Notifications

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.

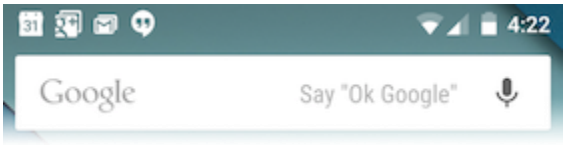

Figure 1. Notifications in the notification area.

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 training for more information. To learn how to design notifications and their interactions, read the Notifications design guide.

Creating a Notification

Notifications in Android o

The Android O Developer Preview introduces new features and capabilities for users and developers related to notifications, including notification channels. To learn about the new changes, see Android O for Developers.

A Notification object *must* contain the following:

- A small icon, set by setSmallIcon()
- A title, set by setContentTitle()
- Detail text, set by setContentText()

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.

Notification actions

Although they're optional, should add at least one action to your 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.

A notification can provide multiple actions. Always define the action that's triggered when the user clicks the notification; usually this action opens an Activity in your application. 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 use additional action buttons, you must also make their functionality available in an Activity in your app; see the section Handling compatibility for more details.

Inside a Notification, the action itself 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 want to start Activity when the user clicks the notification text in the notification drawer, add the PendingIntent by calling setContentIntent().

Starting an Activity when the user clicks the notification is the most common action scenario. You can also start an Activity when the user dismisses a notification. In Android 4.1 and later, you can start an Activity from an action button. To learn more, read the reference guide for NotificationCompat.Builder.

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() and pass in one of the NotificationCompat priority constants. There are five priority levels, ranging from PRIORITY_MIN (-2) to PRIORITY_MAX (2); if not set, the priority defaults to PRIORITY_DEFAULT (0).

For information about setting an appropriate priority level, see "Correctly set and manage notification priority" in the Notifications 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 object and uses it to create the PendingIntent for the action. This pattern is explained in more detail in the section Preserving Navigation when Starting an Activity:

```
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());
```