# 9. Service, BroadcastReciever

## Broadcasts overview

Android apps can send or receive broadcast messages from the Android system and other Android apps, similar to the [publish-subscribe](https://en.wikipedia.org/wiki/Publish%E2%80%93subscribe_pattern) design pattern. These broadcasts are sent when an event of interest occurs. For example, the Android system sends broadcasts when various system events occur, such as when the system boots up or the device starts charging. Apps can also send custom broadcasts, for example, to notify other apps of something that they might be interested in (for example, some new data has been downloaded).

Apps can register to receive specific broadcasts. When a broadcast is sent, the system automatically routes broadcasts to apps that have subscribed to receive that particular type of broadcast.

Generally speaking, broadcasts can be used as a messaging system across apps and outside of the normal user flow. However, you must be careful not to abuse the opportunity to respond to broadcasts and run jobs in the background that can contribute to a slow system performance, as described in the following video.

## About system broadcasts

The system automatically sends broadcasts when various system events occur, such as when the system switches in and out of airplane mode. System broadcasts are sent to all apps that are subscribed to receive the event.

The broadcast message itself is wrapped in an [Intent](https://developer.android.com/reference/android/content/Intent.html) object whose action string identifies the event that occurred (for example android.intent.action.AIRPLANE\_MODE). The intent may also include additional information bundled into its extra field. For example, the airplane mode intent includes a boolean extra that indicates whether or not Airplane Mode is on.

For more information about how to read intents and get the action string from an intent, see [Intents and Intent Filters](https://developer.android.com/guide/components/intents-filters.html).

For a complete list of system broadcast actions, see the BROADCAST\_ACTIONS.TXT file in the Android SDK. Each broadcast action has a constant field associated with it. For example, the value of the constant[ACTION\_AIRPLANE\_MODE\_CHANGED](https://developer.android.com/reference/android/content/Intent.html#ACTION_AIRPLANE_MODE_CHANGED) is android.intent.action.AIRPLANE\_MODE. Documentation for each broadcast action is available in its associated constant field.

## Receiving broadcasts

Apps can receive broadcasts in two ways: through manifest-declared receivers and context-registered receivers.

## Manifest-declared receivers

If you declare a broadcast receiver in your manifest, the system launches your app (if the app is not already running) when the broadcast is sent.

To declare a broadcast receiver in the manifest, perform the following steps:

Specify the [<receiver>](https://developer.android.com/guide/topics/manifest/receiver-element.html) element in your app's manifest.

<receiver android:name=".MyBroadcastReceiver" android:exported="true">

<intent-filter>

<action android:name="android.intent.action.BOOT\_COMPLETED"/>

<action android:name="android.intent.action.INPUT\_METHOD\_CHANGED" />

</intent-filter>

</receiver>

The intent filters specify the broadcast actions your receiver subscribes to.

Subclass [BroadcastReceiver](https://developer.android.com/reference/android/content/BroadcastReceiver.html) and implement [onReceive(Context, Intent)](https://developer.android.com/reference/android/content/BroadcastReceiver.html#onReceive(android.content.Context,%20android.content.Intent)). The broadcast receiver in the following example logs and displays the contents of the broadcast:

public class MyBroadcastReceiver extends BroadcastReceiver {

private static final String TAG = "MyBroadcastReceiver";

@Override

public void onReceive(Context context, Intent intent) {

StringBuilder sb = new StringBuilder();

sb.append("Action: " + intent.getAction() + "\n");

sb.append("URI: " + intent.toUri(Intent.URI\_INTENT\_SCHEME).toString() + "\n");

String log = sb.toString();

Log.d(TAG, log);

Toast.makeText(context, log, Toast.LENGTH\_LONG).show();

}

}

The system package manager registers the receiver when the app is installed. The receiver then becomes a separate entry point into your app which means that the system can start the app and deliver the broadcast if the app is not currently running.

The system creates a new [BroadcastReceiver](https://developer.android.com/reference/android/content/BroadcastReceiver.html) component object to handle each broadcast that it receives. This object is valid only for the duration of the call to [onReceive(Context, Intent)](https://developer.android.com/reference/android/content/BroadcastReceiver.html#onReceive(android.content.Context,%20android.content.Intent)). Once your code returns from this method, the system considers the component no longer active.

### Context-registered receivers

To register a receiver with a context, perform the following steps:

Create an instance of [BroadcastReceiver](https://developer.android.com/reference/android/content/BroadcastReceiver.html).

BroadcastReceiver br = new MyBroadcastReceiver();

Create an [IntentFilter](https://developer.android.com/reference/android/content/IntentFilter.html) and register the receiver by calling [registerReceiver(BroadcastReceiver, IntentFilter)](https://developer.android.com/reference/android/content/Context.html#registerReceiver(android.content.BroadcastReceiver,%20android.content.IntentFilter)):

IntentFilter filter = new IntentFilter(ConnectivityManager.CONNECTIVITY\_ACTION);

filter.addAction(Intent.ACTION\_AIRPLANE\_MODE\_CHANGED);

this.registerReceiver(br, filter);

**Note:** To register for local broadcasts, call [**LocalBroadcastManager.registerReceiver(BroadcastReceiver, IntentFilter)**](https://developer.android.com/reference/android/support/v4/content/LocalBroadcastManager.html#registerReceiver(android.content.BroadcastReceiver,%20android.content.IntentFilter)) instead.

To stop receiving broadcasts, call [unregisterReceiver(android.content.BroadcastReceiver)](https://developer.android.com/reference/android/content/Context.html#unregisterReceiver(android.content.BroadcastReceiver)). Be sure to unregister the receiver when you no longer need it or the context is no longer valid.

## Sending broadcasts

Android provides three ways for apps to send broadcast:

* The [sendOrderedBroadcast(Intent, String)](https://developer.android.com/reference/android/content/Context.html#sendOrderedBroadcast(android.content.Intent,%20java.lang.String)) method sends broadcasts to one receiver at a time. As each receiver executes in turn, it can propagate a result to the next receiver, or it can completely abort the broadcast so that it won't be passed to other receivers. The order receivers run in can be controlled with the android:priority attribute of the matching intent-filter; receivers with the same priority will be run in an arbitrary order.
* The [sendBroadcast(Intent)](https://developer.android.com/reference/android/content/Context.html#sendBroadcast(android.content.Intent)) method sends broadcasts to all receivers in an undefined order. This is called a Normal Broadcast. This is more efficient, but means that receivers cannot read results from other receivers, propagate data received from the broadcast, or abort the broadcast.
* The [LocalBroadcastManager.sendBroadcast](https://developer.android.com/reference/android/support/v4/content/LocalBroadcastManager.html#sendBroadcast(android.content.Intent)) method sends broadcasts to receivers that are in the same app as the sender. If you don't need to send broadcasts across apps, use local broadcasts. The implementation is much more efficient (no interprocess communication needed) and you don't need to worry about any security issues related to other apps being able to receive or send your broadcasts.

The following code snippet demonstrates how to send a broadcast by creating an Intent and calling [sendBroadcast(Intent)](https://developer.android.com/reference/android/content/Context.html#sendBroadcast(android.content.Intent)).

Intent intent = new Intent();

intent.setAction("com.example.broadcast.MY\_NOTIFICATION");

intent.putExtra("data","Notice me senpai!");

sendBroadcast(intent);

## Сервис

[Service](https://developer.android.com/reference/android/app/Service.html) является компонентом приложения, который может выполнять длительные операции в фоновом режиме и не содержит пользовательского интерфейса. Другой компонент приложения может запустить службу, которая продолжит работу в фоновом режиме даже в том случае, когда пользователь перейдет в другое приложение. Кроме того, компонент может привязаться к службе для взаимодействия с ней и даже выполнять межпроцессное взаимодействие (IPC). Например, служба может обрабатывать сетевые транзакции, воспроизводить музыку, выполнять ввод-вывод файла или взаимодействовать с поставщиком контента, и все это в фоновом режиме.

Фактически служба может принимать две формы:

**Запущенная**

Служба является «запущенной», когда компонент приложения (например, операция) запускает ее вызовом [startService()](https://developer.android.com/reference/android/content/Context.html#startService(android.content.Intent)). После запуска служба может работать в фоновом режиме в течение неограниченного времени, даже если уничтожен компонент, который ее запустил. Обычно запущенная служба выполняет одну операцию и не возвращает результатов вызывающему компоненту. Например, она может загружать или выгружать файл по сети. Когда операция выполнена, служба должна остановиться самостоятельно.

**Привязанная**

Служба является «привязанной», когда компонент приложения привязывается к ней вызовом [bindService()](https://developer.android.com/reference/android/content/Context.html#bindService(android.content.Intent,%20android.content.ServiceConnection,%20int)). Привязанная служба предлагает интерфейс клиент-сервер, который позволяет компонентам взаимодействовать со службой, отправлять запросы, получать результаты и даже делать это между разными процессами посредством межпроцессного взаимодействия (IPC). Привязанная служба работает только пока к ней привязан другой компонент приложения. К службе могут быть привязаны несколько компонентов одновременно, но когда все они отменяют привязку, служба уничтожается.

Хотя в этой документации эти два типа служб обсуждаются отдельно, служба может работать обеими способами — она может быть запущенной (и работать в течение неограниченного времени) и допускать привязку. Это зависит от реализации пары методов обратного вызова: [onStartCommand()](https://developer.android.com/reference/android/app/Service.html#onStartCommand(android.content.Intent,%20int,%20int)) позволяет компонентам запускать службу, а [onBind()](https://developer.android.com/reference/android/app/Service.html#onBind(android.content.Intent)) позволяет выполнять привязку.

Независимо от состояния приложения (запущенное, привязанное или и оба сразу) любой компонент приложения может использовать службу (даже из отдельного приложения) подобно тому, как любой компонент может использовать операцию — запустив ее с помощью [Intent](https://developer.android.com/reference/android/content/Intent.html). Однако вы можете объявить закрытую службу в файле манифеста и заблокировать доступ к ней из других приложений. Более подробно это обсуждается в разделе [Объявление службы в манифесте](https://developer.android.com/guide/components/services#Declaring).

**Служба — это просто компонент, который может выполняться в фоновом режиме, даже когда пользователь не взаимодействует с приложением. Следовательно, вы должны создавать службу только в том случае, если вам нужно именно это.**

Помните, что если вы действительно используете службу, она выполняется в **основном потоке вашего приложения** по умолчанию, поэтому вы должны создать новый поток в службе, если она выполняет интенсивные или блокирующие операции.

Чтобы создать службу, необходимо создать подкласс класса [Service](https://developer.android.com/reference/android/app/Service.html) (или одного из существующих его подклассов). В вашей реализации необходимо переопределить некоторые методы обратного вызова, которые обрабатывают ключевые моменты жизненного цикла службы и при необходимости предоставляют механизм привязывания компонентов. Наиболее важные методы обратного вызова, которые необходимо переопределить:

[onStartCommand()](https://developer.android.com/reference/android/app/Service.html#onStartCommand(android.content.Intent,%20int,%20int))

Система вызывает этот метод, когда другой компонент, например, операция, запрашивает запуск этой службы, вызывая [startService()](https://developer.android.com/reference/android/content/Context.html#startService(android.content.Intent)). После выполнения этого метода служба запускается и может в течение неограниченного времени работать в фоновом режиме. Если вы реализуете такой метод, вы обязаны остановить службу посредством вызова [stopSelf()](https://developer.android.com/reference/android/app/Service.html#stopSelf()) или [stopService()](https://developer.android.com/reference/android/content/Context.html#stopService(android.content.Intent)). (Если требуется только обеспечить привязку, реализовывать этот метод не обязательно).

[onBind()](https://developer.android.com/reference/android/app/Service.html#onBind(android.content.Intent))

Система вызывает этот метод, когда другой компонент хочет выполнить привязку к службе (например, для выполнения удаленного вызова процедуры) путем вызова [bindService()](https://developer.android.com/reference/android/content/Context.html#bindService(android.content.Intent,%20android.content.ServiceConnection,%20int)). В вашей реализации этого метода вы должны обеспечить интерфейс, который клиенты используют для взаимодействия со службой, возвращая [IBinder](https://developer.android.com/reference/android/os/IBinder.html). Всегда необходимо реализовывать этот метод, но если вы не хотите разрешать привязку, необходимо возвращать значение null.

[onCreate()](https://developer.android.com/reference/android/app/Service.html#onCreate())

Система вызывает этот метод при первом создании службы для выполнения однократных процедур настройки (перед вызовом [onStartCommand()](https://developer.android.com/reference/android/app/Service.html#onStartCommand(android.content.Intent,%20int,%20int)) или [onBind()](https://developer.android.com/reference/android/app/Service.html#onBind(android.content.Intent))). Если служба уже запущена, этот метод не вызывается.

[onDestroy()](https://developer.android.com/reference/android/app/Service.html#onDestroy())

Система вызывает этот метод, когда служба более не используется и выполняется ее уничтожение. Ваша служба должна реализовать это для очистки ресурсов, таких как потоки, зарегистрированные приемники, ресиверы и т. д. Это последний вызов, который получает служба.

Если компонент запускает службу посредством вызова [startService()](https://developer.android.com/reference/android/content/Context.html#startService(android.content.Intent)) (что приводит к вызову [onStartCommand()](https://developer.android.com/reference/android/app/Service.html#onStartCommand(android.content.Intent,%20int,%20int))), то служба продолжает работу, пока она не остановится самостоятельно с помощью [stopSelf()](https://developer.android.com/reference/android/app/Service.html#stopSelf()) или другой компонент не остановит ее посредством вызова [stopService()](https://developer.android.com/reference/android/content/Context.html#stopService(android.content.Intent)).

Если компонент вызывает [bindService()](https://developer.android.com/reference/android/content/Context.html#bindService(android.content.Intent,%20android.content.ServiceConnection,%20int)) для создания службы (и [onStartCommand()](https://developer.android.com/reference/android/app/Service.html#onStartCommand(android.content.Intent,%20int,%20int)) *не* вызывается), то служба работает, пока к ней привязан компонент. Как только выполняется отмена привязки службы ко всем клиентам, система уничтожает службу.

**Показать, как создать сервис. Рассказать про**

**манифест,**

**exported,**

**onStartCommand**

**stopSelf()**

**Передачу параметров**

**void someTask() {**

**new Thread(new Runnable() {**

**public void run() {**

**for (int *i* = 1; *i* <= 5; *i*++) {**

**Log.*d*("LES", "i = " + *i*);**

**try {**

**TimeUnit.*SECONDS*.sleep(1);**

**} catch (InterruptedException *e*) {**

***e*.printStackTrace();**

**}**

**}**

**//stopSelf();**

**}**

**}).start();**

**}**

Параметры метода **onStartCommand**

Третий параметр – **startId**. Простыми словами – это **счетчик** вызовов startService пока сервис запущен. Т.е. вы запустили сервис методом startService, сработал метод onStartCommand и получил на вход startId = 1. Вызываем еще раз метод startService, сработал метод onStartCommand и получил на вход startId = 2. И так далее. Счетчик сбрасывается, когда сервис будет остановлен методами stopService, stopSelf и пр. После этого вызовы снова идут с единицы.

Но на самом деле алгоритм чуть другой. Сервис останавливается, когда последний полученный (а не последний обработанный) вызов выполняет метод stopSelf(startId). А при этом могут продолжать работать ранее полученные вызовы. Почему так сделано – я не знаю.  
Второй -

А второй параметр **flags** метода onStartCommand дает нам понять, что это повторная попытка вызова onStartCommand.

void readFlags(int flags) {   
switch (flags) {  
 case START\_FLAG\_REDELIVERY:   
case START\_FLAG\_RETRY:   
// restarted by system, might be kill app form stack.   
break; default:  
 // on regular startService call from client. } }

## Параметры запуска сервиса

[START\_NOT\_STICKY](https://developer.android.com/reference/android/app/Service.html#START_NOT_STICKY)

Если система уничтожает службу после возвращения из [onStartCommand()](https://developer.android.com/reference/android/app/Service.html#onStartCommand(android.content.Intent,%20int,%20int)), *не нужно* повторно создавать службу, если нет ожидающих доставки намерений. Это самый безопасный вариант, позволяющий избежать запуска вашей службы, когда это не нужно и когда ваше приложение может просто перезапустить любые незавершенные задания.

[START\_STICKY](https://developer.android.com/reference/android/app/Service.html#START_STICKY)

Если система уничтожает службу после возвращения из [onStartCommand()](https://developer.android.com/reference/android/app/Service.html#onStartCommand(android.content.Intent,%20int,%20int)), повторно создайте службу и вызовите [onStartCommand()](https://developer.android.com/reference/android/app/Service.html#onStartCommand(android.content.Intent,%20int,%20int)), но *не* передавайте последнее намерение повторно. Вместо этого система вызывает метод [onStartCommand()](https://developer.android.com/reference/android/app/Service.html#onStartCommand(android.content.Intent,%20int,%20int)) с намерением, которое имеет значение null, если нет ожидающих намерений для запуска службы. Если ожидающие намерения есть, они доставляются. Это подходит для мультимедийных проигрывателей (или подобных служб), которые не выполняют команды, а работают независимо и ожидают задание.

[START\_REDELIVER\_INTENT](https://developer.android.com/reference/android/app/Service.html#START_REDELIVER_INTENT)

Если система уничтожает службу после возвращения из [onStartCommand()](https://developer.android.com/reference/android/app/Service.html#onStartCommand(android.content.Intent,%20int,%20int)), повторно создайте службу и вызовите [onStartCommand()](https://developer.android.com/reference/android/app/Service.html#onStartCommand(android.content.Intent,%20int,%20int)) с последним намерением, которое было доставлено в службу. Все ожидающие намерения доставляются по очереди. Это подходит для служб, активно выполняющих задание, которое должно быть возобновлено немедленно, например, для загрузок файла.

## Связь с активити через BroadcastReceiver

показать! просто на примере. Взят со стартандроид

## [Биндинг. ServiceConnection](https://startandroid.ru/ru/uroki/vse-uroki-spiskom/162-urok-97-service-binding-serviceconnection.html)

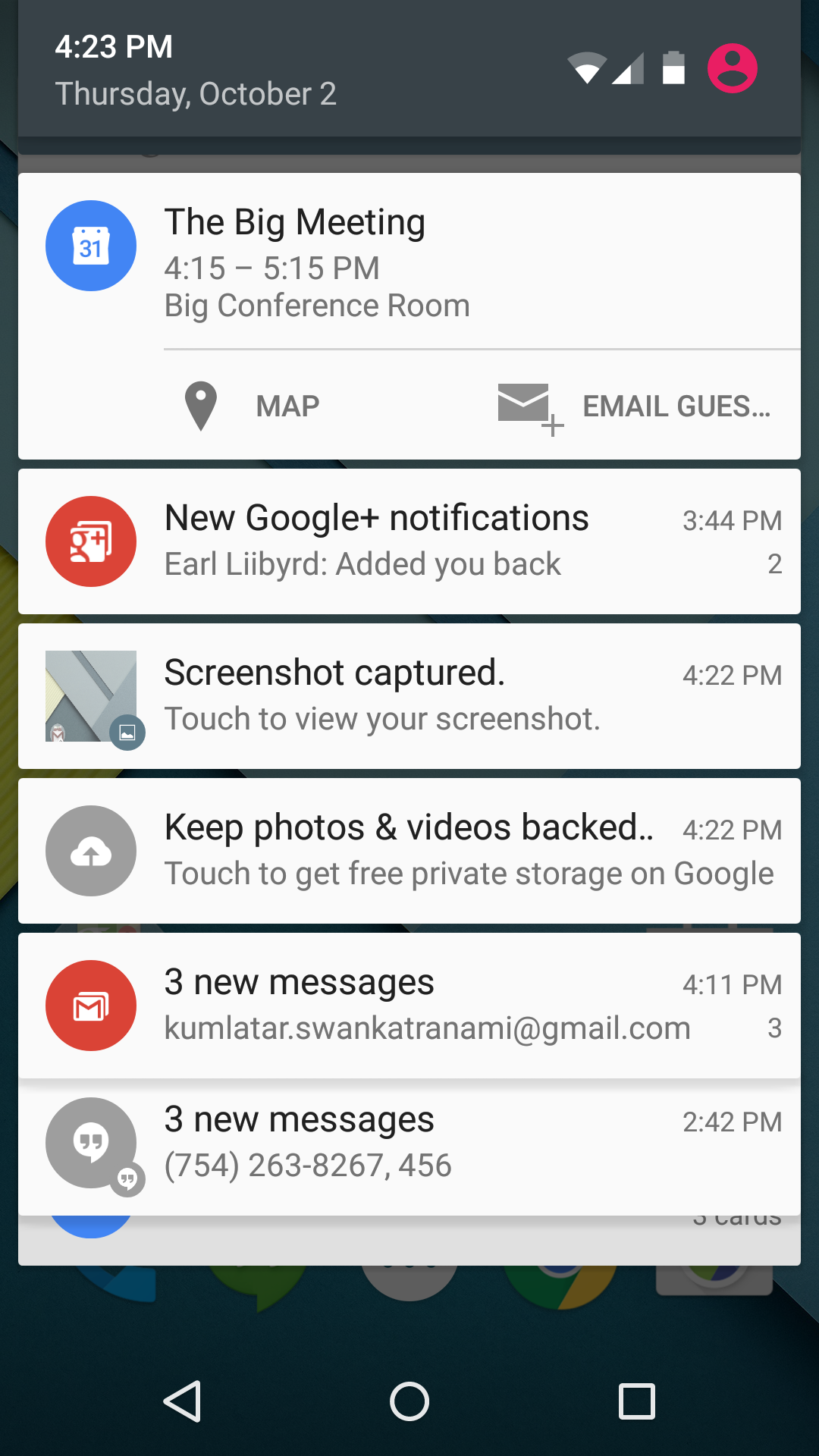
зачем это, показать примеры. Взят со стартандроид

Не нужно биндиится и анбиндится в onResume/onPause, рассказать почему

## Foreground service

Рассказать, тем отличаются.

## Уведомления



# 

Уведомление ― это сообщение, которое может быть выведено на экран за пределами обычного пользовательского интерфейса приложения. Когда вы сообщаете системе о необходимости выдать уведомление, оно сначала отображается в виде значка в **области уведомлений**. Чтобы просмотреть подробные сведения об уведомлении, пользователь открывает **панель уведомлений**. И областью уведомлений, и панелью уведомлений управляет система, а пользователь может их просматривать в любое время.

## Создание уведомления

Информация о пользовательском интерфейсе и действия для уведомления указываются в объекте[NotificationCompat.Builder](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html?hl=ru). Чтобы создать само уведомление, вызывается метод[NotificationCompat.Builder.build()](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html?hl=ru#build()), который возвращает объект [Notification](https://developer.android.com/reference/android/app/Notification.html?hl=ru), содержащий заданные вами спецификации. Чтобы выдать уведомление, объект [Notification](https://developer.android.com/reference/android/app/Notification.html?hl=ru) передается в систему путем вызова метода [NotificationManager.notify()](https://developer.android.com/reference/java/lang/Object.html?hl=ru#notify()).

### Обязательное содержимое уведомления

Объект [Notification](https://developer.android.com/reference/android/app/Notification.html?hl=ru) *должен* содержать следующие элементы:

* небольшой значок, заданный с помощью [setSmallIcon()](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html?hl=ru#setSmallIcon(int));
* заголовок, заданный с помощью [setContentTitle()](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html?hl=ru#setContentTitle(java.lang.CharSequence));
* подробный текст, заданный с помощью [setContentText()](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html?hl=ru#setContentText(java.lang.CharSequence)).

### Необязательные содержимое и настройки уведомления

Все прочие настройки и содержимое уведомления являются необязательными. Подробные сведения о них см. в справочной документации по [NotificationCompat.Builder](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html?hl=ru).

#### Задание.

Общая цель - сделать приложение, которое получает погоду и скачивает файл по ссылке во внутреннюю память.

1. Сделать активити на котором будет отображаться погода.
2. При старте активити должен запускаться Service через bind service. Т.е. сервис должен автоматически уничтожаться, когда уничтожается активити.
3. Каждую минуту сервис должен ходить по адресу и получать текущую температуру <http://api.openweathermap.org/data/2.5/weather?q=saransk&units=metric&appid=a924f0f5b30839d1ecb95f0a6416a0c2>   
   Результат о полученной погоде нужно передавать в активити, не используя широковещательные сообщения. Нужно делать, как описывается в этом посте: <https://stackoverflow.com/a/23587641/1120982> Т.е. в активити должна быть вью, которая обновляется после того, как сервис получил погоду
4. На активити добавить кнопку - скачать файл. При нажатии на кнопку должен запускаться сервис через startService() При запуске сервиса в него нужно передавать ссылку на .zip архив с картинкой. Сформировать архив нужно самостоятельно. Взять любую картинку, желательно размером побольше, зазиповать ее, и выложить в удобное для вас место, чтобы была прямая ссылка на файл.
5. Если закрыть активити, то сервис не должен завершаться. Сервис должен завершаться тогда, когда файл будет скачан и распакован.
6. Пометить сервис, как foreground сервис.
7. Алгоритм работы сервиса: при старте он получает в себя ссылку на архив, начинает скачиваться во внутреннюю память (чтобы не иметь дело с разрешениями) Во время скачки прогресс скачивания нужно отправлять в активити через широковещательные сообщения. Как только скачка завершилась, нужно, чтобы файл распаковался. После этого передать путь к распакованной картинке в активити и там отобразить ее в imageView. После этого сервис должен завершиться.

#### Полезные ссылки:

1. <https://developer.android.com/guide/components/services> - официальная докуемнтация по Сервисам
2. <https://developer.android.com/guide/components/bound-services> - оф. документация по привязанным сервисам
3. <https://developer.android.com/guide/components/broadcasts> - официальная документация по broadcast reciever
4. <https://startandroid.ru/ru/uroki/vse-uroki-spiskom/157-urok-92-service-prostoj-primer.html> - уроки по сервисам на старт андроид
5. Уведомления - <https://developer.android.com/guide/topics/ui/notifiers/notifications?hl=ru>
6. <https://developer.android.com/training/best-background> весь раздел