer.android.com/reference/android/app/PendingIntent.html) to start in a fresh task. There's no need to create a back stack, though, because the started [Activity](https://developer.android.com/reference/android/app/Activity.html) isn't part of the application's activity flow. Clicking *Back* will still take the user to the Home screen.

Setting up a regular activity PendingIntent

To set up a [PendingIntent](https://developer.android.com/reference/android/app/PendingIntent.html) that starts a direct entry [Activity](https://developer.android.com/reference/android/app/Activity.html), follow these steps:

1. Define your application's [Activity](https://developer.android.com/reference/android/app/Activity.html) hierarchy in the manifest.
   1. Add support for Android 4.0.3 and earlier. To do this, specify the parent of the [Activity](https://developer.android.com/reference/android/app/Activity.html) you're starting by adding a [<meta-data>](https://developer.android.com/guide/topics/manifest/meta-data-element.html) element as the child of the [<activity>](https://developer.android.com/guide/topics/manifest/activity-element.html).

For this element, set [android:name](https://developer.android.com/guide/topics/manifest/meta-data-element.html#nm)="android.support.PARENT\_ACTIVITY". Set [android:value](https://developer.android.com/guide/topics/manifest/meta-data-element.html#val)="<parent\_activity\_name>" where<parent\_activity\_name> is the value of [android:name](https://developer.android.com/guide/topics/manifest/meta-data-element.html#nm) for the parent [<activity>](https://developer.android.com/guide/topics/manifest/activity-element.html) element. See the following XML for an example.

* 1. Also add support for Android 4.1 and later. To do this, add the [android:parentActivityName](https://developer.android.com/guide/topics/manifest/activity-element.html#parent) attribute to the [<activity>](https://developer.android.com/guide/topics/manifest/activity-element.html) element of the [Activity](https://developer.android.com/reference/android/app/Activity.html)you're starting.

The final XML should look like this:

<activity  
    android:name=".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=".ResultActivity"  
    android:parentActivityName=".MainActivity">  
    <meta-data  
        android:name="android.support.PARENT\_ACTIVITY"  
        android:value=".MainActivity"/>  
</activity>

1. Create a back stack based on the [Intent](https://developer.android.com/reference/android/content/Intent.html) that starts the [Activity](https://developer.android.com/reference/android/app/Activity.html):
   1. Create the [Intent](https://developer.android.com/reference/android/content/Intent.html) to start the [Activity](https://developer.android.com/reference/android/app/Activity.html).
   2. Create a stack builder by calling [TaskStackBuilder.create()](https://developer.android.com/reference/android/app/TaskStackBuilder.html#create(android.content.Context)).
   3. Add the back stack to the stack builder by calling [addParentStack()](https://developer.android.com/reference/android/support/v4/app/TaskStackBuilder.html#addParentStack(android.app.Activity)). For each [Activity](https://developer.android.com/reference/android/app/Activity.html) in the hierarchy you've defined in the manifest, the back stack contains an [Intent](https://developer.android.com/reference/android/content/Intent.html) object that starts the [Activity](https://developer.android.com/reference/android/app/Activity.html). This method also adds flags that start the stack in a fresh task.

**Note:** Although the argument to [addParentStack()](https://developer.android.com/reference/android/support/v4/app/TaskStackBuilder.html#addParentStack(android.app.Activity)) is a reference to the started [Activity](https://developer.android.com/reference/android/app/Activity.html), the method call doesn't add the [Intent](https://developer.android.com/reference/android/content/Intent.html) that starts the [Activity](https://developer.android.com/reference/android/app/Activity.html). Instead, that's taken care of in the next step.

* 1. Add the [Intent](https://developer.android.com/reference/android/content/Intent.html) that starts the [Activity](https://developer.android.com/reference/android/app/Activity.html) from the notification, by calling [addNextIntent()](https://developer.android.com/reference/android/support/v4/app/TaskStackBuilder.html#addNextIntent(android.content.Intent)). Pass the [Intent](https://developer.android.com/reference/android/content/Intent.html) you created in the first step as the argument to [addNextIntent()](https://developer.android.com/reference/android/support/v4/app/TaskStackBuilder.html#addNextIntent(android.content.Intent)).
  2. If you need to, add arguments to [Intent](https://developer.android.com/reference/android/content/Intent.html) objects on the stack by calling [TaskStackBuilder.editIntentAt()](https://developer.android.com/reference/android/support/v4/app/TaskStackBuilder.html#editIntentAt(int)). This is sometimes necessary to ensure that the target [Activity](https://developer.android.com/reference/android/app/Activity.html) displays meaningful data when the user navigates to it using *Back*.
  3. Get a [PendingIntent](https://developer.android.com/reference/android/app/PendingIntent.html) for this back stack by calling [getPendingIntent()](https://developer.android.com/reference/android/support/v4/app/TaskStackBuilder.html#getPendingIntent(int, int)). You can then use this [PendingIntent](https://developer.android.com/reference/android/app/PendingIntent.html) as the argument to[setContentIntent()](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html#setContentIntent(android.app.PendingIntent)).

The following code snippet demonstrates the process:

...  
Intent resultIntent = new Intent(this, ResultActivity.class);  
TaskStackBuilder stackBuilder = TaskStackBuilder.create(this);  
// Adds the back stack  
stackBuilder.addParentStack(ResultActivity.class);  
// Adds the Intent to the top of the stack  
stackBuilder.addNextIntent(resultIntent);  
// Gets a PendingIntent containing the entire back stack  
PendingIntent resultPendingIntent =  
        stackBuilder.getPendingIntent(0, PendingIntent.FLAG\_UPDATE\_CURRENT);  
...  
NotificationCompat.Builder builder = new NotificationCompat.Builder(this);  
builder.setContentIntent(resultPendingIntent);  
NotificationManager mNotificationManager =  
    (NotificationManager) getSystemService(Context.NOTIFICATION\_SERVICE);  
mNotificationManager.notify(id, builder.build());

Setting up a special activity PendingIntent

The following section describes how to set up a special activity [PendingIntent](https://developer.android.com/reference/android/app/PendingIntent.html).

A special [Activity](https://developer.android.com/reference/android/app/Activity.html) doesn't need a back stack, so you don't have to define its [Activity](https://developer.android.com/reference/android/app/Activity.html) hierarchy in the manifest, and you don't have to call[addParentStack()](https://developer.android.com/reference/android/support/v4/app/TaskStackBuilder.html#addParentStack(android.app.Activity)) to build a back stack. Instead, use the manifest to set up the [Activity](https://developer.android.com/reference/android/app/Activity.html) task options, and create the [PendingIntent](https://developer.android.com/reference/android/app/PendingIntent.html) by calling[getActivity()](https://developer.android.com/reference/android/app/PendingIntent.html#getActivity(android.content.Context, int, android.content.Intent, int)):

1. In your manifest, add the following attributes to the [<activity>](https://developer.android.com/guide/topics/manifest/activity-element.html) element for the [Activity](https://developer.android.com/reference/android/app/Activity.html)

[android:name](https://developer.android.com/guide/topics/manifest/activity-element.html#nm)="*activityclass*"

The activity's fully-qualified class name.

[android:taskAffinity](https://developer.android.com/guide/topics/manifest/activity-element.html#aff)=""

Combined with the [FLAG\_ACTIVITY\_NEW\_TASK](https://developer.android.com/reference/android/content/Intent.html#FLAG_ACTIVITY_NEW_TASK) flag that you set in code, this ensures that this [Activity](https://developer.android.com/reference/android/app/Activity.html) doesn't go into the application's default task. Any existing tasks that have the application's default affinity are not affected.

[android:excludeFromRecents](https://developer.android.com/guide/topics/manifest/activity-element.html#exclude)="true"

Excludes the new task from *Recents*, so that the user can't accidentally navigate back to it.

This snippet shows the element:

<activity  
    android:name=".ResultActivity"  
...  
    android:launchMode="singleTask"  
    android:taskAffinity=""  
    android:excludeFromRecents="true">  
</activity>  
...

1. Build and issue the notification:
   1. Create an [Intent](https://developer.android.com/reference/android/content/Intent.html) that starts the [Activity](https://developer.android.com/reference/android/app/Activity.html).
   2. Set the [Activity](https://developer.android.com/reference/android/app/Activity.html) to start in a new, empty task by calling [setFlags()](https://developer.android.com/reference/android/content/Intent.html#setFlags(int)) with the flags [FLAG\_ACTIVITY\_NEW\_TASK](https://developer.android.com/reference/android/content/Intent.html#FLAG_ACTIVITY_NEW_TASK) and [FLAG\_ACTIVITY\_CLEAR\_TASK](https://developer.android.com/reference/android/content/Intent.html#FLAG_ACTIVITY_CLEAR_TASK).
   3. Set any other options you need for the [Intent](https://developer.android.com/reference/android/content/Intent.html).
   4. Create a [PendingIntent](https://developer.android.com/reference/android/app/PendingIntent.html) from the [Intent](https://developer.android.com/reference/android/content/Intent.html) by calling [getActivity()](https://developer.android.com/reference/android/app/PendingIntent.html#getActivity(android.content.Context, int, android.content.Intent, int)). You can then use this [PendingIntent](https://developer.android.com/reference/android/app/PendingIntent.html) as the argument to[setContentIntent()](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html#setContentIntent(android.app.PendingIntent)).

The following code snippet demonstrates the process:

// Instantiate a Builder object.  
NotificationCompat.Builder builder = new NotificationCompat.Builder(this);  
// Creates an Intent for the Activity  
Intent notifyIntent =  
        new Intent(this, ResultActivity.class);  
// Sets the Activity to start in a new, empty task  
notifyIntent.setFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK  
                        | Intent.FLAG\_ACTIVITY\_CLEAR\_TASK);  
// Creates the PendingIntent  
PendingIntent notifyPendingIntent =  
        PendingIntent.getActivity(  
        this,  
        0,  
        notifyIntent,  
        PendingIntent.FLAG\_UPDATE\_CURRENT  
);  
  
// Puts the PendingIntent into the notification builder  
builder.setContentIntent(notifyPendingIntent);  
// Notifications are issued by sending them to the  
// NotificationManager system service.  
NotificationManager mNotificationManager =  
    (NotificationManager) getSystemService(Context.NOTIFICATION\_SERVICE);  
// Builds an anonymous Notification object from the builder, and  
// passes it to the NotificationManager  
mNotificationManager.notify(id, builder.build());

Displaying Progress in a Notification

Notifications can include an animated progress indicator that shows users the status of an ongoing operation. If you can estimate how long the operation takes and how much of it is complete at any time, use the "determinate" form of the indicator (a progress bar). If you can't estimate the length of the operation, use the "indeterminate" form of the indicator (an activity indicator).

Progress indicators are displayed with the platform's implementation of the [ProgressBar](https://developer.android.com/reference/android/widget/ProgressBar.html) class.

To use a progress indicator on platforms starting with Android 4.0, call [setProgress()](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html#setProgress(int, int, boolean)). For previous versions, you must create your own custom notification layout that includes a [ProgressBar](https://developer.android.com/reference/android/widget/ProgressBar.html) view.

The following sections describe how to display progress in a notification using [setProgress()](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html#setProgress(int, int, boolean)).

Displaying a fixed-duration progress indicator

To display a determinate progress bar, add the bar to your notification by calling [setProgress(max, progress, false)](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html#setProgress(int, int, boolean)) and then issue the notification. As your operation proceeds, increment progress, and update the notification. At the end of the operation, progress should equal max. A common way to call [setProgress()](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html#setProgress(int, int, boolean)) is to set max to 100 and then increment progress as a "percent complete" value for the operation.

You can either leave the progress bar showing when the operation is done, or remove it. In either case, remember to update the notification text to show that the operation is complete. To remove the progress bar, call [setProgress(0, 0, false)](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html#setProgress(int, int, boolean)). For example:

...  
mNotifyManager =  
        (NotificationManager) getSystemService(Context.NOTIFICATION\_SERVICE);  
mBuilder = new NotificationCompat.Builder(this);  
mBuilder.setContentTitle("Picture Download")  
    .setContentText("Download in progress")  
    .setSmallIcon(R.drawable.ic\_notification);  
// Start a lengthy operation in a background thread  
new Thread(  
    new Runnable() {  
        @Override  
        public void run() {  
            int incr;  
            // Do the "lengthy" operation 20 times  
            for (incr = 0; incr <= 100; incr+=5) {  
                    // Sets the progress indicator to a max value, the  
                    // current completion percentage, and "determinate"  
                    // state  
                    mBuilder.setProgress(100, incr, false);  
                    // Displays the progress bar for the first time.  
                    mNotifyManager.notify(0, mBuilder.build());  
                        // Sleeps the thread, simulating an operation  
                        // that takes time  
                        try {  
                            // Sleep for 5 seconds  
                            Thread.sleep(5\*1000);  
                        } catch (InterruptedException e) {  
                            Log.d(TAG, "sleep failure");  
                        }  
            }  
            // When the loop is finished, updates the notification  
            mBuilder.setContentText("Download complete")  
            // Removes the progress bar  
                    .setProgress(0,0,false);  
            mNotifyManager.notify(ID, mBuilder.build());  
        }  
    }  
// Starts the thread by calling the run() method in its Runnable  
).start();

Displaying a continuing activity indicator

To display an indeterminate activity indicator, add it to your notification with [setProgress(0, 0, true)](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html#setProgress(int, int, boolean)) (the first two arguments are ignored), and issue the notification. The result is an indicator that has the same style as a progress bar, except that its animation is ongoing.

Issue the notification at the beginning of the operation. The animation will run until you modify your notification. When the operation is done, call[setProgress(0, 0, false)](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html#setProgress(int, int, boolean)) and then update the notification to remove the activity indicator. Always do this; otherwise, the animation will run even when the operation is complete. Also remember to change the notification text to indicate that the operation is complete.

To see how activity indicators work, refer to the preceding snippet. Locate the following lines:

// Sets the progress indicator to a max value, the current completion  
// percentage, and "determinate" state  
mBuilder.setProgress(100, incr, false);  
// Issues the notification  
mNotifyManager.notify(0, mBuilder.build());

Replace the lines you've found with the following lines:

 // Sets an activity indicator for an operation of indeterminate length  
mBuilder.setProgress(0, 0, true);  
// Issues the notification  
mNotifyManager.notify(0, mBuilder.build());

Notification Metadata

Notifications may be sorted according to metadata that you assign with the following [NotificationCompat.Builder](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html) methods:

* [setCategory()](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html#setCategory(java.lang.String)) tells the system how to handle your app notifications when the device is in Priority mode (for example, if your notification represents an incoming call, instant message, or alarm).
* [setPriority()](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html#setPriority(int)) causes notifications with the priority field set to PRIORITY\_MAX or PRIORITY\_HIGH to appear in a small floating window if the notification also has sound or vibration.
* [addPerson()](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html#addPerson(java.lang.String)) allows you to add a list of people to a notification. Your app can use this to signal to the system that it should group together notifications from the specified people, or rank notifications from these people as being more important.



**Figure 3.** Fullscreen activity showing a heads-up notification

Heads-up Notifications

With Android 5.0 (API level 21), notifications can appear in a small floating window (also called a *heads-up notification*) when the device is active (that is, the device is unlocked and its screen is on). These notifications appear similar to the compact form of your notification, except that the heads-up notification also shows action buttons. Users can act on, or dismiss, a heads-up notification without leaving the current app.

Examples of conditions that may trigger heads-up notifications include:

* The user's activity is in fullscreen mode (the app uses [fullScreenIntent](https://developer.android.com/reference/android/app/Notification.html#fullScreenIntent)), or
* The notification has high priority and uses ringtones or vibrations

Lock Screen Notifications

With the release of Android 5.0 (API level 21), notifications may now appear on the lock screen. Your app can use this functionality to provide media playback controls and other common actions. Users can choose via Settings whether to display notifications on the lock screen, and you can designate whether a notification from your app is visible on the lock screen.

Setting Visibility

Your app can control the level of detail visible in notifications displayed on a secure lock screen. You call [setVisibility()](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html#setVisibility(int)) and specify one of the following values:

* [VISIBILITY\_PUBLIC](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.html#VISIBILITY_PUBLIC) shows the notification's full content.
* [VISIBILITY\_SECRET](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.html#VISIBILITY_SECRET) doesn't show any part of this notification on the lock screen.
* [VISIBILITY\_PRIVATE](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.html#VISIBILITY_PRIVATE) shows basic information, such as the notification's icon and the content title, but hides the notification's full content.

When [VISIBILITY\_PRIVATE](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.html#VISIBILITY_PRIVATE) is set, you can also provide an alternate version of the notification content which hides certain details. For example, an SMS app might display a notification that shows *You have 3 new text messages*, but hides the message contents and senders. To provide this alternative notification, first create the replacement notification using [NotificationCompat.Builder](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html). When you create the private notification object, attach the replacement notification to it through the [setPublicVersion()](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html#setPublicVersion(android.app.Notification)) method.

Controlling Media Playback on the Lock Screen

In Android 5.0 (API level 21) the lock screen no longer displays media controls based on the [RemoteControlClient](https://developer.android.com/reference/android/media/RemoteControlClient.html), which is now deprecated. Instead, use the [Notification.MediaStyle](https://developer.android.com/reference/android/app/Notification.MediaStyle.html) template with the [addAction()](https://developer.android.com/reference/android/app/Notification.Builder.html#addAction(android.app.Notification.Action)) method, which converts actions into clickable icons.

**Note:** The template and the [addAction()](https://developer.android.com/reference/android/app/Notification.Builder.html#addAction(android.app.Notification.Action)) method are not included in the support library, so these features run in Android 5.0 and higher only.

To display media playback controls on the lock screen in Android 5.0, set the visibility to [VISIBILITY\_PUBLIC](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.html#VISIBILITY_PUBLIC), as described above. Then add the actions and set the [Notification.MediaStyle](https://developer.android.com/reference/android/app/Notification.MediaStyle.html) template, as described in the following sample code:

Notification notification = new Notification.Builder(context)  
    // Show controls on lock screen even when user hides sensitive content.  
    .setVisibility(Notification.VISIBILITY\_PUBLIC)  
    .setSmallIcon(R.drawable.ic\_stat\_player)  
    // Add media control buttons that invoke intents in your media service  
    .addAction(R.drawable.ic\_prev, "Previous", prevPendingIntent) // #0  
    .addAction(R.drawable.ic\_pause, "Pause", pausePendingIntent)  // #1  
    .addAction(R.drawable.ic\_next, "Next", nextPendingIntent)     // #2  
    // Apply the media style template  
    .setStyle(new Notification.MediaStyle()  
    .setShowActionsInCompactView(1 /\* #1: pause button \*/)  
    .setMediaSession(mMediaSession.getSessionToken())  
    .setContentTitle("Wonderful music")  
    .setContentText("My Awesome Band")  
    .setLargeIcon(albumArtBitmap)  
    .build();

**Note:** The deprecation of [RemoteControlClient](https://developer.android.com/reference/android/media/RemoteControlClient.html) has further implications for controlling media. See [Media Playback Control](https://developer.android.com/about/versions/android-5.0.html#MediaPlaybackControl) for more information about the new APIs for managing the media session and controlling playback.

Custom Notification Layouts

The notifications framework allows you to define a custom notification layout, which defines the notification's appearance in a [RemoteViews](https://developer.android.com/reference/android/widget/RemoteViews.html) object. Custom layout notifications are similar to normal notifications, but they're based on a [RemoteViews](https://developer.android.com/reference/android/widget/RemoteViews.html) defined in a XML layout file.

The height available for a custom notification layout depends on the notification view. Normal view layouts are limited to 64 dp, and expanded view layouts are limited to 256 dp.

To define a custom notification layout, start by instantiating a [RemoteViews](https://developer.android.com/reference/android/widget/RemoteViews.html) object that inflates an XML layout file. Then, instead of calling methods such as [setContentTitle()](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html#setContentTitle(java.lang.CharSequence)), call [setContent()](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html#setContent(android.widget.RemoteViews)). To set content details in the custom notification, use the methods in [RemoteViews](https://developer.android.com/reference/android/widget/RemoteViews.html) to set the values of the view's children:

1. Create an XML layout for the notification in a separate file. You can use any file name you wish, but you must use the extension .xml
2. In your app, use [RemoteViews](https://developer.android.com/reference/android/widget/RemoteViews.html) methods to define your notification's icons and text. Put this [RemoteViews](https://developer.android.com/reference/android/widget/RemoteViews.html) object into your[NotificationCompat.Builder](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html) by calling [setContent()](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html#setContent(android.widget.RemoteViews)). Avoid setting a background [Drawable](https://developer.android.com/reference/android/graphics/drawable/Drawable.html) on your [RemoteViews](https://developer.android.com/reference/android/widget/RemoteViews.html) object, because your text color may become unreadable.

The [RemoteViews](https://developer.android.com/reference/android/widget/RemoteViews.html) class also includes methods that you can use to easily add a [Chronometer](https://developer.android.com/reference/android/widget/Chronometer.html) or [ProgressBar](https://developer.android.com/reference/android/widget/ProgressBar.html) to your notification's layout. For more information about creating custom layouts for your notification, refer to the [RemoteViews](https://developer.android.com/reference/android/widget/RemoteViews.html) reference documentation.

**Caution:** When you use a custom notification layout, take special care to ensure that your custom layout works with different device orientations and resolutions. While this advice applies to all View layouts, it's especially important for notifications because the space in the notification drawer is very restricted. Don't make your custom layout too complex, and be sure to test it in various configurations.

Using style resources for custom notification text

Always use style resources for the text of a custom notification. The background color of the notification can vary across different devices and versions, and using style resources helps you account for this. Starting in Android 2.3, the system defined a style for the standard notification layout text. If you use the same style in applications that target Android 2.3 or higher, you'll ensure that your text is visible against the display background.

Get news & tips SIGN UP FOR THE NEWSLETTER

[Blog](http://android-developers.blogspot.com/) [Support](https://developer.android.com/support.html)

[ANDROID DEVELOPERS ON YOUTUBE](https://www.youtube.com/user/androiddevelopers) [ANDROID DEVELOPERS ON GOOGLE+](https://plus.google.com/+AndroidDevelopers) [ANDROID DEVELOPERS ON TWITTER](https://twitter.com/AndroidDev)

Except as noted, this content is licensed under [Creative Commons Attribution 2.5](http://creativecommons.org/licenses/by/2.5/). For details and restrictions, see the [Content License](https://developer.android.com/license.html).

[About Android](https://developer.android.com/about/android.html) [Auto](https://developer.android.com/auto/index.html) [TV](https://developer.android.com/tv/index.html) [Wear](https://developer.android.com/wear/index.html) [Legal](https://developer.android.com/legal.html)

Top of Form



Bottom of Form

This site uses cookies to store your preferences for site-specific language and display options.