Advanced Notification Techniques

In the following sections, you'll learn to enhance Notifi cations to provide additional alerting through hardware, in particular, by making the device ring, fl ash, and vibrate.

As each enhancement is described, you will be provided with a code snippet that can be added to the Earthquake example to provide user feedback on the severity of each earthquake as it's detected.

To use the Notifi cation techniques described here without also displaying the status bar icon, simply cancel the Notifi cation directly after triggering it. This stops the icon from displaying but doesn't interrupt the other effects.

Making Sounds

Using an audio alert to notify the user of a device event (like incoming calls) is a technique that predates the mobile, and has stood the test of time. Most native phone events from incoming calls to new messages and low battery are announced by an audible ringtone.

Android lets you play any audio fi le on the phone as a Notifi cation by assigning a location URI to the sound property, as shown in the snippet below:

```
notification.sound = ringURI;
```

To use your own custom audio, push the fi le onto your device, or include it as a raw resource, as described in Chapter 6.

The following snippet can be added to the announceNewQuake method within the Earthquake Service from the earlier example. It adds an audio component to the earthquake Notifi cation, ringing one of the default phone ringtones if a signifi cant earthquake (one with magnitude greater than 6) occurs.

```
if (quake.getMagnitude() > 6) {
Uri ringURI = Uri.fromFile(new File("/system/media/audio/ringtones/ringer.mp3"));
newEarthquakeNotification.sound = ringURI;
}
```

Vibrating the Phone

You can use the phone's vibration function to execute a vibration pattern specifi c to your Notifi cation. Android lets you control the pattern of a vibration; you can use vibration to convey information as well as get the user's attention.

To set a vibration pattern, assign an array of longs to the Notifi cation's vibrate property. Construct the array so that every alternate number is the length of time (in milliseconds) to vibrate or pause, respectively.

Before you can use vibration in your application, you need to be granted permission. Add a uses-permission to your application to request access to the device vibration using the following code snippet:

```
<uses-permission android:name="android.permission.VIBRATE"/>
```

The following example shows how to modify a Notifi cation to vibrate in a repeating pattern of 1 second on, 1 second off, for 5 seconds total.

```
long[] vibrate = new long[] { 1000, 1000, 1000, 1000, 1000 };
notification.vibrate = vibrate;
```

You can take advantage of this fi ne-grained control to pass information to your users. In the following update to the announceNewQuake method, the phone is set to vibrate in a pattern based on the power of the quake. Earthquakes are measured on an exponential scale, so you'll use the same scale when creating the vibration pattern.

For a barely perceptible magnitude 1 quake, the phone will vibrate for a fraction of a second; but for magnitude 10, an earthquake that would split the earth in two, your users will have a head start on the Apocalypse when their devices vibrate for a full 20 seconds. Most signifi cant quakes fall between 3 and 7 on the Richter scale, so the more likely scenario is a more reasonable 200-millisecond to 4-second vibration duration range.

```
double vibrateLength = 100*Math.exp(0.53*quake.getMagnitude());
long[] vibrate = new long[] {100, 100, (long)vibrateLength };
newEarthquakeNotification.vibrate = vibrate;
```

The current Android Emulator does not visually or audibly indicate that the device is vibrating. To confi rm that your Notifi cation is behaving appropriately, you can monitor the log for "Vibration On"/"Vibration Off."

Flashing the Lights

Notifi cations also include properties to confi gure the color and fl ash frequency of the device's LED.

The ledARGB property can be used to set the LED's color, while the ledOffMS and ledOnMS properties let you set the frequency and pattern of the fl ashing LED. You can turn the LED on by setting the ledOnMS property to 1 and the ledOffMS property to 0, or turn it off by setting both properties to 0.

Once you have confi gured the LED settings, you must also add the FLAG_SHOW_LIGHTS fl ag to the Notifi cation's flags property.

The following code snippet shows how to turn on the red device LED:

```
notification.ledARGB = Color.RED;
notification.ledOffMS = 0;
notification.ledOnMS = 1;
notification.flags = notification.flags | Notification.FLAG_SHOW_LIGHTS;
```

Controlling the color and fl ash frequency is another opportunity to pass additional information to users.

In the Earthquake monitoring example, you can help your users perceive the nuances of an exponential scale by also using the device's LED to help convey the magnitude. In the snippet below, the color of the LED depends on the size of the quake, and the frequency of the fl ashing is inversely related to the power of the quake:

```
int color;
if (quake.getMagnitude() < 5.4)
color = Color.GREEN;
else if (quake.getMagnitude() < 6)
color = Color.YELLOW;
else
color = Color.RED;
newEarthquakeNotification.ledARGB = color;
newEarthquakeNotification.ledOffMS = (int)vibrateLength;
newEarthquakeNotification.ledOnMS = (int)vibrateLength;
newEarthquakeNotification.flags = newEarthquakeNotification.flags |
Notification.FLAG_SHOW_LIGHTS;
```

The current Android Emulator does not visually illustrate the LEDs. This makes it quite diffi cult to confi rm that your LEDs are fl ashing correctly. In hardware, each device may have different limitations in regard to setting the color of the LED. In such cases, as close an approximation as possible will be used.

Ongoing and Insistent Notifications

Notifications can be configured as ongoing and/or insistent by setting the FLAG_INSISTENT and FLAG_ONGOING_EVENT flags.

Notifications fl agged as ongoing, as in the snippet below, are used to represent events that are currently in progress (such as an incoming call). Ongoing events are separated from "normal" Notifications within the extended status bar window.

```
notification.flags = notification.flags | Notification.FLAG_ONGOING_EVENT;
```

Insistent Notifications repeat continuously until canceled. The code snippet below shows how to set a Notification as insistent:

```
notification.flags = notification.flags | Notification.FLAG_INSISTENT;
```

Insistent Notifications are handled by continuously repeating the initial Notification effects until the Notification is canceled. Insistent Notifications should be reserved for situations like Alarms, where timely and immediate response is required.