# Introducing Notifications

Notifi cations are a way for your applications to alert users, without using an Activity. Notifi cations are handled by the Notifi cation Manger, and currently include the ability to:

- ❑ Create a new status bar icon.
- ❑ Display additional information (and launch an Intent) in the extended status bar window.
- ❑ Flash the lights/LEDs.
- ❑ Vibrate the phone.
- ❑ Sound audible alerts (ringtones, media store sounds).

Notifications are the preferred way for invisible application components (Broadcast Receivers, Services,and inactive Activities) to alert users that events have occurred that require attention.

As a User Interface metaphor, Notifi cations are particularly well suited to mobile devices. It's likely that your users will have their phones with them at all times but quite unlikely that they will be paying attention to them, or your application, at any given time. Generally, users will have several applications open in the background, and they won't be paying attention to any of them.

In this environment, it's important that your applications be able to alert users when specifi c events occur that require their attention.

Notifi cations can be persisted through insistent repetition, or (more commonly) by using an icon on the status bar. Status bar icons can be updated regularly or expanded to show additional information using the expanded status bar window shown in Figure 8-3.


Figure 8-3

*To display the expanded status bar view, click a status bar icon and drag it toward the bottom of the screen. To "lock" it in place, ensure that you release your drag only after the window covers the entire screen. To hide it, simply drag it back upward.*

## *Introducing the Notifi cation Manager*

The *Notifi cation Manager* is a system Service used to handle Notifi cations. Get a reference to it using the getSystemService method, as shown in the snippet below:

```
String svcName = Context.NOTIFICATION_SERVICE;
NotificationManager notificationManager;
notificationManager = (NotificationManager)getSystemService(svcName);
```

Using the Notifi cation Manager, you can trigger new Notifi cations, modify existing ones, or remove those that are no longer needed or wanted.

# *Creating Notifi cations*

Creating and confi guring a new Notifi cation is done in three parts.

Firstly, you create a new Notifi cation object, passing in the icon to display in the status bar, along with the status bar ticker-text, and the time of this Notifi cation, as shown in the following code snippet:

```
// Choose a drawable to display as the status bar icon
int icon = R.drawable.icon;
// Text to display in the status bar when the notification is launched
String tickerText = “Notification”;
// The extended status bar orders notification in time order
long when = System.currentTimeMillis();
Notification notification = new Notification(icon, tickerText, when);
```

The ticker-text will scroll along the status bar when the Notifi cation is fired. Secondly, confi gure the appearance of the Notifi cation within the extended status window using the setLatestEventInfo method. This extended status window displays the icon and time defi ned in the constructor and also shows a title and a details string. Notifi cations often represent a request for action or attention, so you can specify a PendingIntent that will be fi red if a user clicks the Notifi cation item. The code snippet below uses setLatestEventInfo to set these values:

```
Context context = getApplicationContext();
// Text to display in the extended status window
String expandedText = “Extended status text”;
// Title for the expanded status
String expandedTitle = “Notification Title”;
// Intent to launch an activity when the extended text is clicked
Intent intent = new Intent(this, MyActivity.class);
PendingIntent launchIntent = PendingIntent.getActivity(context, 0, intent, 0);
notification.setLatestEventInfo(context, expandedTitle, expandedText, launchIntent);
```

It’s good form to use one Notifi cation icon to represent multiple instances of the same event (e.g., receiving multiple SMS messages). To demonstrate this to users, update the values set by setLatestEventInfo to reflect the most recent message and re-trigger the Notification to update its values.

You can also use the number property to display the number of events a status bar icon represents. Setting this value greater than 1, as shown below, overlays the values as a small number over the status bar icon:

```
notification.number++;
```

As with all changes to a Notification, you will need to re-trigger it to apply the change. To remove the overlay, set the number value to 0 or -1.

Finally, you can enhance Notifications using various properties on the Notification object to flash the device