Notifications Overview

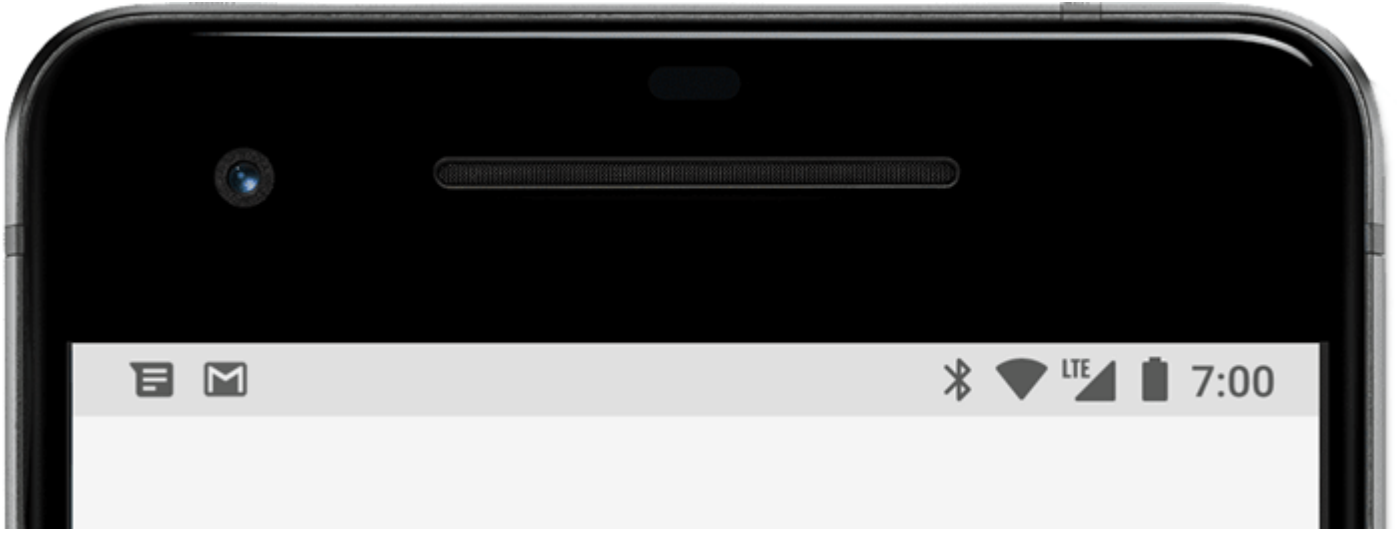
A notification is a message that Android displays outside your app's UI to provide the user with reminders, communication from other people, or other timely information from your app. Users can tap the notification to open your app or take an action directly from the notification.

Appearances on a device

Notifications appear to users in different locations and formats, such as an icon in the status bar, a more detailed entry in the notification drawer, as a badge on the app's icon, and on paired wearables automatically.

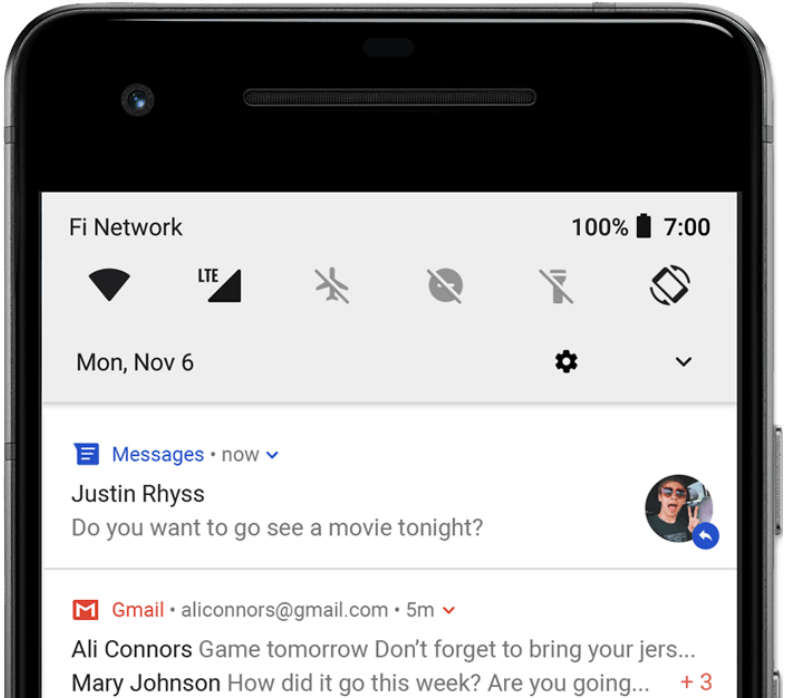
Status bar and notification drawer

When you issue a notification, it first appears as an icon in the status bar.



Status bar and notification drawer

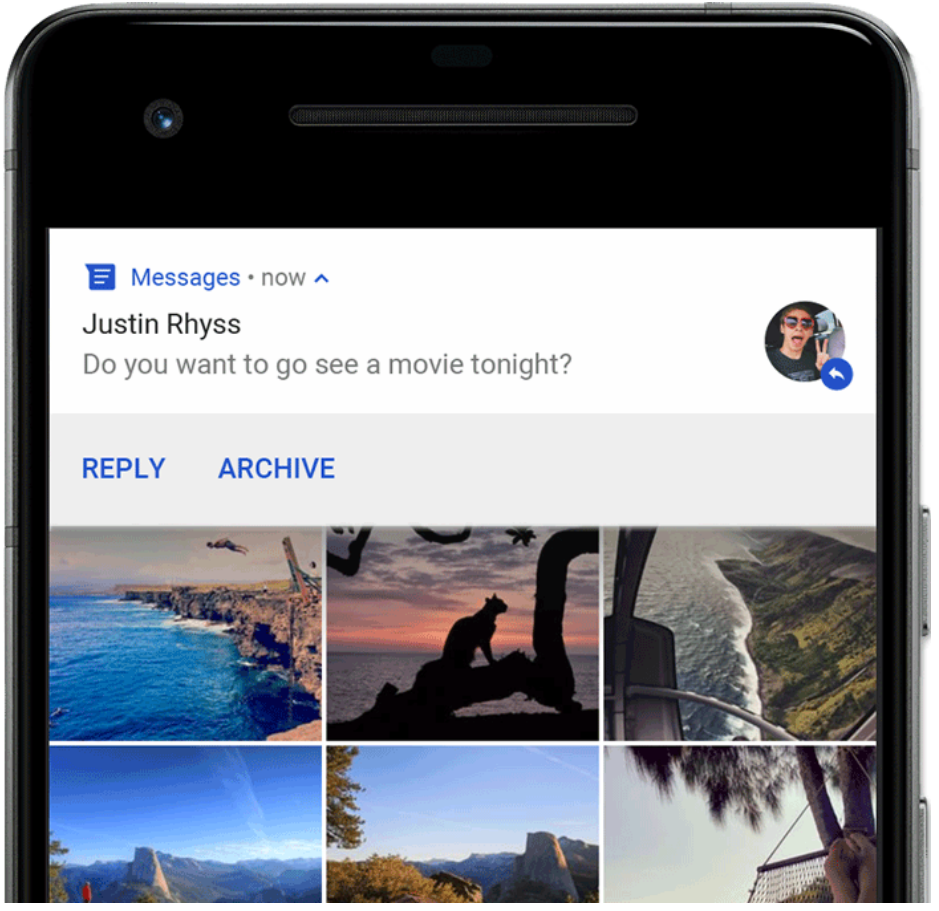
Users can swipe down on the status bar to open the notification drawer, where they can view more details and take actions with the notification.



Heads-up notification

Beginning with Android 5.0, notifications can briefly appear in a floating window called a heads-up notification. This behavior is normally for important notifications that the user should know about immediately, and it appears only if the device is unlocked.

Heads-up notification

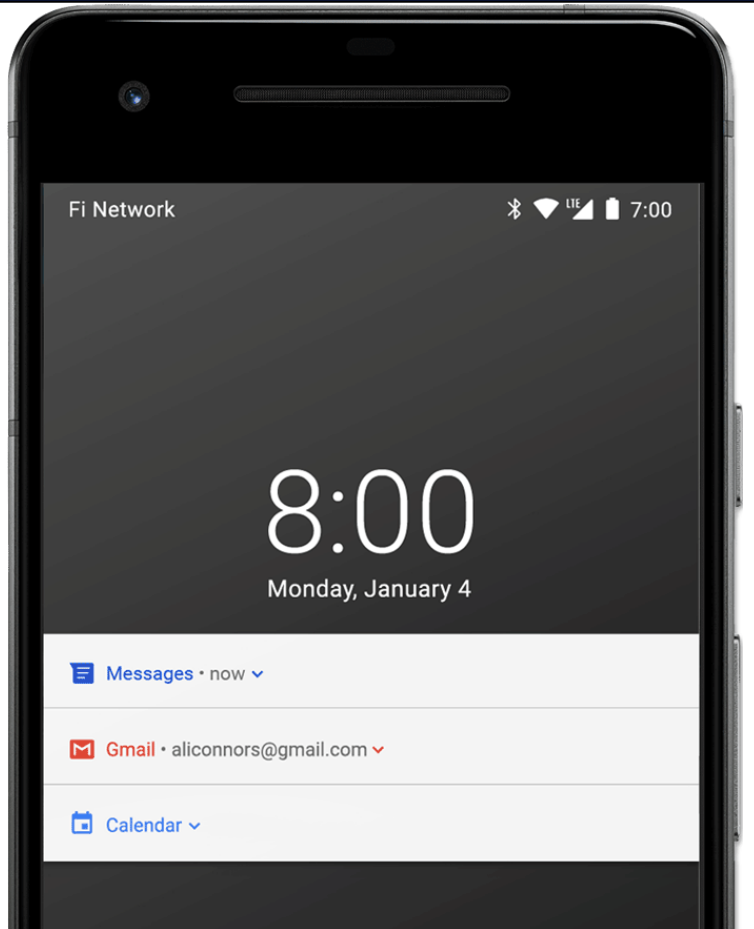


Lock screen

Beginning with Android 5.0, notifications can appear on the lock screen.

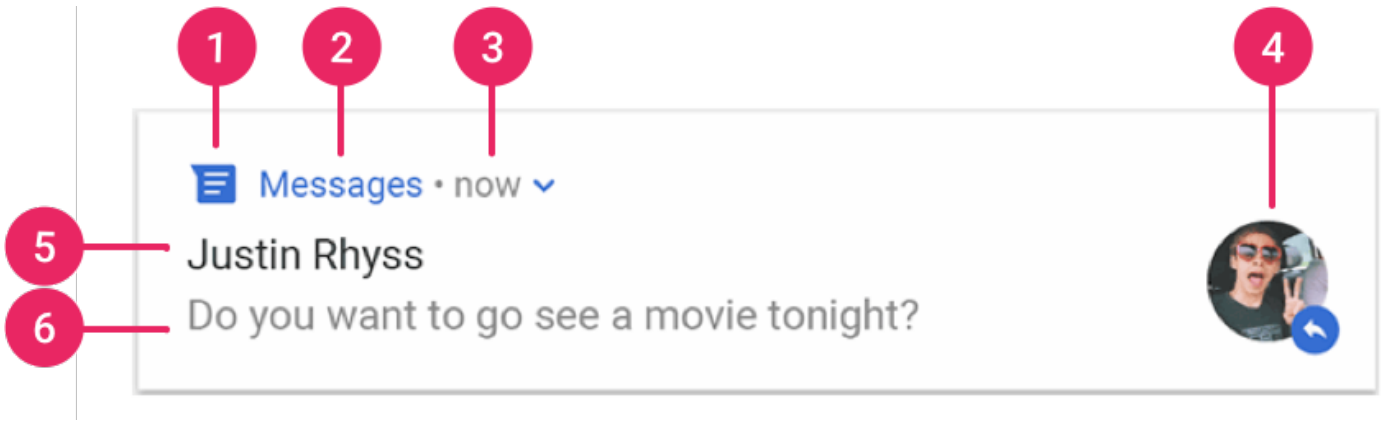
You can programmatically set the level of detail visible in notifications posted by your app on a secure lock screen, or even whether the notification will show on the lock screen at all.

Look Screen


Fi Network
7:00
8:00
Monday, January 4
Messages • now
Gmail • aliconnors@gmail.com
Calendar

Notification anatomy

The design of a notification is determined by system templates—your app simply defines the contents for each portion of the template. Some details of the notification appear only in the expanded view.

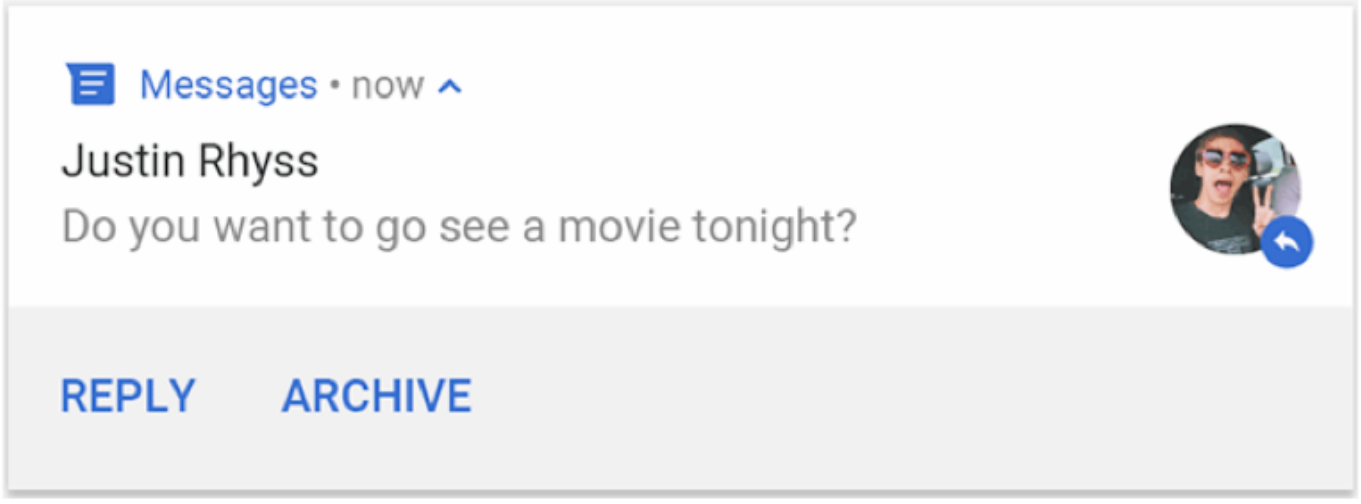


Notification anatomy

1. Small icon: This is required and set with **setSmallIcon()**.
2. App name: This is provided by the system.
3. Time stamp: This is provided by the system but you can override with **setWhen()** or hide it with **setShowWhen(false).**
4. Large icon: This is optional (usually used only for contact photos; do not use it for your app icon) and set with **setLargeIcon()**.
5. Title: This is optional and set with **setContentTitle()**.
6. Text: This is optional and set with **setContentText()**.

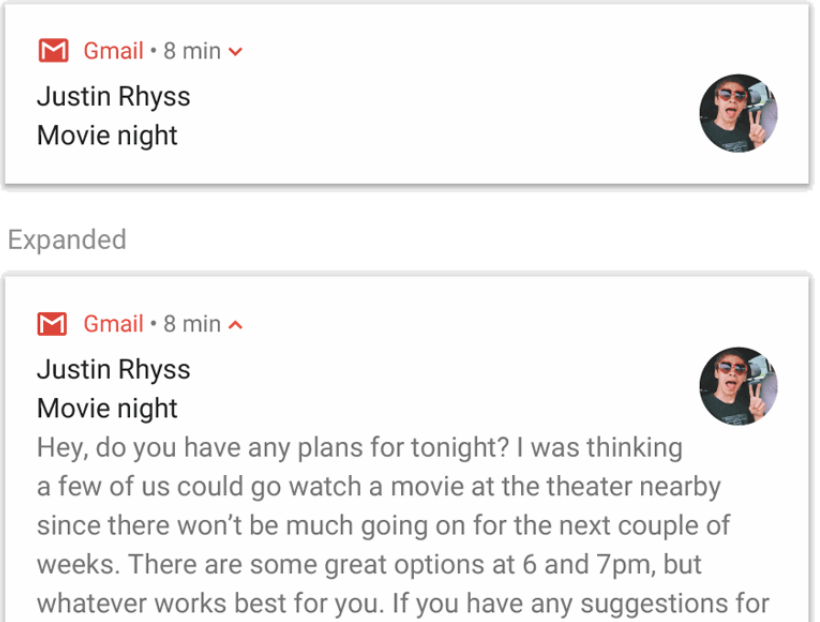
Notification actions

Although it's not required, every notification should open an appropriate app activity when tapped. In addition to this default notification action, you can add action buttons that complete an app-related task from the notification (often without opening an activity)



Expandable notification

- By default, the notification's text content is truncated to fit on one line. If you want your notification to be longer, you can enable a larger text area that's expandable by applying an additional template,

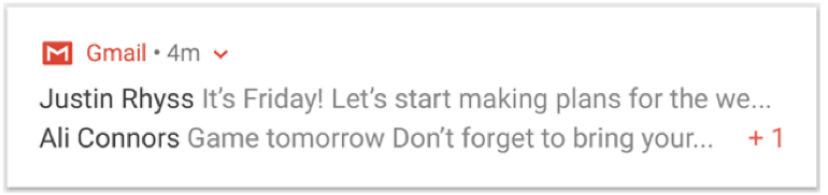


Notification updates and groups

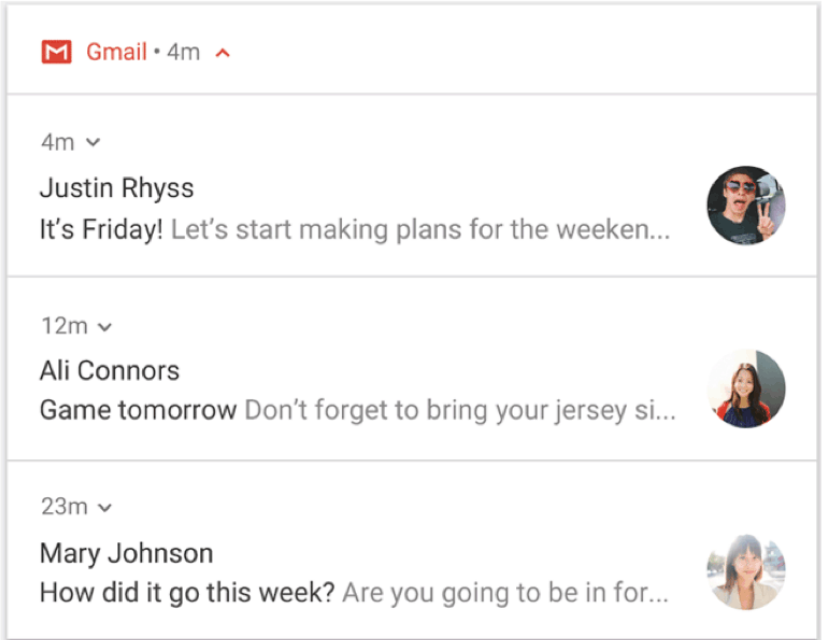
To avoid bombarding your users with multiple or redundant notifications when you have additional updates, you should consider updating an existing notification rather than issuing a new one, or consider using the inbox-style notification to show conversation updates.

However, if it's necessary to deliver multiple notifications, you should consider grouping those separate notifications into a group (available on Android 7.0 and higher). A notification group allows you to collapse multiple notifications into just one post in the notification drawer

Notification updates and groups



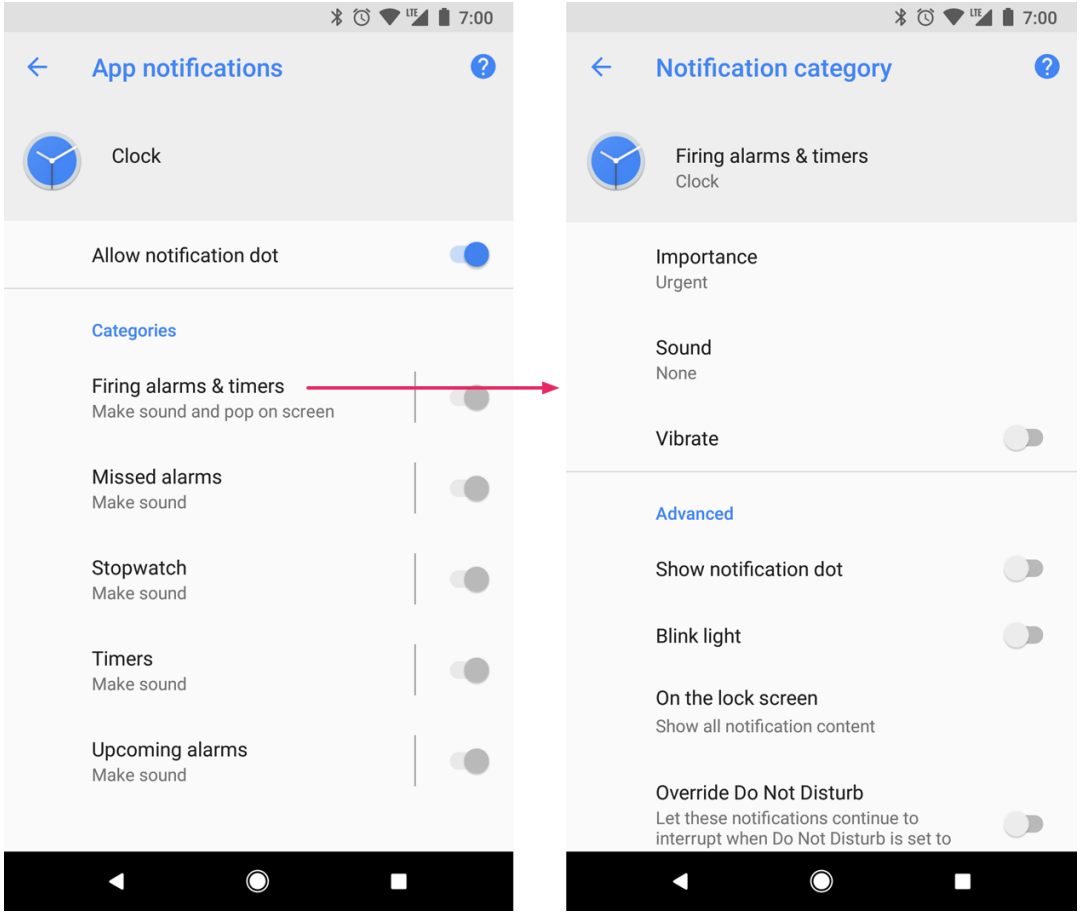
Expanded



Notification channels

Starting in Android 8.0 (API level 26), all notifications must be assigned to a channel or it will not appear. By categorizing notifications into channels, users can disable specific notification channels for your app (instead of disabling all your notifications), and users can control the visual and auditory options for each channel—all from the Android system settings

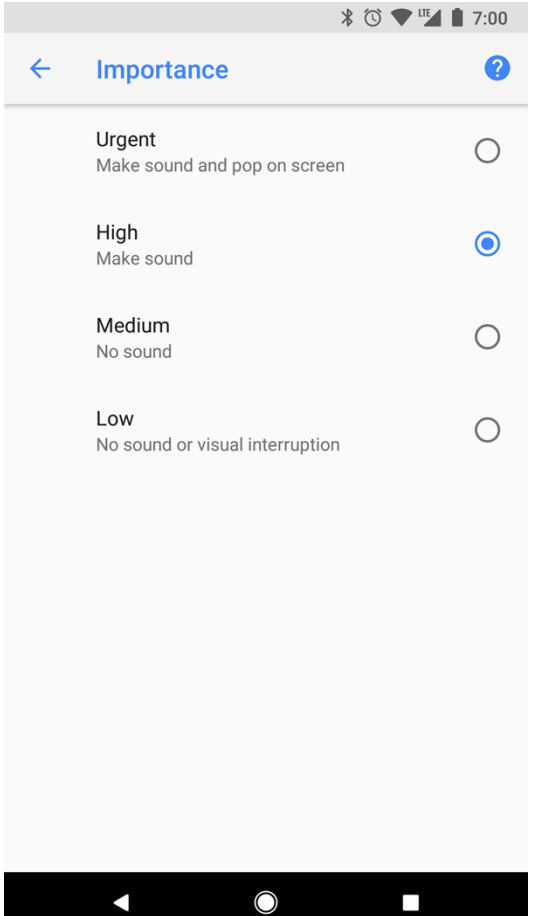
Notification channels



Notification importance

Android uses the importance of a notification to determine how much the notification should interrupt the user (visually and audibly). The higher the importance of a notification, the more interruptive the notification will be On Android 8.0 (API level 26) and above, importance of a notification is determined by the importance of the channel the notification was posted to. Users can change the importance of a notification channel in the system settings

Notification importance



Notifications for foreground services

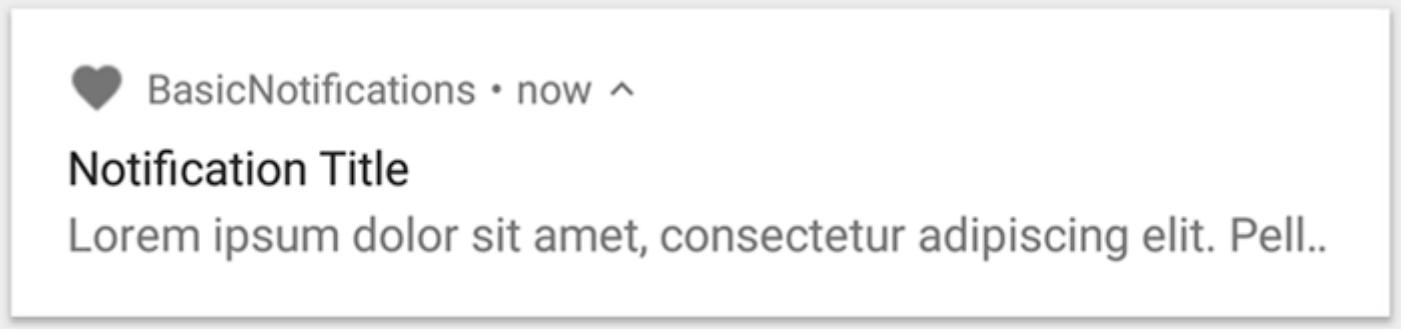
A notification is required when your app is running a "foreground service"—a Service running in the background that's long living and noticeable to the user, such as a media player. This notification cannot be dismissed like other notifications. To remove the notification, the service must be either stopped or removed from the "foreground" state.

Create a Notification

Notifications provide short, timely information about events in your app while it's not in use.

Create a basic notification

A notification in its most basic and compact form (also known as collapsed form) displays an icon, a title, and a small amount of content text. In this section, you'll learn how to create a notification that the user can click on to launch an activity in your app.



Set the notification content

To get started, you need to set the notification's content and channel using a NotificationCompat.Builder object. The following example shows how to create a notification with the following:

- A small icon, set by **setSmallIcon()**. This is the only user-visible content that's required.
- A title, set by **setContentTitle()**.
- The body text, set by **setContentText()**.
- The notification priority, set by **setPriority()**. The priority determines how intrusive the notification should be on Android 7.1 and lower.

Set the notification content

```
NotificationCompat.Builder builder = new
NotificationCompat.Builder(this, "CHANNEL_ID")
        .setSmallIcon(R.drawable.ic_launcher_background)
        .setContentTitle("Title")
        .setContentText("Message")
        .setPriority(NotificationCompat.PRIORITY_DEFAULT);
```

Notice that the NotificationCompat.Builder constructor requires that you provide a channel ID. This is required for compatibility with Android 8.0 (API level 26) and higher, but is ignored by older versions.

Expandable Notification

By default, the notification's text content is truncated to fit one line. If you want your notification to be longer, you can enable an expandable notification by adding a style template with setStyle(). For example, the following code creates a larger text area:

Expandable Notification

```
NotificationCompat.Builder builder = new
NotificationCompat.Builder(this, "CHANNEL_ID")
        .setSmallIcon(R.drawable.ic_launcher_background)
        .setContentTitle("Title")
        .setStyle(new NotificationCompat.BigTextStyle()
                .bigText("Notice that the
NotificationCompat.Builder constructor requires that you
provide a channel ID. This is required for compatibility
with Android 8.0 (API level 26) and higher, but is ignored
by older versions."))
        .setPriority(NotificationCompat.PRIORITY_DEFAULT);
```

Create a channel and set the importance

- Before you can deliver the notification on Android 8.0 and higher, you must register your app's notification channel with the system by passing an instance of NotificationChannel to createNotificationChannel() Because you must create the notification channel before posting any notifications on Android 8.0 and higher, you should execute this code as soon as your app starts. It's safe to call this repeatedly because creating an existing notification channel performs no operation.

Channel setup code

```
private void createNotificationChannel() {
    // Create the NotificationChannel, but only on API 26+ because
    // the NotificationChannel class is new and not in the support library
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        CharSequence name = getString(R.string.channel_name);
        String description = getString(R.string.channel_description);
        int importance = NotificationManager.IMPORTANCE_DEFAULT;
        NotificationChannel channel = new NotificationChannel(CHANNEL_ID, name,
importance);
        channel.setDescription(description);
        // Register the channel with the system; you can't change the importance
        // or other notification behaviors after this
        NotificationManager notificationManager =
getSystemService(NotificationManager.class);
        notificationManager.createNotificationChannel(channel);
    }
}
```

Set the notification's tap action

Every notification should respond to a tap, usually to open an activity in your app that corresponds to the notification. To do so, you must specify a content intent defined with a PendingIntent object and pass it to setContentIntent().

Set the notification's tap action

```
Intent intent = new Intent(this, Main2Activity.class);
intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK);
PendingIntent pendingIntent = PendingIntent.getActivity(this, 0,
intent, 0);

//set content to notification

.setContentIntent(pendingIntent)

.setAutoCancel(true)
```

Show the notification

To make the notification appear, call NotificationManagerCompat.notify(), passing it a unique ID for the notification and the result of NotificationCompat.Builder.build()

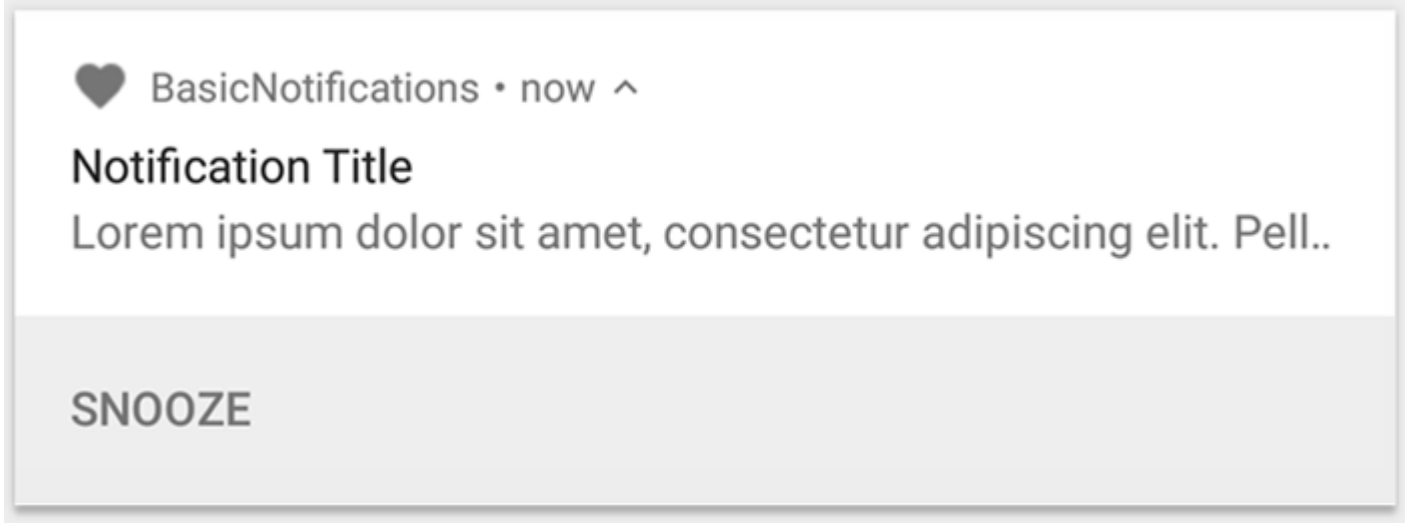
Remember to save the notification ID that you pass to NotificationManagerCompat.notify() because you'll need it later if you want to update or remove the notification.

Show Notification code

```
NotificationManagerCompat notificationManager =
        NotificationManagerCompat.from(this);
// notificationId is a unique int for each notification
                that you must define
        notificationManager.notify(notificationId,
                builder.build());
```

Add action buttons

A notification can offer up to three action buttons that allow the user to respond quickly, such as snooze a reminder or even reply to a text message. But these action buttons should not duplicate the action performed when the user taps the notification.



Add action buttons

To add an action button, pass a PendingIntent to the addAction() method. This is just like setting up the notification's default tap action, except instead of launching an activity, you can do a variety of other things such as start a BroadcastReceiver that performs a job in the background so the action does not interrupt the app that's already open.