Кроме [Toast-уведомлений](http://developer.alexanderklimov.ru/android/toast.php), существует также другой тип уведомлений, который выводится в системной строке состояния в виде значка с небольшим текстом. Если открыть окно уведомлений, то можно увидеть расширенную текстовую информацию об уведомлении.

Когда пользователь открывает расширенное сообщение, Android запускает объект **Intent**, который определён в соответствии с уведомлением. Можно также конфигурировать уведомление с добавлением звука, вибрации и мигающих индикаторов на мобильном устройстве.

Этот вид уведомления удобен в том случае, когда приложение работает в фоновом режиме и должно уведомить пользователя о каком-либо важном событии. Уведомление будет висеть до тех пор, пока пользователь не отреагирует на него, в отличие от Toast-сообщения, которое исчезнет через несколько секунд. Фоновое приложение создаёт уведомление в строке состояния, но не запускает активность самостоятельно для получения пользовательского взаимодействия. Это должен сделать только сам пользователь в удобное ему время.

Чтобы создать уведомление в строке состояния, необходимо использовать два класса:

* **Notification** — определяем свойства уведомления строки состояния: значок, расширенное сообщение и дополнительные параметры настройки (звук и др.)
* **NotificationManager** — системный сервис Android, который управляет всеми уведомлениями. Экземпляр **NotificationManager** создаётся при помощи вызова метода **from()**, а затем, когда надо показать уведомление пользователю, вызывается метод **notify()**

Добавим на экран активности кнопку и создадим простой пример для демонстрации работы уведомления.

Для начала нужно создать идентификатор уведомления. Он нужен, чтобы можно было различать уведомления друг от друга. Если будет один идентификатор, то каждое новое уведомление затрёт предыдущее.

Также следует создать идентификатор канала. Каналы появились в API 28, но старые устройства будут просто игнорировать данный параметр при вызове конструктора **NotificationCompat.Builder**.

public class MainActivity extends AppCompatActivity {

// Идентификатор уведомления

private static final int NOTIFY\_ID = 101;

// Идентификатор канала

private static String CHANNEL\_ID = "Cat channel";

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

Button button = findViewById(R.id.button);

button.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

NotificationCompat.Builder builder =

new NotificationCompat.Builder(MainActivity.this, CHANNEL\_ID)

.setSmallIcon(R.drawable.ic\_pets\_black\_24dp)

.setContentTitle("Напоминание")

.setContentText("Пора покормить кота")

.setPriority(NotificationCompat.PRIORITY\_DEFAULT);

NotificationManagerCompat notificationManager =

NotificationManagerCompat.from(MainActivity.this);

notificationManager.notify(NOTIFY\_ID, builder.build());

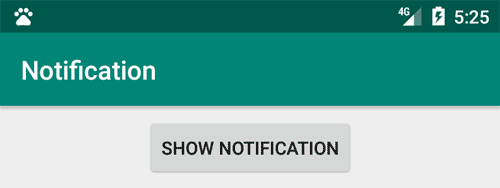
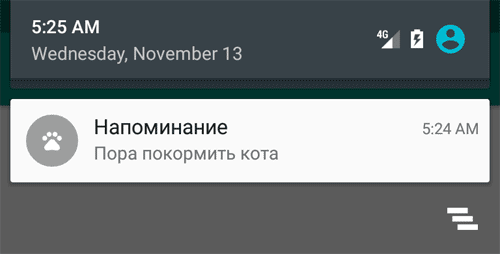
}

});

}

}

Запустим пример и нажмём кнопку. В строке состояния появится значок. Раскроем уведомление и увидим текст. Уведомление можно смахнуть в сторону для удаления.

Нажатие на уведомление ни к чему не приведёт. Нужен дополнительный код.

Создадим новые объекты **Intent** и **PendingIntent**, которые описывают намерения и целевые действия. В нашем случае мы хотим запустить нашу активность, когда пользователь среагирует на уведомление. Присоединяем объекты через **setContentIntent()**.

button.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

**Intent notificationIntent = new Intent(MainActivity.this, MainActivity.class);**

**PendingIntent contentIntent = PendingIntent.getActivity(MainActivity.this,**

**0, notificationIntent,**

**PendingIntent.FLAG\_CANCEL\_CURRENT);**

NotificationCompat.Builder builder =

new NotificationCompat.Builder(MainActivity.this, CHANNEL\_ID)

.setSmallIcon(R.drawable.ic\_pets\_black\_24dp)

.setContentTitle("Напоминание")

.setContentText("Пора покормить кота")

.setPriority(NotificationCompat.PRIORITY\_DEFAULT)

**.setContentIntent(contentIntent);**

NotificationManagerCompat notificationManager =

NotificationManagerCompat.from(MainActivity.this);

notificationManager.notify(NOTIFY\_ID, builder.build());

}

});

Теперь можно создать уведомление и затем закрыть приложение. Если нажать на уведомление, оно откроет заново ваше приложение.

Сделаем уведомление более красивым, добавив другие необязательные настройки.

NotificationCompat.Builder builder =

new NotificationCompat.Builder(MainActivity.this, CHANNEL\_ID)

.setSmallIcon(R.drawable.ic\_pets\_black\_24dp)

.setContentTitle("Напоминание")

.setContentText("Пора покормить кота")

.setPriority(NotificationCompat.PRIORITY\_DEFAULT)

.setContentIntent(contentIntent)

**// необязательные настройки**

**.setLargeIcon(BitmapFactory.decodeResource(getResources(),**

**R.drawable.hungrycat)) // большая картинка**

**.setTicker("Последнее китайское предупреждение!") // до Lollipop**

**.setAutoCancel(true); // автоматически закрыть уведомление после нажатия**

Теперь в уведомлении мы видим картинку. Метод **setTicker()** выводит сообщение в строке состояния на короткое время, а затем исчезает. Это работает только на старых устройствах и сейчас можно уже не использовать.

Как я уже упоминалось, если вам нужно обновить уведомление, то просто ещё раз отправьте его устройству под этим же идентификатором, но с другим текстом и картинкой.

Если уведомления разного типа, то нужно обновлять идентификаторы.

// Объявим переменную в начале класса

private int counter = 101;

// Теперь у уведомлений будут новые идентификаторы

notificationManager.notify(counter++, notification);

**Удаление собственных уведомлений**

Вы можете из программы удалить своё уведомление, посланное по глупости.

// Удаляем конкретное уведомление

notificationManager.cancel(NOTIFY\_ID);

// Удаляем все свои уведомления

notificationManager.cancelAll();

Если уведомления с указанным идентификатором не будет, то ничего страшного при удалении не произойдёт, поэтому проверку не нужно устраивать.

**Использование настроек по умолчанию**

Можно добавить вибрацию, звуковой сигнал или мерцание светодиодами для ваших уведомлений при помощи настроек по умолчанию. В свойстве **defaults** вы можете сочетать следующие константы:

* Notification.DEFAULT\_LIGHTS
* Notification.DEFAULT\_SOUND
* Notification.DEFAULT\_VIBRATE

Чтобы к уведомлению добавить звук и вибрации по умолчанию, используйте код:

notification.defaults = Notification.DEFAULT\_SOUND |

Notification.DEFAULT\_VIBRATE;

Если хотите установить сразу все значения по умолчанию, задействуйте константу **Notification.DEFAULT\_ALL**.

**Звуковое сопровождение**

Использование звуковых оповещений для уведомления пользователя о событиях, связанных с устройством (например, входящий звонок), стало привычным. Большинство стандартных событий, от входящих звонков до новых сообщений и низкого заряда батареи, объявляются с помощью звуковых мелодий. Android позволяет проигрывать любой звуковой файл на телефоне в качестве уведомления. Чтобы это сделать, нужно присвоить свойству **sound** путь **URI**:

notification.sound = ringURI;

Также можно использовать собственный звуковой файл, загруженный на устройстве или добавленный в проект в качестве ресурса.

Uri ringURI =

RingtoneManager.getDefaultUri(RingtoneManager.TYPE\_NOTIFICATION);

notification.sound = ringURI;

С SD-карты:

notification.sound = Uri.parse("file:///sdcard/cat.mp3"); // если знаем точный путь!

**Виброзвонок**

Чтобы использовать виброзвонок, передайте в свойство **vibrate** объекта **Notification** массив значений типа **long**. Постройте массив, учитывая, что значения, отвечающие за продолжительность вибрации (в миллисекундах), чередуются со значениями, которые означают длину паузы между вибрациями.

Прежде чем использовать виброзвонок в своем приложении, необходимо получить нужные полномочия, прописав их в манифесте:

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

В следующем примере показано, как изменить уведомление, чтобы одна секунда вибрации сменялась одной секундой паузы на протяжении пяти секунд:

long[] vibrate = new long[] { 1000, 1000, 1000, 1000, 1000 };

notification.vibrate = vibrate;

В настоящее время эмулятор Android не умеет оповещать о вибрации ни визуально, ни с помощью звуковых сигналов.

**Светодиодная индикация**

Объект **Notification** включает в себя свойства для настройки цвета и частоты мерцания светодиодов устройства. Здесь стоит обратить внимание, что конкретные модели устройств могут не содержать светодиодные индикаторы или иметь другие цвета.

Свойство **ledARGB** может устанавливать цвет для светодиодной подсветки. Свойства **ledOffMS** и **ledOnMS** позволяют регулировать частоту и поведение светодиодов. Вы можете включить светодиоды, присвоив свойству **ledOnMS** значение 1, а **ledOffMS** – 0. Присвоив им обоим значения 0, светодиоды можно выключить.

Настроив работу со светодиодами, необходимо также добавить флаг **FLAG\_SHOW\_LIGHTS** к свойству **flags** объекта **Notification**.

В следующем фрагменте кода показано, как включить на устройстве красный светодиод:

notification.ledARGB = Color.RED;

notification.ledOffMS = 0;

notification.ledOnMS = 1;

notification.flags = notification.flags | Notification.FLAG\_SHOW\_LIGHTS;

В настоящее время эмулятор Android не умеет визуально показывать активность светодиодов.

**Текущие и настойчивые уведомления**

Вы можете делать уведомления текущими и/или настойчивыми, устанавливая флаги **FLAG\_INSISTENT** и **FLAG\_ONGOING\_EVENT**. Уведомления, помеченные как текущие, используются для представления событий, которые выполняются в данный момент времени (например, загрузка файла, фоновое проигрывание музыки). Текущие уведомления необходимы для сервисов, работающих на переднем плане. Пример установки флагов:

notification.flags = notification.flags | Notification.FLAG\_ONGOING\_EVENT;

В расширенной статусной строке текущие события отделены от обычных, чтобы вы сразу могли их отличить.

Настойчивые уведомления непрерывно повторяют звуковые сигналы, вибрируют и мерцают светодиодами, пока не будут остановлены. Подобные уведомления, как правило, используются для событий, которые требуют немедленного и своевременного внимания, таких как входящий звонок, срабатывание будильника или время кормёжки кота. В следующем фрагменте кода показано, как сделать уведомление настойчивым:

notification.flags = notification.flags | Notification.FLAG\_INSISTENT;

**Меняем цвет значка**

По умолчанию, значок выводится в сером круге. Вы можете изменить цвет круга, вызвав новый метод **setColor()**, который появился в API 21:

NotificationCompat.Builder builder =

new NotificationCompat.Builder(MainActivity.this, CHANNEL\_ID)

...

**.setColor(Color.GREEN)**

.build();

**Анимированный значок для уведомления**

Возьмём код из примера и заменим одну строчку, которая отвечает за вывод маленького значка - **.setSmallIcon(android.R.drawable.stat\_sys\_upload)**:

NotificationCompat.Builder builder =

new NotificationCompat.Builder(MainActivity.this, CHANNEL\_ID)

.setSmallIcon(android.R.drawable.stat\_sys\_upload)

... // другой код

.setAutoCancel(true); // автоматически закрыть уведомление после нажатия

Запускаем код и создаём уведомление. Мы увидите, что в строке состояния выводится анимированный значок стрелки.

**Уведомление с тремя кнопками**

Это может быть удобным, если приложение состоит из нескольких активностей или нужно предложить три разных варианта развития сценария.

За появление кнопок в уведомлении отвечает метод **setAction()**.

Intent notificationIntent = new Intent(MainActivity.this, SecondActivity.class);

PendingIntent pendingIntent = PendingIntent.getActivity(MainActivity.this,

0, notificationIntent,

PendingIntent.FLAG\_CANCEL\_CURRENT);

NotificationCompat.Builder builder =

new NotificationCompat.Builder(MainActivity.this, CHANNEL\_ID)

.setSmallIcon(R.drawable.ic\_pets\_black\_24dp)

.setContentTitle("Посылка")

.setContentText("Это я, почтальон Печкин. Принес для вас посылку")

.setPriority(NotificationCompat.PRIORITY\_DEFAULT)

.setContentIntent(pendingIntent)

.setLargeIcon(BitmapFactory.decodeResource(getResources(),

R.drawable.hungrycat)) // большая картинка

**.addAction(R.drawable.ic\_lock\_open\_black\_24dp, "Открыть", pendingIntent)**

**.addAction(R.drawable.ic\_refresh\_white\_24dp, "Отказаться", pendingIntent)**

**.addAction(R.drawable.ic\_pets\_black\_24dp, "Другой вариант", pendingIntent)**

.setAutoCancel(true); // автоматически закрыть уведомление после нажатия

NotificationManagerCompat notificationManager =

NotificationManagerCompat.from(MainActivity.this);

notificationManager.notify(NOTIFY\_ID, builder.build());

Обратите внимание, что у кнопок текст может обрезаться и пользователь не увидит текст, поэтому следует придумать "говорящие" значки, по которым будет понятен смысл нажатия.

**Уведомление с длинным текстом. BigTextStyle().bigText()**

В предыдущих приерах, длинный текст, помещённый в метод **setContentText()**, вывелся на экран не полностью. Если информация слишком важная и вам хочется её показать в уведомлении полностью, то подойдёт вариант со стилем **BigTextStyle**:

Intent notificationIntent = new Intent(MainActivity.this, SecondActivity.class);

PendingIntent pendingIntent = PendingIntent.getActivity(MainActivity.this,

0, notificationIntent,

PendingIntent.FLAG\_CANCEL\_CURRENT);

String bigText = "Это я, почтальон Печкин. Принёс для вас посылку. "

+ "Только я вам её не отдам. Потому что у вас документов нету. ";

NotificationCompat.Builder builder =

new NotificationCompat.Builder(MainActivity.this, CHANNEL\_ID)

.setSmallIcon(R.drawable.ic\_pets\_black\_24dp)

.setContentTitle("Посылка")

.setContentText("Это я, почтальон Печкин. Принес для вас посылку")

.setPriority(NotificationCompat.PRIORITY\_DEFAULT)

.setContentIntent(pendingIntent)

.setLargeIcon(BitmapFactory.decodeResource(getResources(),

R.drawable.hungrycat)) // большая картинка

.addAction(R.drawable.ic\_pets\_black\_24dp, "Запустить активность",

pendingIntent)

.setStyle(new NotificationCompat.BigTextStyle().bigText(bigText))

.setAutoCancel(true); // автоматически закрыть уведомление после нажатия

NotificationManagerCompat notificationManager =

NotificationManagerCompat.from(MainActivity.this);

notificationManager.notify(NOTIFY\_ID, builder.build());

**Уведомление с большой картинкой: BigPictureStyle().bigPicture()**

Пример с большой картинкой аналогичен с предыдущим примером. Только мы задаём уже другой стиль для уведомления. Вместо стиля длинного текста используется стиль **BigPictureStyle().bigPicture()**:

Intent notificationIntent = new Intent(MainActivity.this, MainActivity.class);

PendingIntent pendingIntent = PendingIntent.getActivity(MainActivity.this,

0, notificationIntent,

PendingIntent.FLAG\_CANCEL\_CURRENT);

NotificationCompat.Builder builder =

new NotificationCompat.Builder(MainActivity.this, CHANNEL\_ID)

.setSmallIcon(R.drawable.ic\_pets\_black\_24dp)

.setContentTitle("Посылка")

.setContentText("Это я, почтальон Печкин. Принёс для вас посылку")

.setPriority(NotificationCompat.PRIORITY\_DEFAULT)

.setContentIntent(pendingIntent)

.setLargeIcon(BitmapFactory.decodeResource(getResources(),

R.drawable.hungrycat)) // большая картинка

.addAction(R.drawable.ic\_pets\_black\_24dp, "Запустить активность",

pendingIntent)

// большая картинка из ресурсов

.setStyle(new NotificationCompat.BigPictureStyle()

.bigPicture(BitmapFactory.decodeResource(getResources(),

R.drawable.hungrycat)))

.setAutoCancel(true); // автоматически закрыть уведомление после нажатия

NotificationManagerCompat notificationManager =

NotificationManagerCompat.from(MainActivity.this);

notificationManager.notify(NOTIFY\_ID, builder.build());

Слишком большая картинка будет обрезана.

**Уведомление в стиле InboxStyle**

Есть ещё один стиль **InboxStyle**, напоминающий стиль писем в папке Входящие. Стиль разместит до пяти ваших строк в виде списка. Меняется только вызов **setStyle()**

...

.setStyle(new NotificationCompat.InboxStyle()

.addLine("Первое сообщение").addLine("Второе сообщение")

.addLine("Третье сообщение").addLine("Четвертое сообщение")

.setSummaryText("+2 more"))

**Уведомление в стиле мессенджера: MessagingStyle**

Стиль **MessagingStyle** пригодится для отображения сообщений из мессенджера или чата. Появился в Android Nougat.

Intent notificationIntent = new Intent(MainActivity.this, SecondActivity.class);

PendingIntent pendingIntent = PendingIntent.getActivity(MainActivity.this,

0, notificationIntent,

PendingIntent.FLAG\_CANCEL\_CURRENT);

Person murzik = new Person.Builder().setName("Мурзик").build();

Person vaska = new Person.Builder().setName("Васька").build();

NotificationCompat.MessagingStyle messagingStyle = new NotificationCompat.MessagingStyle

(murzik)

.setConversationTitle("Android chat")

.addMessage("Привет котаны!", System.currentTimeMillis(), murzik)

.addMessage("А вы знали, что chat по-французски кошка?", System

.currentTimeMillis(),

murzik)

.addMessage("Круто!", System.currentTimeMillis(),

vaska)

.addMessage("Ми-ми-ми", System.currentTimeMillis(), vaska)

.addMessage("Мурзик, откуда ты знаешь французский?", System.currentTimeMillis(),

vaska)

.addMessage("Шерше ля фам, т.е. ищите кошечку!", System.currentTimeMillis(),

murzik);

NotificationCompat.Builder builder =

new NotificationCompat.Builder(MainActivity.this, CHANNEL\_ID)

.setSmallIcon(R.drawable.ic\_pets\_black\_24dp)

.setContentIntent(pendingIntent)

.addAction(R.drawable.ic\_pets\_black\_24dp, "Запустить активность",

pendingIntent)

.setStyle(messagingStyle)

.setAutoCancel(true); // автоматически закрыть уведомление после нажатия

NotificationManagerCompat notificationManager =

NotificationManagerCompat.from(MainActivity.this);

notificationManager.notify(NOTIFY\_ID, builder.build());

В конструкторе **MessagingStyle** вы должны указать имя текущего пользователя, который будет видеть свои сообщения.

У класса **Person** есть другие полезные методы: **setIcon()** (значок), **setData()** (картинки) и др.

Разговор строится через цепочку вызовов методов **addMessage()**, в которых указывается текст сообщения, время, отправитель. Количество сообщений может быть любым. При большом количестве (задано в **MessagingStyle.MAXIMUM\_RETAINED\_MESSAGES**) старые сообщения начнут удаляться автоматически.

Подводя итоги, следует отметить, у уведомлений очень много методов, которые можно использовать в своём приложении. Вот как может выглядеть полный набор:

new Notification.Builder(this.getApplicationContext())

.setAutoCancel(boolean autoCancel)

.setContent(RemoteViews views)

.setContentInfo(CharSequence info)

.setContentIntent(PendingIntent intent)

.setContentText(CharSequence text)

.setContentTitle(CharSequence title)

.setDefaults(int defaults)

.setDeleteIntent(PendingIntent intent))

.setFullScreenIntent(PendingIntent intent, boolean highPriority)

.setLargeIcon(Bitmap icon)

.setLights(int argb, int onMs, int offMs)

.setNumber(int number)

.setOngoing(boolean ongoing)

.setOnlyAlertOnce(boolean onlyAlertOnce)

.setPriority(int pri)

.setProgress(int max, int progress, boolean indeterminate)

.setShowWhen(boolean show)

.setSmallIcon(int icon, int level)

.setSmallIcon(int icon)

.setSound(Uri sound)

.setSound(Uri sound, int streamType)

.setStyle(Notification.Style style)

.setSubText(CharSequence text)

.setTicker(CharSequence tickerText, RemoteViews views)

.setTicker(CharSequence tickerText)

.setUsesChronometer(boolean b)

.setVibrate(long[] pattern)

.setWhen(long when)

.addAction(int icon, CharSequence title, PendingIntent intent)

.build()

* **setSmallIcon()** устанавливает маленький значок, который выводится в строке состояния, а также в правой части открытого уведомления.
* **setLargeIcon()** устанавливает большой значок, который выводится в открытом уведомлении слева.
* **setWhen()** определяет время для уведомления, по умолчанию время создания уведомления
* **setTicker()** выводит временную строку в строке состояния, которая затем исчезает. Остаётся только маленький значок (см. выше)
* **setNumber()** добавляет число справа от уведомления (не везде работает)
* **setShowWhen()** - показывать ли время в уведомлении (в Android 7.0 по умолчанию не показывается)
* **setUsesChronometer()** выводит счётчик вместо времени, показывающий сколько прошло от времени when. Полезно для уведомления секундомера или звонка
* **setContentInfo()** добавляет текст справа от уведомления (в новых версиях сверху)
* **setColor()** закрашивает значок и название приложения указанным цветом
* **setOngoing()** выводит уведомление поверх обычных уведомлений, такое уведомление нельзя закрыть или смахнуть.
* **setVibrate()** - виброзвонок
* **setSound()** - звук
* **setLights()** - цвет LED-индикатора
* **setPriority()** устанавливает приоритет от -2 (**NotificationCompat.PRIORITY\_MIN**) до 2 (**NotificationCompat.PRIORITY\_MAX**)
* **setTimeoutAfter()** (появилось в API 26) - устанавливает таймаут, после которого уведомление удалится
* **setProgress()** - индикатор прогресса