I'm not robot

reCAPTCHA

Continue

Android studio service start_sticky

public abstract class Service extends ContextWrapper implements ComponentCallbacks2 A Service is an application component representing a application's desire to perform longer operation, not interacting with the user or supply functions for other applications to use. Each service class must have the appropriate &lt;service&gt; declaration for your package AndroidManifest.xml. The service can be started by Context.startService() and Context.bindService(). Note that services, like other application objects, work in the main topic of their hosting process. This means that if your service is going to do any CPU intensive (such as MP3 recovery) or blocking (such as networks) operations, it should spawn your theme in which to do that job. More information about this can be found in processes and topics. The JobIntentService class is available as a standard service installation with its own thread in which it plans to do its job. Here are topics like: For a detailed discussion on how to create services, see the Service Developer's Guide. What is a service? Most of the confusion about the Service Class actually revolves around what it is not: The service is not a separate process. The Subject of the Service itself does not mean that it is acting in its own process; unless otherwise specified, it shall be carried out by the same process as the programme of which it is part. The service is not a thread. This is not a tool for itself to do work from the main thread (to avoid the program's unspeakable errors). So the service itself is actually very simple, providing two main features: an app device designed to tell the system about what it wants to do in the background (even if the user does not communicate directly with the app). This corresponds to calls to Context.startService(), which asks the system to schedule a job service, to be run until the service or someone else explicitly stop it. An app device designed to expose some of its features to other applications. This corresponds to calls context.bindService(), which allows you to connect to the service for a long time so that it can communicate with it. When a service component is actually created, for any of these reasons, all this system is actually an instant component and call your onCreate() and any other suitable callback on the main thread. It's up to the service to implement these with appropriate behavior, such as creating a secondary thread in which it does its job. Note that the service itself is as simple, you can make your interaction with it as simple or complicated as you want: from treating it as a local Java object that you direct method calls (as shown in the Local Services example), providing a full remote interface using AIDL. Service cycle There are two reasons why the service can be &lt;/service&gt;System. If someone calls Context.startService(), the system will receive the service (create it and, if necessary, call it the onCreate() method) and then call it the onStartCommand(Intent, int, int) method with the arguments provided by the client. The service will continue to run until Context.stopService() or stopSelf() is called. Note that multiple calls context.startService() does not have a socket (although they cause multiple corresponding calls onStartCommand()), so it does not matter how many times it starts the service will be stopped when Context.stopService() or stopSelf() is called; however, services may use their stopSelf(int) method to ensure that the service is not stopped until the initial intentions have been processed. When start services are two additional basic modes of operation that they may decide to run, depending on the value they return from OnStartCommand(): START_STICKY used for services that are clearly run and stopped, if necessary, and START_NOT_STICKY or START_REDELIVER_INTENT used for services that should remain run only for processing all commands sent to them. For more information about semantics, see the linked documentation. Customers can also use Context.bindService() to obtain a permanent connection to the service. This also creates a service if it is not working (when calling Create() at the same time) but does not encourageStartCommand(). The customer will receive the IBinder object so that the service returns from its onBind(intent) method, allowing the customer to then call back to the service. The service will continue until the connection is set up (regardless of whether the client retains the link service IBinder). Usually returned IBinder is a complex interface that was written aidl. The service may be started and has connections associated with it. In this case, the system will keep the service running until it is started or has one or more connections to it with the Context.BIND_AUTO_CREATE flag. When none of these situations are available, the service onDestroy() method is called and the service is effectively terminated. All cleaning (thread stop, deregistration receivers) must be completed upon return from onDestroy(). Universal access to the service can be exercised when it is published in the Declaration &lt;service&gt;tag. In this way, other applications will have to declare the appropriate &lt;uses-permission&gt;item in their manifest in order to start, stop, or link to the service. From Build.VERSION_CODES. GINGERBREAD using Context#startService(IntentService(Intent), you can also set intent #FLAG_GRANT_READ_URI_PERMISSION and/or intent #FLAG_GRANT_WRITE_URI_PERMISSION intentionally. This will give the Office temporary access to specific URI intentionally. Access will remain as long as Service&lt;/uses-permission&gt; &lt;/service&gt;until the Service is completely stopped. This helps to provide access to other apps that have not asked for permission to protect the Service, or even when the Service is not exported at all. In addition, the service can protect individual IPC calls to it with permissions by calling the ContextWrapper.checkCallingPermission (string) method before implementing that call. For more information about permissions and security in general, see the Security and Permissions document. Process cycle Android system will try to maintain the process in which the service will be installed until the service is started or customers are associated with it. When memory is missing and needing to kill existing processes, the priority of the process that contains the service will be greater from the following options: If the service currently executes code for its onCreate(), onStartCommand() or onDestroy() methods, then the hosting process will be a foreground process to ensure that this code can be executed without killing. If a service has been launched, its hosting process is considered less important than any processes that are currently visible to the user on the screen, but more important than any process invisible. Since only a few processes are usually visible to the user, this means that the service should not be fatal, except in memory scarcity conditions. However, since the consumer is not directly aware of the background service, that in the state he is considered a valid candidate to kill, and you should be prepared to make it happen. In particular, long-term services will increasingly be provided, which will be killed and guaranteed to be killed (and, if necessary, re-launched) if they are started sufficiently long. If there are customers associated with the service, the hosting process of the service is never less important than the most important customer. That is, if one of its clients is visible to the user, then the service itself is considered visible. The impact of customer relevance on the importance of the service can be adjusted using context#BIND_ABOVE_CLIENT, context#BIND_ALLOW_OOM_MANAGEMENT, context#BIND_WAIVE_PRIORITY, context#BIND_IMPORTANT, and context#BIND_ADJUST_WITH_ACTIVITY. Launched service can use the startForeground (int, android.app.Notification) API to put the service in a foreground state where the system thinks it is something the user actively knows, and therefore does not kill the candidate when there is a lack of memory. (In theory, it is theoretically possible that the service would be killed under high memory pressure due to the application of the current new knowledge, but in practice this should not be a cause for concern.) Note that most of the time your service works, it can be killed by the system if it is under heavy memory pressure. In this case, the system will then try to Service. An important consequence of this is that if you installStartCommand() to schedule a job that must be done asynchronously or in another topic, you may want to use START_FLAG_REDELIVERY to have the system re-introduce the intention to you so that it does not lose if your service is killed during processing. Other program components that work in the same process as the service (e.g. activities) can, of course, increase the importance of the whole process beyond the importance of the service itself. Local service example One of the most common uses of the service is a secondary component that works with other parts of the program in the same process as other components. All .apk components run in the same process, unless clearly stated otherwise, so this is a typical situation. When used this way, assuming that the components are in the same process, you can greatly simplify their interaction: service customers can simply throw IBinder they receive from it in a specific class that the service publishes. An example of this Use of the Service is shown here. The First Service itself that publishes a custom class when it is linked: Public Class LocalService extends the service to { private NotificationManager mNM; // Unique identification number for the message. // We use it in the message R.string.local_service_started and cancel it. Since we know this service always * works in the same process as its customers, we don't need to deal with *IPC. */ public class LocalBinder extends Binder { LocalService getService() { return LocalService.this; } @Override public void onCreate() { mNM = (NotificationManager)getSystemService(NOTIFICATION_SERVICE); // Show message at we begin. We placed an icon in the status bar. showNotification(); } @Override public int onStartCommand (Intent Target, Intent Flags, int startId) { Log.i (LocalService, Received start id + startId + : + intent); return START_NOT_STICKY; } @Override public voided onDestroy() { // Cancel standing message. Toast.makeText (this is, R.string.local_service_stopped, Toast.LENGTH_SHORT).show(); } @Override public IBinder onBind(Intention) { return mBinder; } // This is the object that receives customer actions. For more examples, see // RemoteService. private final IBinder mBinder = new LocalBinder(); /** * Display the message while this service is running. */ private void showNotification() { // In this example, we will use the same text for cursor and extended charsequence text = getText (R.string.local_service_started); // PendingIntent to start their activity if the user selects this message PendingIntent contentIntent = 0, new intention (this, LocalServiceActivities.Controller.class), 0); Set up information about the views displayed in the notification panel. Message = new Notification.Builder(this) .setSmallIcon(R.drawable.stat_sample) // status icon .setTicker(text) // status text .setWhen(System.currentTimeMillis()) // time stamp .setContentTitle(getText(R.string.local_service_label)) // record .setContentText(text) // record .setContentIntent(contentIntent) // Intended to send, when the record is clicked .build(); Send a message. mNM.notify (REPORT, REPORT); } } } You can now write a customer code that directly accesses the existing service, such as :** * Sample linking and untying location service. * oblige, get an object through which it can communicate with the service. * * Note that this is implemented as an internal class just to keep the sample * all together; this code is usually displayed in some individual classes. */ public static class mapping extends activity { // Do not attempt to dissois from the service unless the customer has received some information about the status of the service. To call a linked service, first make sure that this value is // not null. Private LocalService mBoundService; private ServiceConnection mConnection = new ServiceConnection() { public void onServiceConnected(ComponentName className, IBinder service) { // This is called when the connection to the service has been established // established, giving us a service object that we can use // communicate with the service. Since we are able to make clear//service that we know works in our own process, we can ///throw our IBinder into a specific class and directly access it. mBoundService = ((LocalService.LocalBinder)service).getService(); Tell the user about this in our demo. Toast.makeText (Binding.this, R.string.local_service_connected, Toast.LENGTH_SHORT).show(); } public void onServiceDisconnected(ComponentName className) { // This is called when the connection to the service was // unexpectedly disconnected - i.e. its process crashed. // Because it works in our same process, we should never // see it happen. Toast.makeText (Binding.this, R.string.local_service_disconnected, Toast.LENGTH_SHORT).show(); } }; void doBindService() (// Attempts to communicate with the service. We use the // clear class name because we want a specific service // implementation that we know will be displayed in our own process / / (and thus will not support component replacement of other // applications). if (bindService(Binding.this, LocalService.class), mConnection, Context.BIND_AUTO_CREATE)) { mShouldUnbind = true; } else { Log.e(MY_APP_TAG, Error: Requested service does not exist+ exists, or it access to it.); } } void doUnbindService() { if (mShouldUnbind) { // Release information about the status of the service. unbindService(mConnection); mShouldUnbind = false; } } } @Override protected on voiddestroy() { super.onDestroy(); doUnbindService(); } } } Remote Messenger Service Sample If you need to be able to write a service that can perform a complex connection to customers in remote processes (not just context #startService (intent) use to send commands to it), then you can use messenger class instead of writing all AIDL files. Here's an example of a service that uses Messenger as a client interface. First, the Service itself, messenger publishing to the internal manager when it is linked: Public class MessengerService extends the Service { /** Show and hide our message. */ NotificationManager mNM; /** Tracks all current registered clients. */ ArrayList&lt;Messenger&gt; mClients = new ArrayList&lt;Messenger&gt;(; /** Has the last client defined value. */ int mValue = 0; /** * The command to register the client receiving callback methods * from the service. The Sign-in message field must be the messenger for the * customer where callback messages should be sent. */ static final int MSG_REGISTER_CLIENT = 1; /** * Command to unregister the customer, ot stop receiving callback * from the service. The Message Response fieldThe Connection field must be Client Messenger, as previously provided with MSG_REGISTER_CLIENT. */ static final int MSG_UNREGISTER_CLIENT = 2; /** * Service command to set a new value. This can be sent to * the service to provide a new value, and will be sent to the service * to any registered customers with a new value. */ static final int MSG_SET_VALUE = 3; /** * Customer incoming message manager. Class IncomingHandler extends The Handler { @Override public void handleMessage (Message msg) { switch (msg.what) { case MSG_REGISTER_CLIENT: mClients.add (msg.replyTo); break; case MSG_UNREGISTER_CLIENT: mClients.remove (msg.replyTo); break; case MSG_SET_VALUE: mValue = msg.arg1; for (int i=mClients.size()-1; i&gt;=0; i--) { try { mClients.get(i).send(Message.obtain(null, MSG_SET_VALUE, mValue, 0)); } catch (RemoteException e) { // The client is dead. Remove it from the list; we are going through a list from back to front / so it is safe to do inside the loop. mClients.remove (i); } } break; default: super.handleMessage(msg); } } } / ** * Purpose for sending incominghandler messages to customers. */ ultimate messenger mMessenger = new Messenger (new IncomingHandler ()); @Override public void onCreate() (mNM = (NotificationManager)getSystemService(NOTIFICATION_SERVICE); // Show message about us starting @Override. message R.string.remote_service_started.&lt;/Messenger&gt; &lt;/Messenger&gt; &lt;/Messenger&gt; R.string.remote_service_stopped, Toast.LENGTH_SHORT). } /** * When connecting to the service, we return the interface to our messenger * to send messages to the service. */ @Override public IBinder onBind (intent) { return mMessenger.getBinder(); } /** * Display message when running this service. */ private void showNotification() (// In this example, we will use the same text ticker and extended message CharSequence text =getText(R.string.remote_service_started); // PendingIntent to start its activity if the user selects this message PendingIntent contentIntent = PendingIntent.getActivity (this, 0, new intent (this, Controller.class), 0); // Set up information about the views that appear in the notification panel. Message = new Notification.Builder(this) .setSmallIcon(R.drawable.stat_sample) // status icon .setTicker(text) // status text .setWhen(System.currentTimeMillis()) // time stamp .setContentTitle(getText(R.string.local_service_label)) // record .setContentText(text) // record .setContentIntent(contentIntent) // Intended to send, when the record is clicked .build(); Send a message. We use the line ID because it's a unique number. We use it later to cancel. mNM.notify (R.string.remote_service_started, message); } } If we want this service to work in a remote process (instead of the standard .apk), we can use android:process in your manifest to specify one: &lt;service android:name=.app. MessengerService android:process=:remote&gt;&lt;/service&gt; Please note that the remote name selected here is arbitrary, and you can use other names if you want additional processes. Prefix : Adds a name to the standard batch process name. With that done, customers can now link to the service and send messages to it. Please note that this allows customers to sign up with it to receive messages back as well: /** * Sample mandatory and unlink to remote service. * This indicates that the service that the customer will * associate communicates with him through the implementation of the aidl interface. * * Note that this is implemented as an internal class just to keep the sample * all together; this code is usually displayed in some individual classes. */ Boolean mIsBound; /** When the text view that we use to display state information. */ TextView mCallbackText; /** * Supervisor receives messages from the service. */ class IncomingHandler lengthens Handler { @Override public blank handleMessage (Message msg) (switch (msg.what) { case MessengerService.MSG_SET_VALUE: mCallbackText.setText(Received from service: + msg.arg1); break; default: super.handleMessage(msg); } } } /** * we publish to customers to send messages to IncomingHandler. */ ultimate messenger mMessenger = new Messenger (new IncomingHandler ()); /** * Class for interoperability with the main service interface. */ private ServiceConnection mConnection = new ServiceConnection() { public void onServiceConnected(ComponentName className, IBinder service) { // This is called when the connection to the service was // established, giving us a service object that we can use // communicate with the service. We communicate with our //service through the IDL interface, so get the customer side//representation of that object from the raw service. mService = new Messenger(service); mCallbackText.setText(Included;); We want to monitor the service until we//connected to it. try { Message msg = Message obtain(null, MessengerService.MSG_REGISTER_CLIENT); msg.replyTo=mMessenger; mService.send(msg); // Give it some value as a example. msg = Message.obtain(null, MessengerService.MSG_SET_VALUE, this.hashCode(), 0); mService.send(msg); } catch (RemoteException e) { // In this case, the service crashed before we could even // do something with it; Toast.makeText (Binding.this, R.string.remote_service_connected, Toast.LENGTH_SHORT).show(); } public void onServiceDisconnected(ComponentName className) { // This is called when the connection to the service was // unexpectedly disconnected - that is, its process crashed. mService = null; mCallbackText.setText (Disconnected.); // As part of the sample, tell the user what happened. Toast.makeText (Binding.this, R.string.remote_service_disconnected, Toast.LENGTH_SHORT).show(); } }; void doBindService() { // Make a connection to the service. We use a clear //class name because there is no reason to allow other // programs to change our component. bindService(Binding.this, MessengerService.class), mConnection, Context.BIND_AUTO_CREATE); mIsBound = True; mCallbackText.setText(Binding;); } void doUnbindService() { if (mIsBound) { // If we have received a service and therefore have registered // it, then now is the time to unregister the MessengerService.MSG_UNREGISTER_CLIENT. } / / / Disconnect our existing connection. unbindService(mConnection); mIsBound = false; mCallbackText.setText(Unbinding;); } } int START_CONTINUATION_MASK Bits returned onStartCommand (Intent, int, int) describing how to continue the service if it is killed. START_FLAG_REDELIVERY This flag is set int, int) if the intention is to re-delivery previously delivered intent, because the service previously returned START_REDELIVER_INTENT but was killed before calling stopSelf (int) for that intention. int START_FLAG_RETRY This flag is set onStartCommand (Intent, int, int) if the intention is to repeat because the initial attempt never got to or returned from onStartCommand (android.content.Intent, int, int). int START_NOT_STICKY Constant return from onStartCommand (Intent, int, int): if this service process is killed while it is started (returning from onStartCommand (intent, int, int)), and there is no new start intention to deliver to it, then take the service from the start state and not create until the future explicit call context #startService. int START_REDELIVER_INTENT Constant return from onStartCommand (Intent, int, int): if the process of this service is killed while it is started (returning from onStartCommand (intent, int, int)), then it will be scheduled again and the last delivered intention to re-deliver to it again via onCommand (intention, int, int). int START_STICKY Constant return from onStartCommand (Intent, int, int): if the process of this service is killed while it is started (after returning from onStartCommand (intent, int, int)), then leave it in the started state, but does not keep this delivered intention. int START_STICKY_COMPATIBILITY Constant return from onStartCommand (Intent, int, int): a compatibility version of START_STICKY that does not guarantee that onStartCommand (intent, int, int) will be called again after the murder. int STOP_FOREGROUND_DETACH flag stopForeground(int): If set, the message previously displayed in startForeground(int, Not) will be separated from the service. int STOP_FOREGROUND_REMOVE flag stopForeground(int): If set, the message previously displayed in startForeground(int, Not) will be removed. From the class android.content.Context String ACCESSIBILITY_SERVICE Use with getSystemService (java.lang.String) to get AccessibilityManager to provide user feedback about UI events through registered event listeners. String ACCOUNT_SERVICE Use with getSystemService (java.lang.String) to get an AccountManager to get intentions at your choice. String ACTIVITY_SERVICE Use with getSystemService (java.lang.String) to obtain activitymanager for global system state. String ALARM_SERVICE Use with getSystemService (java.lang.String) to get alarmmanager to get intentions at your choice. String APPWIDGET_SERVICE Use with getSystemService (java.lang.String) to get AppWidgetManager to access AppWidgets. String APP_OPS_SERVICE Use with getSystemService (java.lang.String) to obtain AppOpsManager monitoring program operations on the device. AUDIO_SERVICE The AUDIO_SERVICE Use with get AudioManager to handle the control volumes, ringer modes and audio route. String BATTERY_SERVICE Use with getSystemService (java.lang.String) to obtain a BatteryManager to manage battery status. int BIND_ABOVE_CLIENT flag for bindService(Intent, ServiceConnection, int): Indicates that the client application associated with this service considers that the service is more important than the application itself. int BIND_ADJUST_WITH_ACTIVITY flag for bindService(Intent, ServiceConnection, int): If linking out of an activity, allow the importance of the target service process to increase depending on whether the activity is visible to the user, regardless of whether another flag is used to reduce the amount that the overall importance of the client process is used to influence it. int BIND_ALLOW_OOM_MANAGEMENT flag for bindService(Intent, ServiceConnection, int): Allow the process running the associated service to pass through normal memory control. int BIND_AUTO_CREATE flag for bindService(In target, ServiceConnection, int): Automatically create the service as long as the binding is available. int BIND_DEBUG_UNBIND flag for bindService(Intent, ServiceConnection, int): include debugging help for unmatched calls to unbind. int BIND_EXTERNAL_SERVICE flag for bindService(Intent,ServiceConnection, int): The service is bound to an external service. int BIND_IMPORTANT flag bindService(In target, ServiceConnection, int): This service is very important for the customer and should therefore be moved to the foreground process level when the customer is present. int BIND_INCLUDE_CAPABILITIES flag for bindService(Intent, ServiceConnection, int): If linking from an application that has specific capabilities because of its foreground status, such as an activity or foreground service, then this flag will allow the linked app to get the same capabilities if it also has the necessary permissions. int BIND_NOT_FOREGROUND bindService(Intent, ServiceConnection, int): Prevents this mapping from being raised by the destination service process to the foreground planning priority. int BIND_NOT_PERCEPTIBLE flag for bindService(Intent, ServiceConnection, int): If the mapping from an application that is visible or user-perceived, reduce the importance of the target service below the noticeable level. int BIND_WAIVE_PRIORITY flag for bindService(Intent, ServiceConnection, int): Don't impact planning or memory management priority on the target service resource. String BIOMETRIC_SERVICE Use with getSystemService (java.lang.String) to obtain BiometricManager to handle biometric and PIN/model/password authentication. String BLOB_STORE_SERVICE Use with getSystemService (java.lang.String) to get BlobStoreManager to contribute and access data blobs from the blob store maintained by the system. String BLUETOOTH_SERVICE Use with getSystemService(java.lang.String) to retrieve a BluetoothManager for String Bluetooth. String CAMERA_SERVICE Use with getSystemService (java.lang.String) to get CameraManager to interact with camera devices. String CAPTIONING_SERVICE Use with getSystemService (java.lang.String) to get captioningmanager to get signature properties and listen to service signature preferences. String CARRIER_CONFIG_SERVICE Use with getSystemService (java.lang.String) to obtain carrierconfigmanager read operator configuration values. String CLIPBOARD_SERVICE Use with getSystemService (java.lang.String) to obtain ClipboardManager to access and change the contents of the global clipboard. String COMPANION_DEVICE_SERVICE Use with getSystemService (java.lang.String) to obtain CompanionDeviceManager control companion devices String CONNECTIVITY_DIAGNOSTICS_SERVICE Use with getSystemService (java.lang.String) to obtain ConnectivityDiagnosticsManager to perform network connection diagnostics as well as get network connection information from the system. String CONNECTIVITY_SERVICE Use with getSystemService (java.lang.String) to get ConnectivityManager to manage network connections. String CONSUMER_IR_SERVICE Use with getSystemService (java.lang.String) to get ConsumerIrManager to transmit infrared signals from the device. int CONTEXT_IGNORE_SECURITY flag to use with createPackageContext (string, int): Ignore all security restrictions in context requested, allowing it to always be loaded. int CONTEXT_INCLUDE_CODE flag to use with createPackageContext(String, int): Add application code with context. int CONTEXT_RESTRICTED flag to use with createPackageContext (string, int): Limited context can disable specific functions. String CROSS_PROFILE_APPS_SERVICE Use with getSystemService (java.lang.String) to obtain CrossProfileApps cross-profile transactions. String DEVICE_POLICY_SERVICE Use with getSystemService (java.lang.String) to obtain DevicePolicyManager to work with Global Device Policy Management. String DISPLAY_SERVICE Use with getSystemService(java.lang.String) to get DisplayManager to interact with display devices. String DOWNLOAD_SERVICE Use with getSystemService (java.lang.String) to get DownloadManager to request HTTP downloads. String DROPBOX_SERVICE Use with getSystemService (java.lang.String) to obtain DropBoxManager instances of write diagnostic logs. String EUICC_SERVICE Use with getSystemService (java.lang.String) to obtain EuiccManager to manage the eUICC (embedded SIM) of the device. String FILE_INTEGRITY_SERVICE Use with getSystemService(java.lang.String) to obtain FileIntegrityManager. String

FINGERPRINT_SERVICE Use with getSystemService (java.lang.String) to get fingerprintmanager to handle fingerprints. Line Use getSystemService(java.lang.String) to obtain the HardwarePropertiesManager to access the Hardware Properties service. String INPUT_METHOD_SERVICE Use getSystemService (java.lang.String) to obtain InputMethodManager to access input methods. String INPUT_SERVICE Use with getSystemService (java.lang.String) to obtain InputManager for interaction with input devices. String IPSEC_SERVICE Use with getSystemService (java.lang.String) to get IpSecManager for encryption sockets or networks with IPSec. String JOB_SCHEDULER_SERVICE Use with getSystemService (java.lang.String) to get the JobScheduler instance to manage sometimes background tasks. String KEYGUARD_SERVICE Use with getSystemService (java.lang.String) to get KeyguardManager to control keyboard protection. String LAUNCHER_APPS_SERVICE Use with getSystemService (java.lang.String) to receive LauncherApps requests and monitor run applications across user profiles. String LAYOUT_INFLATER_SERVICE Use with getSystemService(java.lang.String) to obtain layoutInflater for inflator layout resources in this context. String LOCATION_SERVICE Use with getSystemService(java.lang.String) to get LocationManager to control local updates. String MEDIA_PROJECTION_SERVICE Use with getSystemService (java.lang.String) to obtain mediaprojectionManager instances to manage media projection sessions. Line MEDIA_ROUTER_SERVICE Use with getSystemService(Class) to get MediaRouter to control and manage routing media. String MEDIA_SESSION_SERVICE Use with getSystemService (java.lang.String) to get MediaSessionManager to manage media sessions. String MIDI_SERVICE Use with getSystemService (java.lang.String) to obtain MidiManager to access the MIDI service. int MODE_APPEND file creation mode: Use with openFileOutput (String, int) if the file already exists, then write the data to an existing file instead of deleting it. int MODE_ENABLE_WRITE_AHEAD_LOGGING database open flag: When set up, the database opens with forward-write logging enabled by default. int MODE_MULTI_PROCESS This constant was outdated at API level 23. MODE_MULTI_PROCESS works reliably in some versions of Android, and there is no mechanism to match the modification of parallel processes. Apps should not try to use it. Instead, they should use a clear cross-process data management method, such as ContentProvider. int MODE_NO_LOCALIZED_COLLATORS database open flag: When set, the database is opened without support for localized collators. int MODE_PRIVATE File creation mode: The default mode in which only a caller (or all programs that share the same user ID) can access the created file. int MODE_WORLD_READABLE This constant was outdated at API level 17. Create files are very dangerous and can cause security holes in applications. It is very constrained; Instead, applications should use more formal mechanism interactions, such as ContentProvider, BroadcastReceiver, and Services. There is no guarantee that this access mode will remain in the file, for example, when it will be backed up and restored. int MODE_WORLD_WRITEABLE This constant was outdated at API level 17. Creating global recorded files is very dangerous and can cause security holes in applications. It is very constrained; Instead, applications should use more formal mechanism interactions, such as ContentProvider, BroadcastReceiver, and Services. There is no guarantee that this access mode will remain in the file, for example, when it will be backed up and restored. String NETWORK_STATS_SERVICE Use with getSystemService (java.lang.String) to receive NetworkStatsManager queries for network usage statistics. String NFC_SERVICE Use with getSystemService (java.lang.String) to obtain NfcManager by using NFC. String NOTIFICATION_SERVICE Use with getSystemService (java.lang.String) to obtain NotificationManager to inform the user about background events. String NSD_SERVICE Use with getSystemService (java.lang.String) to get NsdManager to handle the Web service discovery string POWER_SERVICE Use with getSystemService (java.lang.String) to get PowerManager under control of power management, including wake locks, which allows you to maintain the device while you are using long tasks. Lines PRINT_SERVICE print prints and print tasks in PrintManager. int RECEIVER_VISIBLE_TO_INSTANT_APPS flag for registerReceiver(BroadcastReceiver, IntentFilter): The recipient can accept broadcasts from snapshot programs. String RESTRICTIONS_SERVICE Use with getSystemService(java.lang.String) to obtain RestrictionsManager to obtain application restrictions and request permissions for restricted operations. String ROLE_SERVICE Use getSystemService (java.lang.String) to get RoleManager to manage roles. String SEARCH_SERVICE Use with getSystemService (java.lang.String) to get SearchManager to manage searches. String SENSOR_SERVICE Use with getSystemService (java.lang.String) to get sensormanager to access sensors. String SHORTCUT_SERVICE Use the getSystemService (java.lang.String) to obtain shortcutmanager to access the Startup Shortcut service. String STORAGE_SERVICE Use with getSystemService (java.lang.String) to obtain StorageManager to access system storage features. String STORAGE_STATS_SERVICE Use with getSystemService (java.lang.String) to obtain StorageStatsManager to access system retention statistics. String SYSTEM_HEALTH_SERVICE Use with getSystemService(java.lang.String) to retrieve a SystemHealthManager for accessing system health String SYSTEM_HEALTH_SERVICE power, memory, etc.). String TELECOM_SERVICE Use the device with getSystemService (java.lang.String) to obtain TelecomManager to manage telecommunications-related functions. String TELEPHONY_IMS_SERVICE Use with getSystemService (java.lang.String) to get ImsManager. String TELEPHONY_SERVICE Use with getSystemService(java.lang.String) to get telephonyManager to manage the control telephony functions of the device. String TELEPHONY_SUBSCRIPTION_SERVICE Use the getSystemService (java.lang.String) to obtain SubscriptionManager to manage the management telephony subscription device. String TEXT_CLASSIFICATION_SERVICE Use with getSystemService (java.lang.String) to obtain TextClassificationManager text classification services. String TEXT_SERVICES_MANAGER_SERVICE Use with getSystemService(java.lang.String) to obtain textservicesmanager to access text services. String TV_INPUT_SERVICE Use with getSystemService(java.lang.String) to get TvInputManager for interaction with the TV input device. String UI_MODE_SERVICE Use with getSystemService (java.lang.String) to obtain UiModeManager control UI modes. String USAGE_STATS_SERVICE Use with getSystemService (java.lang.String) to obtain UsageStatsManager requests for device usage statistics. String USB_SERVICE Use with getSystemService (java.lang.String) to obtain UsbManager access to USB devices (as a USB computer) and control the behavior of this device as a USB device. String USER_SERVICE Use with getSystemService (java.lang.String) to get a UserManager to manage users on devices that support multiple users. String VIBRATOR_SERVICE Use with getSystemService (java.lang.String) to get the vibrator interacting with vibration hardware. String VPN_MANAGEMENT_SERVICE Use with getSystemService (java.lang.String) to get VpnManager to manage profiles on the platform built-in VPN. String WALLPAPER_SERVICE Use with getSystemService (java.lang.String) to get com.android.server.WallpaperService to access wallpapers. String WIFI_AWARE_SERVICE Use with getSystemService (java.lang.String) to get WifiAwareManager handling management wi-fi aware. String WIFI_P2P_SERVICE Use with getSystemService (java.lang.String) to obtain WifiP2pManager handling management for Wi-Fi peer-to-peer connections. String WIFI_RTT_RANGING_SERVICE Use with getSystemService (java.lang.String) to get WifiRttManager for range devices with Wifi. String WIFI_SERVICE Use with getSystemService (java.lang.String) to obtain WifiManager management control for Wi-Fi access. String WINDOW_SERVICE Use the getSystemService (java.lang.String) to obtain WindowManager to access System Window Manager. final application getApplication() – Return the application that owns this service. final int If the service has become a foreground service by calling startForeground (int, android.app.Notification) or startForeground(int, android.app.Notification, int), getForegroundServiceType() returns the current foreground service type. abstract IBinder onBind(intent purpose) Return the communication channel to the service. Undo onConfigurationChanged (Configuration newConfig) is called by the system when the device configuration changes while your component is running. void onCreate() Called system when the service is created for the first time. undo onDestroy() Called the system to notify the Service that it is no longer in use and is removed. void onLowMemory() This is called when the common system runs with little memory, and active processes should crop up your memory usage. void onRebind (Intent of Intent) Called when new customers are connected to the service, after it was previously reported that all have been disconnected by their OnUnbind (intent). void onStart(Intent intent intent, int startId) This method was no longer used at API level 15. Implement OnStartCommand (android.content.Intent, int, int) instead. int onStartCommand(Intents, int notes, int startId) is called by the system every time a customer explicitly starts the service by calling Context.startService(intent), submitting arguments that it has provided and a unique integer token indicating the start request. Undo onTaskRemoved(Intent rootIntent) This is called if the service is currently running and the user has removed the task that is retrieved from the service application. void onTrimMemory (int level) Called when the operating system has found that this is a good time process for finishing unnecessary memory from your process. boolean onUnbind (Intent) Called when all customers disconnected from a certain interface posted service. final void startForeground (int id, Message message) If your service is running (run through Context#startService(Intent)), then also make this service run to the first plan by providing a permanent message that must be displayed to the user while in this state. final void startForeground (int id, message message, int foregroundServiceType) A crowded version startForeground (int, android.app.Notification) with additional foregroundServiceType parameter. final void stopForeground (int flag) Remove this service from the foreground state, so it must be killed if more memory is needed. final void stopForeground (boolean removeNotification). final void stopSelf() Stop the service if it was previously started. the final void stopSelf (int startId) Old stopSelfResult(int) version that does not return the result. final boolean stopSelfResult(int startId) Stop service if the last time it was startedId. From class void attachBaseContext for this context. boolean bindIsolatedService(In target service, int flag, line instanceName, executor, ServiceConnection conn) Variation bindService(intent, ServiceConnection, int), which in a specific case of individual services, allows the caller to generate multiple instances of the service from a single component declaration. boolean bindService (Intent Service, int Flags, Executor Executor, ServiceConnection conn) Same as bindService (android.content.Intent, android.content.ServiceConnection, int) with executor control of ServiceConnection callback. boolean bindService(Intent service, ServiceConnection conn, int flags) Access the application service if necessary. int checkCallingOrSelfPermission (string right) Determine whether the IPC call process or you have been granted some permission. int checkCallingOrSelfUriPermission(Uri uri, int modeFlags) Determine whether the IPC call process or you have been granted permission to access a specific URI. int checkCallingPermission(line permission) Determine whether the IPC you are managing has been granted a specific permission for the call process. int checkCallingUriPermission(Uri uri, int modeFlags) Determine whether the call process and user ID have the right to access a specific URI. int checkPermission (string permissions, int pid, int uid) Determine whether a given permission is allowed for a particular process and user ID run on the system. int checkSelfPermission(Line right) Determine whether you have been granted a certain permission. int checkUriPermission(Uri uri, String readPermission, String writePermission, int pid, int uid, int modeFlags) Check Uri and normal edition. int checkUriPermission(Uri uri, int pid, int uid, int modeFlags) – determine whether a particular process and user ID have been granted access to a specific URI. void clearWallpaper() This method is no longer used. Use WallpaperManager.clear() instead. This method requires the caller to hold Manifest.permission.SET_WALLPAPER permissions. Context createAttributionContext(String attributionTag) – Returns a new object in the current context but an attribute to another tag. Context createConfigurationContext(Configuration overrideConfiguration) – Returns a new entity in the current context, but whose resources are adjusted to match the specified configuration. Context createDeviceProtectedStorageContext() – Provides a new entity in the current context, but any storage API supports device-protected storage. Context CreationDisplayContext(Display display) – Present a new context object in the current context, but whose resources are adjusted to match the metrics of the specified display. The createPackageContext (string packageName, int flags) context provides a new context object that is specified in the application name. createWindowContext(int type, Bundle options) Creates a non-activity window context. String[] databaseList() – Returns an array of rows named private databases associated with an application package in this context. boolean deleteDatabase(String name) Delete the existing private SQLiteDatabase associated with the application package in this context. boolean deleteFile(String name) Delete the specified private file associated with the application package in this context. boolean deleteSharedPreferences(String name) Delete the existing shared preferences file. void enforceCallingOrSelfPermission (String permission, line message) If neither you nor the call process in the IPC you are handling has been granted some permission, quit SecurityException. void enforceCallingOrSelfUriPermission(Uri uri, int modeFlags, string message) If the IPC call process or you were not granted access to a specific URI, quit SecurityException. void enforceCallingPermission (String permission, string message) If the call process in the IPC you are handling has not been granted some permission, quit SecurityException. void enforceCallingUriPermission(Uri uri, int modeFlags, string message) If the call process and user ID were not granted access to a specific URI, quit SecurityException. void enforcePermission(string permission, int pid, int uid, string message) If the privilege is not allowed for a particular process and the user ID runs on the system, quit SecurityException. void enforceUriPermission(Uri uri, String readPermission, String writePermission, int pid, int uid, int modeFlags, string message) Run uri and normal rights. void enforceUriPermission(Uri uri, int pid, int uid, int modeFlags, string message) If a certain process and user ID were not granted access to a specific URI, quit SecurityException. String[] fileList() – Returns an array of rows named private files associated with a program package in this context. Context getApplicationContext() – Returns the context of one global application object in the current process. ApplicationInfo getApplicationInfo() – Returns all application information for this context package. AssetManager getAssets() Returns the AssetManager instance application package. The getBaseContext() File getCacheDir() Provides an absolute path to the application-specific cache directory in the file system. ClassLoader getClassLoader() - Return the class loader that you can use to scan classes in this package. The getCodeCacheDir() file provides an absolute path to the application-specific cache directory in the file system for storing the code in the cache. ContentResolver getContentResolver() - Return your application package to the ContentResolver instance. File getDataDir() Returns the absolute path to the directory, where all private files belonging to this program are stored. GetDatabasePath(String Name) file file system that stores a database created by using openOrCreateDatabase(String, int, SQLiteDatabase.CursorFactory). File getDir (String Name, int mode) Get, create, if necessary, a new directory where the program can embed its custom data files. Show getDisplay() Get the screen that is associated with this context. The getExternalCacheDir() file returns the absolute path to the application-specific directory on the main shared/external storage device where the program can embed cache files that it has. File[] getExternalCacheDirs() Returns absolute paths to application-specific directories on all shared/external storage devices where the program can embed cached files that it has. The getExternalFilesDir (String Type) file Provides an absolute path to the directory on the primary shared/external storage device where the program can place the permanent files it has. File[] getExternalFilesDirs(String type) Returns absolute paths to application-specific directories on all shared/external storage devices where the program can place the permanent files it has. File[] getExternalMediaDirs() This method is outdated. These directories still exist and are scanned, but developers are prompted to move the embedded content to the MediaStore collection directly because any program can directly contribute to new media to MediaStore without the necessary permissions, starting with Build.VERSION_CODES.Q. File getFileStreamPath (string name) Returns the absolute path in the file system where the file created with openFileOutput (String, int) is stored. File getFilesDir() Returns the absolute path to the directory in the file system where files created by openFileOutput(String, int) are stored. Executor getMainExecutor() – Return an executable file that will perform extracted tasks in the main topic associated with this context. Looper getMainLooper() Return looper key threads in the current process. The getNoBackupFilesDir() file returns an absolute path to a directory in a file system similar to getFilesDir(). File getObbDir() Return the primary shared/external storage directory where these programs' OBB files (if any) can be found. File[] getObbDirs() Returns absolute paths to application-specific directories on all shared/external storage devices where you can find OBB files (if any). Rows getPackageCodePath() Return all the way to the main Android package in this context. PackageManager getPackageManager() return PackageManager instance to find global package information. Line – Returns the package name for this program. Rows getPackageResourcePath() Return all the way to the main Android package in this context. Resources getResources() Returns resource instances of application applications SharedPreferences getSharedPreferences (String Name, Int Mode) Receive and store preferences for file name content by returning SharedPreferences, through which you can get and change your values. Object getSystemService(String Name) - Return the handle to the system-level service by name. The getSystemServiceName(Class&lt;?&gt; serviceClass) string is retrieved by the specified Resource.Theme getTheme() Return subject object associated with this context. Drawing getWallpaper() This method is no longer used. Use WallpaperManager.get() instead. int getWallpaperDesiredMinimumHeight() This method is outdated. Use WallpaperManager.getDesiredMinimumHeight() instead. int getWallpaperDesiredMinimumWidth() This method is no longer used. Use WallpaperManager.getDesiredMinimumWidth() instead. void grantUriPermission(String toPackage, Uri uri, int modeFlags) Grant access to a specific Uri in another package, regardless of whether the package has a common right to access uri content provider. boolean isDeviceProtectedStorage() Indicates whether this context storage API supports device-protected storage. boolean isRestricted() Indicates whether this context is restricted. boolean moveDatabaseFrom(Context sourceContext, String name) Move an existing database file from the specified source repository context to this context. boolean moveSharedPreferencesFrom(Context sourceContext, String name) Move the existing shared preferences file from the specified source repository context to this context. FileInputStream openFileInput(String name) Open the private file associated with the program pack for read in this context. FileOutputStream openFileOutput (String name, int mode) Open the private file associated with the application package for writing in this context. SQLiteDatabase openOrCreateDatabase (String name, int mode, SQLiteDatabase.CursorFactory factory, DatabaseErrorHandler errorHandler) Open a new private SQLiteDatabase associated with this context application package. SQLiteDatabase openOrCreateDatabase (String name, int mode, SQLiteDatabase.CursorFactory factory) Open a new private SQLiteDatabase associated with the application package in this context. Drawing peekWallpaper() This method is no longer used. Use WallpaperManager.peek() instead. Intent RegisterReceiver(BroadcastReceiver receiver, IntentFilter filter) Register BroadcastReceiver to run in the main activity thread. Intent RegistryReceiver(BroadcastReceiver receiver, IntentFilter filter, int flags) Register to receive intent broadcasts, while the receiver optionally encounters Instant Apps. receiver, IntentFilter Filter, String broadcastPermission, Handler Scheduler) Sign up to receive intent broadcasts that run in the planning context. void removeStickyBroadcast(Intent intent) This method is no longer used. Adhesive broadcasts should not be used. They do not give any security (anyone can access them), no protection (anyone can modify them), and many other problems. The recommended model is to use a non-stick broadcast to report something has changed, and another program mechanism to get the current value whenever desired. void removeStickyBroadcastAsUser(Intent intent intent, UserHandle user) This method is outdated. Adhesive broadcasts should not be used. They do not give any security (anyone can access them), no protection (anyone can modify them), and many other problems. The recommended model is to use a non-stick broadcast to report something has changed, and another program mechanism to get the current value whenever desired. void revokeUriPermission(Uri uri, int modeFlags) Remove all rights to access certain content provider Uri that were previously included with grantUriPermission (String, Uri, int) or any other mechanism. void revokeUriPermission(String targetPackage, Uri uri, int modeFlags) Remove permissions to access certain content provider Uri that was previously added with grantUriPermission (String, Uri, int) for a specific target package. void sendBroadcast(Intent intent intent, String receiverPermission) Broadcast the specified intent to all interested BroadcastReceivers, allowing optional required right to be enforced. void sendBroadcast(Intent intent) Broadcast given to all interested BroadcastReceivers. void sendBroadcastAsUser (Intent, UserHandle user) Version sendBroadcast (android.content.Intent) which allows you to specify the user of the broadcast will be sent. void sendBroadcastAsUser (Intent, UserHandle user, String receiverPermission) Version sendBroadcast (android.content.Intent, java.lang.String), which allows you to specify the user of the broadcast will be sent. void sendOrderedBroadcast(Intent intent, String receiverPermission, String receiverAppOp, BroadcastReceiver resultEas, Handler Planning, int initialCode, String initialData, Bundle initialExtras) Version sendOrderedBroadcast (android.content.Intent, java.lang.String, android.content.BroadcastReceiver, android.os.Handler, int, java.lang.String, android.os.Bundle), which allows you to specify app op op.strings to enforce the restrictions on which receivers broadcast will be sent. void sendOrderedBroadcast(Intent intent intent, int initialCode, String receiverPermission, String receiverAppOp, BroadcastReceiver Handler scheduler, String initialData, Bundle initialExtras, Bundle options) void sendOrderedBroadcast (Intent intent intent(Intent intent, String receiverPermission, BroadcastRece resultReceiver, resultReceiver, scheduler, int initialCode, String initialData, Bundle initialExtras) Version sendBroadcast (android.content.Intent) that allows you to get data back from the broadcast. void sendOrderedBroadcast(Intent intent intent, String receiverPermission) Broadcast specified intent to all interested BroadcastReceivers, delivering them one at a time, so that more preferred receivers use the broadcast before it is delivered to less preferred receivers. void sendOrderedBroadcastAsUser(intent purpose, UserHandle user, String receiverPermission, BroadcastReceiver resultReceiver, Handler planning, int initialCode, String initialData, Bundle initialExtras) Version sendOrderedBroadcast (android.content.Intent, java.lang.String, android.content.BroadcastReceiver, android.os.Handler, int, java.lang.String, android.os.Bundle), which allows you to specify the user to whom the broadcast will be sent. void sendStickyBroadcast(Intent intent) This method is no longer used. Adhesive broadcasts should not be used. They do not give any security (anyone can access them), no protection (anyone can modify them), and many other problems. The recommended model is to use a non-stick broadcast to report something has changed, and another program mechanism to get the current value whenever desired. void sendStickyBroadcastAsUser(Intent intent intent, UserHandle user) This method is outdated. Adhesive broadcasts should not be used. They do not give any security (anyone can access them), no protection (anyone can modify them), and many other problems. The recommended model is to use a non-stick broadcast to report something has changed, and another program mechanism to get the current value whenever desired. void sendStickyOrderedBroadcast(Intent intent intent, BroadcastReceiver resultReceiver, Handler scheduler, int initialCode, String initialData, Bundle initialExtras) This method is outdated. Adhesive broadcasts should not be used. They do not give any security (anyone can access them), no protection (anyone can modify them), and many other problems. The recommended model is to use a non-stick broadcast to report something has changed, and another program mechanism to get the current value whenever desired. void sendStickyOrderedBroadcastAsUser(Intent intent, UserHandle user, BroadcastReceiver resultReceiver, Handler Planning, int initialCode, String initialData, Bundle initialExtras) This method is outdated. Adhesive broadcasts should not be used. They do not give any security (anyone can access them), no protection (anyone can modify them), and many other problems. The recommended model is to use a non-stick to report that something has changed and another programme mechanism to obtain current value whenever desired. void setTheme(int resid) Set the main topic for this context. void setWallpaper(Rasmap rasmap) This method is outdated. Use Use Instead. This method requires the caller to hold Manifest.permission.SET_WALLPAPER permissions. void setWallpaper(InputStream data) This method is no longer used. Use WallpaperManager.set() instead. This method requires the caller to hold Manifest.permission.SET_WALLPAPER permissions. Undo startActivities(Intentions[] Intentions, Bundle Options) Run several new activities. void startActivities(Intent[i.e. same as startActivities(android.content.Intent[], android.os.Bundle) without options specified. void startActivity (Intention intent) Same as startActivity (android.content.Intent, android.os.Bundle) without any options specified. void startActivity(Intent intent intent, Bundle options) Start new activity. ComponentName startForegroundService (Intent Service) Similar to startService (android.content.Intent), but with the implicit promise that the Service will call startForeground (int, android.app.Notification) when it starts to run. boolean startInstrumentation(ComponentName className, Boolean startInstrumentation(ComponentName className, Boolean String ProfileFile, Package Arguments) Start executing instrumentation class. void startIntentSender (IntentSender intent, Intent Fill, Intent FlagMask, int flagValues, int extraFlags, Bundle Options) How to startActivity (android.content.Intent, android.os.Bundle), but given intentsender start. ComponentName startService(Intent service) - Request the specified application service to run. boolean stopService(Intent name) - Request a stop of the specified application service. unbindService(ServiceConnection conn) Disconnect from the application service. unregisterReceiver(BroadcastReceiver receiver) Unregister BroadcastReceiver. undo updateServiceGroup (ServiceConnection conn, int group, int importance) For a service previously associated with bindService (intent, ServiceConnection, int) or related method, change how the system manages that service process compared to other processes. From the class android.content.Context boolean bindIsolatedService (Intent Service, int flag, line instanceName, Executor, ServiceConnection conn) Variation bindService (Intent, ServiceConnection, int), which in a specific case of individual services, allows callers to generate multiple instance services from a single component declaration. boolean bindService (Intent Service, int Flags, Executor Executor, ServiceConnection conn) Same as bindService (android.content.Intent, android.content.ServiceConnection, int) with executor control of ServiceConnection callback. abstract boolean bindService (Intent Service, ServiceConnection conn, int flags) Connect to the Application Service to create it Need. boolean bindServiceAsUser (Intent Service, ServiceConnection conn, int flags, UserHandle user) connects to the service to a particular user in the same way as bindService (android.content.Intent, android.content.ServiceConnection, int). Abstract int checkCallingOrSelfPermission (string permission) Determine whether the invoicing process in IPC or you have been granted some permission. Abstract int checkCallingOrSelfUriPermission(Uri uri, int modeFlags) Determine whether the IPC call process or you have been granted permission to access a specific URI. Abstract int checkCallingPermission(string permission) Determine whether the invoicing process IPC you are working on has been granted some permission. Abstract int checkCallingUriPermission(Uri uri, int modeFlags) Determine whether the call process and user ID have the right to access a specific URI. Abstract int checkPermission(String permissions, int pid, int uid) Determine whether a particular process and user ID are allowed for a particular process and user ID that is executed in the system. Abstract int checkSelfPermission(line right) Determine whether you have been granted a certain permission. abstract int checkUriPermission(Uri uri, String readPermission, String writePermission, int pid, int uid, int modeFlags) Check Uri and normal edition. Abstract int checkUriPermission(Uri uri, int pid, int uid, int modeFlags) Determine whether a certain process and user ID have been granted permission to access a specific URI. Abstract void clearWallpaper() This method was no longer used api level 15. Use WallpaperManager.clear() instead. This method requires the caller to hold Manifest.permission.SET_WALLPAPER permissions. Context createAttributionContext(String attributionTag) – Returns a new object in the current context but an attribute to another tag. Abstract context createConfigurationContext(Configuration overrideConfiguration) – Returns a new entity in the current context, but whose resources are adjusted to match the specified configuration. Abstract context createContextForSplit(string splitName) – Returns a new context object for the given split name. Abstract context createDeviceProtectedStorageContext() – Provides a new context object in the current context, but any storage API supports device-protected storage. Abstract context createDisplayContext(Display display) – To present a new context object in the current context, any resource is adjusted to match the metrics of the specified display. The abstract createPackageContext (string packageName, int flag) provides a new context object that is specified in the application name. Create a contextWindowContext(int type, Grouping Options) Creates a non-activity window context. Abstract String[] databaseList() – Returns an array of rows named private databases associated with this Package. Abstract boolean deleteDatabase(String Name) Delete the existing private SQLiteDatabase associated with the application package in this context. Abstract boolean deleteFile(String Name) Delete the specified private file associated with the application package in this context. Abstract bull logic deleteSharedPreferences(String Name) Delete the existing shared preferences file. abstract void enforceCallingOrSelfPermission (String permission, line message) If neither you nor the invocation process in IPC you are handling has been granted some permission, quit SecurityException. abstract void enforceCallingOrSelfUriPermission(Uri uri, int modeFlags, string message) If the call process in IPC or you were not granted access to a specific URI, quit SecurityException. abstract void enforceCallingPermission (String permission, string message) If the call process in the IPC you are working on has not been granted a certain permission, quit SecurityException. abstract void enforceCallingUriPermission(Uri uri, int modeFlags, string message) If the call process and user ID were not granted access to a specific URI, quit SecurityException. abstract void enforcePermission (String permissions, int pid, int uid, string message) If the granted permission is not allowed for a particular process and the user ID runs on the system, quit SecurityException. abstract void enforceUriPermission(Uri uri, string readPermission, String writePermission, int pid, int uid, int modeFlags, string message) Run uri and normal rights. abstract void enforceUriPermission(Uri uri, int pid, int uid, int modeFlags, string message) If a certain process and user ID were not granted access to a specific URI, quit SecurityException. Abstract String[] fileList() – Returns an array of rows named private files associated with an application package in this context. Abstract context getApplicationContext() – Returns the context of one global application object in the current process. Abstract ApplicationInfo getApplicationInfo() - Return all application information in this context package. The abstract AssetManager getAssets() returns an instance of the AssetManager for the application package. Line getAttributionTag() Assignment can be used in complex applications to logically separate the application part. Abstract getCacheDir() Provides an absolute path to the application-specific cache directory in the file system. An abstract ClassLoader getClassLoader() Return Class Loader can use to retrieve classes in this package. Abstract getCodeCacheDir() Provides an absolute path to the application-specific cache directory in the file system for storing the code in the cache. final int getColor(int id) Returns the color associated with the current theme of the resource ID and style. final ColorStateList id) Returns a list of color statuses associated with a specific resource ID and listed for the current topic. Abstract ContentResolver getContentResolver() Return the ContentResolver instance to your application package. The abstract getDataDir() returns the absolute path to the directory in the file system where all private files belonging to this program are stored. Abstract getDatabasePath (String Name) File provides absolute path in the file system where the database is stored by using openOrCreateDatabase(String, int, SQLiteDatabase.CursorFactory). abstract file getDir (string name, int mode) Get, create, if necessary, a new directory where the program can embed its custom data files. Show getDisplay() Get the screen that is associated with this context. final drawable getDrawable(int id) Returns the tow object associated with a specific resource ID and style current theme. An abstract file getExternalCacheDir() Returns the absolute path to the application-specific directory of the primary shared/external storage device where the program can embed cached files it manages. Abstract File [] getExternalCacheDirs() Returns absolute paths to application-specific directories on all shared/external storage devices where the program can embed cached files that it has. The abstract getExternalFilesDir (String Type) Returns the absolute path to the directory on the primary shared/external storage device where the program can place the permanent files it has. Abstract file [] getExternalFilesDirs (String Type) Provides absolute paths to application-specific directories on all shared/external storage devices where the program can contain the permanent files it has. Abstract file [] getExternalMediaDirs() This method was no longer used at API level 30. These directories still exist and are scanned, but developers are prompted to move the embedded content to the MediaStore collection directly because any program can contribute to new media to media without the necessary permissions, starting with the Build.VERSION_CODES.Q. abstract file getFileStreamPath (string name) Returns the absolute path in the file system where the file created with openFileOutput (String, int) is stored. Abstract file getFilesDir() Returns the absolute path to the directory in the file system where files created using openFileOutput(String, int) are stored. Executor getMainExecutor() – Return an executable file that will perform extracted tasks in the main topic associated with this context. Abstract Looper getMainLooper() Return Looper for the current thread. an abstract file getNoBackupFilesDir() Returns the absolute path to the directory in the file system similar to getFilesDir(). abstract file getObbDir() Return shared /external storage directory where these programs' OBB files (if any) can be found. abstract file[] getObbDirs() Returns absolute paths to directories on all shared/external storage devices where you can find OBB files (if any). String getOpPackageName() – Returns the package name that should be used by AppOpsManager for calls from this context so that the application ops manager uid validation works with the name. Abstract String getPackageCodePath() Return all the way to this context's main Android package. Abstract PackageManager getPackageManager() Return packagemanager instances to find global package information. Abstract String getPackageName() - Provide a package name for this application. Abstract String getPackageResourcePath() Return all the way to this context's main Android package. Abstract resource getResources() Returns the resource instance of the application package. Abstract SharedPreferences getSharedPreferences (String Name, Int Mode) Get and store preferences for file contents by returning the name of the SharedPreferences through which you can get and change your values. Final Line GetString(int resId, Object... formatArgs) Returns a localized formatted row from the default row table in the application package that replaces the format arguments as defined in Formatter and String.format (String, Object...). final row getString(int resId) – Returns a localized row from the default line table in the application batch. final &lt;T&gt;T getSystemService(Class&lt;T&gt; serviceClass) - Return the handle to the system-level service by class. &lt;?? &gt; Final CharSequence getText (int resId) Return localized, style CharSequence from the application package to the default row table. Use WallpaperManager.get() instead. Abstract int getWallpaperDesiredMinimumHeight() This method was outdated API level 15. Use WallpaperManager.getDesiredMinimumHeight() instead. Abstract int getWallpaperDesiredMinimumWidth() This method was an outdated API level of 15. Use WallpaperManager.getDesiredMinimumWidth() instead. abstract void grantUriPermission (String toPackage, Uri uri, int modeFlags) Grant access to a specific Uri in another package, regardless of whether this package has a common right to access uri content provider. Abstract bull logic isDeviceProtectedStorage() – Indicates whether this context storage API supports device-protected storage. boolean isRestricted() Indicates whether this context is restricted. Abstract boolean moveDatabaseFrom(Context sourceContext, String Name) Move an existing database file from submitted&lt;/T&gt; &lt;/T&gt;storage context. Abstract bull logic moveSharedPreferencesFrom(Context sourceContext, String Name) Move the existing shared preferences file from the specified source repository context to this context. final TypedArray obtainStyledAttributes (AttributeSet set, int [] attrs) Get style attribute information in this context topic. final TypedArray obtainStyledAttributes (AttributeSet set, int [] attrs, int defStyleAttr, int defStyleRes) – Restore style attribute information in this context topic. final TypedArray obtainStyledAttributes(int resid, int[] attrs) Get style attribute information in this context topic. final TypedArray obtainStyledAttributes(int[] attrs) Get style attribute information in this context topic. Abstract FileInputStream openFileInput(string name) Open the private file associated with the program package for reading in this context. Abstract FileOutputStream openFileOutput (String Name, int Mode) Open the private file associated with the application package for writing in this context. abstract SQLiteDatabase openOrCreateDatabase (String name, int mode, SQLiteDatabase.CursorFactory factory, DatabaseErrorHandler errorHandler) Open a new private SQLiteDatabase associated with this context application package. Abstract SQLiteDatabase openOrCreateDatabase (String name, int mode, SQLiteDatabase.CursorFactory factory) Open a new private SQLiteDatabase associated with this context application package. Abstract drawable peekWallpaper() This method was no longer used at API level 15. Use WallpaperManager.peek() instead. void registerComponentCallbacks(ComponentCallbacks callbacks) Add new ComponentCallbacks to the main application context, which will be called at the same time as componentCallbacks methods of performance and other components called. Abstract Intent RegistryReceiver(BroadcastReceiver Receiver, IntentFilter Filter) Register BroadcastReceiver to run in the main activity thread. abstract intent registryReceiver (BroadcastReceiver receiver, IntentFilter filter, int flags) Register to receive intent broadcasts, while the receiver optionally encounters instant apps. Abstract intent registryReceiver (BroadcastReceiver receiver, IntentFilter filter, String broadcastPermission, Processing program scheduler, int flags) Register to receive intent broadcasts, run under the context of the scheduler. abstract intent registryReceiver(BroadcastReceiver receiver, IntentFilter Filter, Line broadcastPermission, Handler scheduler) Register to receive intent broadcasts, run in the context of planning. abstract void removeStickyBroadcast(Intent intent) This method was no longer used at API level 21. Adhesive broadcasts should not be used. They do not give any (anyone can access them), no protection (anyone can modify them), and many other problems. The recommended model is to use a non-stick broadcast to report a report something has changed, with another application mechanism to get the current value as soon as you wish. abstract void removeStickyBroadcastAsUser(Intent Target, UserHandle user) This method was no longer used api level 21. Adhesive broadcasts should not be used. They do not give any security (anyone can access them), no protection (anyone can modify them), and many other problems. The recommended model is to use a non-stick broadcast to report something has changed, and another program mechanism to get the current value whenever desired. abstract void revokeUriPermission(Uri uri, int modeFlags) Remove all rights to access certain content provider Uri that were previously included with grantUriPermission (String, Uri, int) or any other mechanism. abstract void revokeUriPermission (String toPackage, Uri uri, int modeFlags) Remove permissions to access certain content provider Uri that was previously added with grantUriPermission (String, Uri, int) for a specific target package. abstract void sendBroadcast(Intention intention, String receiverPermission) Broadcast the given goal to all interested BroadcastReceivers, allowing optional required right to be enforced. abstract void sendBroadcast(Intent) Broadcast given intent to all interested BroadcastReceivers. abstract void sendBroadcastAsUser (Intent, UserHandle user) Version sendBroadcast (android.content.Intent) which allows you to specify the user of the broadcast will be sent. abstract void sendBroadcastAsUser (Intent intent, UserHandle user, String receiverPermission) Version sendBroadcast (android.content.Intent, java.lang.String), which allows you to specify the user of the broadcast will be sent. void sendBroadcastWithMultiplePermissions(Intent intent, String [receiverPermissions) Broadcast given intent to all interested BroadcastReceivers, allowing array of required rights to be executed. void sendOrderedBroadcast(Intent intent intent, String receiverPermission, String receiverAppOp, BroadcastReceiver resultToo, Handler Planning, int initialCode, String initialData, Bundle initialExtras) Version sendOrderedBroadcast (android.content.Intent, java.lang.String, android.content.BroadcastReceiver, android.os.Handler, int, java.lang.String, android.os.Bundle), which allows you to specify the app op to enforce the restrictions on which the receivers broadcast will be sent. , String receiverPermission, BroadcastReceiver resultReceiver, Handler scheduler, int initialCode, String initialData, Bundle, initialExtras) Version sendBroadcast (android.content.Intent), which allows you to retrieve data back from the broadcast. abstract void sendOrderedBroadcast(Purpose of intent, tunerPermission) Broadcast the given intention to all interested BroadcastReceivers, deliver them one by one so that more can be more receivers to consume the broadcast before delivering it to less desirable receivers. abstract void sendOrderedBroadcastAsUser(intent purpose, UserHandle user, String receiverPermission, BroadcastReceiver resultReceiver, Handler planning, int initialCode, String initialData, Bundle initialExtras) Version sendOrderedBroadcast (android.content.Intent, java.lang.String, android.content.BroadcastReceiver, android.os.Handler, int, java.lang.String, android.os.Bundle), which allows you to specify the user to whom the broadcast will be sent. abstract void sendStickyBroadcast(Intent) This method was outdated API level 21. Adhesive broadcasts should not be used. They do not give any security (anyone can access them), no protection (anyone can modify them), and many other problems. The recommended model is to use a non-stick broadcast to report something has changed, and another program mechanism to get the current value whenever desired. abstract void sendStickyBroadcastAsUser(intent, UserHandle user) This method was outdated API level 21. Adhesive broadcasts should not be used. They do not give any security (anyone can access them), no protection (anyone can modify them), and many other problems. The recommended model is to use a non-stick broadcast to report something has changed, and another program mechanism to get the current value whenever desired. the abstract void sendStickyOrderedBroadcast(Intent intent, BroadcastReceiver resultReceiver, Handler scheduler, int initialCode, String initialData, Bundle initialExtras) This method was no longer used at API level 21. Adhesive broadcasts should not be used. They do not give any security (anyone can access them), no protection (anyone can modify them), and many other problems. The recommended model is to use a non-stick broadcast to report something has changed, and another program mechanism to get the current value whenever desired. abstract void sendStickyOrderedBroadcastAsUser(Intent intent, UserHandle user, BroadcastReceiver resultReceiver, Handler Scheduler, int initialCode, String initialData, Bundle initialExtras) This method was no longer used api level 21. Adhesive broadcasts should not be used. They do not give any security (anyone can access them), no protection (anyone can modify them), and many other problems. The recommended model is to use a non-stick broadcast to report something has changed, and another program mechanism to get the current value whenever desired. Abstract void setTheme (int resid) Set the main topic for this context. abstract void setWallpaper method was no longer used api level 15. Use WallpaperManager.set() instead. This method requires the caller to hold Manifest.permission.SET_WALLPAPER permissions. Abstract void setWallpaper(InputStream data) This method was no longer used api level 15. Use Use Instead. This method requires the caller to hold Manifest.permission.SET_WALLPAPER permissions. Abstract void startActivities(Intentions[] intentions, Bundle options) Start several new activities. abstract void startActivities(Intentions [] intentions) Same as startActivities (android.content.Intent [], android.os.Bundle) without options specified. abstract void startActivity (Intent) Same as startActivity (android.content.Intent, android.os.Bundle) without the options specified. abstract void startActivity(Intent intent, Bundle options) Start a new activity. Abstract ComponentName startForegroundService (Intent Service) Similar to startService (android.content.Intent), but with the implicit promise that the Service will call startForeground (int, android.app.Notification) when it starts to run. abstract bull logic startInstrumentation(ComponentName className, String ProfileFile, Package Arguments) Start executing instrumentation class. abstract void startIntentSender (IntentSender intent, Intent fillInIntent, int flagsMask, int flagsValues, int extraFlags) Same as startIntentSender (android.content.IntentSender, android.content.Intent, int, int, int, android.os.Bundle) without the specified options. abstract void startIntentSender (IntentSender intent, Intent fillInIntent, int flagsMask, int flagsValues, int extraFlags, Bundle options) As startActivity (android.content.Intent, android.os.Bundle), but given intentSender start. Abstract ComponentName startService(Intent service) Request the specified application service to run. abstract boolean stopService(Intent service) Request a stop for a particular application service. abstract unbindService(ServiceConnection conn) Disconnect from the application service. undo unregisterComponentCallbacks (ComponentCallbacks callback) Remove componentCallbacks object that was previously registered with registerComponentCallbacks (android.content.ComponentCallbacks). abstract void unregisterReceiver(BroadcastReceiver receiver) Unregister previously registered BroadcastReceiver. undo updateServiceGroup (ServiceConnection conn, int group, int importance) For a service previously associated with bindService (intent, ServiceConnection, int) or related method, change how the system manages that service process compared to other processes. From the class java.lang.Object Object clone() creates and returns a copy of this object. boolean equals(Object obj) Indicates whether any other object is equal to this object. void finalize() is called to the garbage collector object when the garbage collection determines that there are no more references to the object. final class&lt;?&gt; getClass() – Returns the runtime class for this object. int hashCode() Returns the hash code value of the object. object monitor. Eilutes toString() – Returns the row mapping of the object. final blank wait (long time-out, int nanos) Causes the current thread to wait until the next thread calls the notify() method or notifyAll() method of this object, or some other thread interrupts the current thread, or a certain real time has passed. final blank wait(long timeout) Causes the current thread to wait until the next thread is called by the notify() method or notifyAll() method for this object, or the specified time has passed. The final voided wait() causes the current thread to wait until the next thread calls the notify() method or notifyAll() method for this object. Constant society static final int START_NOT_STICKY Constant return from onStartCommand (intent, int, int): if this service process is killed while it is started (returning from onStartCommand (Intention, int, int)), and there is no new start intent to deliver to it, then take the service out of the started state and not create until the future explicit call of context #startService. The service will not receive an onStartCommand (android.content.Intent, int, int) call with zero intent because it will not restart if there is no pending intent to deliver. This regime makes sense for things that want to do some work as a result of starting, but can be stopped when memory pressures and clearly start yourself later to do more work. An example of such a service would be one that surveys data from the server: it could schedule a signal survey every N minutes having an alarm to start your services. When its onStartCommand (Intention, int, int) is called out of the signal, it plans a new signal N minutes later, and spawns a thread to perform on its network. If his process is killed during this check, the service will not restart until the alarm turns off START_REDELIVER_INTENT. int, int) if this service process is killed while it is started (returning from onStartCommand (intent, int, int)), then it will be res planned and the last delivered intention to re-deliver to it again viaStartCommand (intent, int, int). This intention will be reset until service calls stopSelf(int) with the start ID provided by OnStartCommand(In target, int, int). The service will not receive an onStartCommand (android.content.Intent, int, int) call with zero intent, because it will restart only if it is not completed to process all the intent sent to it (and all such pending events will be delivered at the restart site). Constant value: 3 (0x00000003) public static final int START_STICKY Constant return from onStartCommand(Intent, int, int): If the process of this service is killed and it is started (after from onStartCommand(Intention, int, int)), then leave it in the opened state, but without maintaining this delivered intention. Later, the system will try to re-create the service. Because it is in the start state, it will guarantee that you callStartCommand(Intent, int, int) when a new instance of the service is created; If there are no waiting start commands to be delivered to the service, they will be called with a zero object of intent, so you must make sure that you check this. This mode makes sense for things that will clearly start and stop arbitrary periods, such as a service that performs background music playback. Constant value: 1 (0x00000001) public barrel final int STOP_FOREGROUND_DETACH flag stopForeground(int): If set, the message previously submitted to startForeground(int, Message) will be separated from the service. It only makes sense when STOP_FOREGROUND_REMOVE not set up - in this case the message will remain displayed, but completely separate from the service and therefore no longer replaced, except for instant calls to the message manager. Constant value: 2 (0x00000002) Public Constructors Public Service () Public methods public methods public final int getForegroundServiceType() If the service has become a foreground service by calling startForeground (int, android.app.Notification) or startForeground (int, android.app.Notification, int), getForegroundServiceType() returns the current front service type. If foregroundServiceType is not specified in the declaration, the ServiceInfo#FOREGROUND_SERVICE_TYPE_NONE is returned. If the service is not a foreground service, the ServiceInfo#FOREGROUND_SERVICE_TYPE_NONE is returned. public abstract IBinder onBind (Intent) Return the communication channel to the service. May return null if customers are unable to sign in to the service. Returned IBinder is usually a complex interface that has been described using aidl. Note that unlike other components of the program, calls to the IBinder interface returned here may not occur in the main process thread. More information about the main thread can be found in processes and threads. Parameter purpose: The intention that was used to associate this service as specified in Context.bindService. Please note that any additional measures that were included in the intentions at the time will not be visible here. Returns IBinder to return the IBinder, through which customers can apply to the service. Public undo onConfigurationChanged (Configuration newConfig) is called by the system when the device configuration changes while your component is running. Note that, unlike activities, other components never restart when configuration changes: They always have to deal with the results of the change, such as retrieving resources. While this feature has been called, your resource object will be updated to resource values corresponding to the new new For more information, see Changes to the handling run time. Parameters in the newConfig configuration: A new device configuration. This value cannot be null. public voidedcreate () Called system when the service is first created. Do not call this method directly. public invalid onDestroy () Called the system to notify the Service that it is no longer in use and is removed. Currently, the service should clear all available resources (topics, registered receivers, etc.). When you return, there will be no more calls to this Service object and it is effectively dead. Do not call this method directly. public hollowmemory() This is called when the common system runs with little memory, and active processes should crop up your memory usage. Although the exact point at which this will be called is not defined, it will usually happen when the whole background process is killed. That is, before reaching the point of the killing processes hosting services and acquired interfaces, that we would avoid killing. You should install this method to release caches or other unnecessary resources that you can store. The system will perform garbage collection for you, returning from this method. It is desirable that you install ComponentCallbacks2#onTrimMemory from ComponentCallbacks2 to gradually unload your resources according to the various levels of memory needs. This API is api level 14 and higher, so you should only use this LowMemory() method as for older versions that can be treated in the same way as ComponentCallbacks2#onTrimMemory with ComponentCallbacks2 # TRIM_MEMORY_COMPLETE-level backup. public invalid onRebind (Intention of Intention) Called when new customers are connected to the service, after it was previously reported that all have been disconnected by their OnUnbind (intent). This will only be called if the implementation of onUnbind(Intent) has been ignored in order to bring the truth back. Parameter purpose: The intention that was used to associate this service as specified in Context.bindService. Please note that any additional measures that were included in the intentions at the time will not be visible here. public int onStartCommand (intent, intent flags, int startId) is called the system every time a customer explicitly starts the service by calling Context.startService(Intent), submitting the arguments it has submitted and a unique integer token representing the start request. Do not call this method directly. For backward compatibility, the default installation calls Start (Intent, int) and returns START_STICKY or START_STICKY_COMPATIBILITY. Please note that the system calls this in your service's main topic. The service's primary thread is the same thread where user interface operations are running in the same process. Always avoid extinguishing the loop of the main thread events. For long-term operations, network calls, or a large disk drive you should start a new thread or use AsyncTask. See also: public void onTaskRemoved (Intent rootIntent) This is called if the service is currently running and the user has removed the task that comes from the service application. If you have set ServiceInfo.FLAG_STOP_WITH_TASK then you will not receive this callback; instead, the service will simply be stopped. Settings rootIntent: The original root intent that was used to run a task that is removed. public boolean onUnbind (Intent) Called when all customers disconnected from a certain interface posted service. The default installation does nothing and returns false. Parameter purpose: The intention that was used to associate this service as specified in Context.bindService. Please note that any additional measures that were included in the intentions at the time will not be visible here. Returns the boolean return true if you want the onRebind(Intent) method to be later called when new customers are bound to it. public final void startForeground (int id, message message) If your service is running (run through Context#startService(Intent)), then also make this service run to the first plan by providing a regular message that must be displayed to the user while in this state. By default, the services started are background, which means that their process will not be provided for a foreground CPU planning (unless something in this process is foreground) and if the system has to kill them to recover more memory (for example, displaying a large page in a web browser), they may be killed without too much damage. You can use startForeground(int, Message) if killing your service would be dangerous for the user, for example, if your service performs background music playback, so the user would notice if their music stopped playing. Note that when calling this method, the service does not add to the start state itself, even if the name sounds like it. You always have to call ContextWrapper.startService (android.content.Intent) first to tell the system it should keep the service running, and then use this method to say that it works harder. Applications that use the API Build.VERSION_CODES. P or later must request permission Manifest.permission.FOREGROUND_SERVICE before you can use this API. Applications created using the SDK version Build.VERSION_CODES. Q or later you specify foreground service types by using the R.attr.foregroundServiceType attribute in the service item manifest file. The value of the R.attr.foregroundServiceType attribute can be multiple flags for ORed combined. See also: public final void startForeground (int id, message, int foregroundServiceType) Overloaded version startForeground (int, android.app.Notification) with additional Setting. Applications created using the SDK version Build.VERSION_CODES. Q or later can specify foreground service types types Service item of the R.attr.foregroundServiceType manifest file. The value of the R.attr.foregroundServiceType attribute can be multiple flags for ORed combined. The ForegroundServiceType parameter must be a subclass flag that is specified in the declaration attribute R.attr.foregroundServiceType, if not, illegalArgumentException is discarded. Specify the foregroundServiceType parameter as ServiceInfo.FOREGROUND_SERVICE_TYPE_MANIFEST use all the flags specified in the foregroundServiceType manifest attribute. Parameter id int: Identifier of this message by NotificationManager#notify(int, Message); must be 0. message: The message that will be displayed. This value cannot be null. foregroundServiceType int: There must be a subset of the flags of the declaration attribute R.attr.foregroundServiceType. The value is a combination of 0 or ServiceInfo.FOREGROUND_SERVICE_TYPE_MANIFEST, ServiceInfo.FOREGROUND_SERVICE_TYPE_NONE, ServiceInfo.FOREGROUND_SERVICE_TYPE_DATA_SYNC, ServiceInfo.FOREGROUND_SERVICE_TYPE_MEDIA_PLAYBACK, ServiceInfo.FOREGROUND_SERVICE_TYPE_PHONE_CALL, ServiceInfo.FOREGROUND_SERVICE_TYPE_LOCATION, ServiceInfo.FOREGROUND_SERVICE_TYPE_CONNECTED_DEVICE, ServiceInfo.FOREGROUND_SERVICE_TYPE_MEDIA_PROJECTION, ServiceInfo.FOREGROUND_SERVICE_TYPE_CAMERA, and ServiceInfo.FOREGROUND_SERVICE_TYPE_MICROPHONE See also: ServiceInfo.FOREGROUND_SERVICE_TYPE_MANIFEST public final boolean stopSelfResult (int startId) Stop the service if the last time it was started was startId. This is the same as calling Context.stopService (intent) for this specific service, but allows you to safely avoid stopping if you are to start a request from a client that you have not yet seen onStart (intent, int). Be careful to make your calls to this feature.. If you call this feature the latest ID you received before you call an earlier EMS ID, the service will still be stopped immediately. If you eventually process records in a row (for example, by sending them in separate threads), then you are responsible for stopping them in the same order that you received them. Returns the boolean returns true if the startId corresponds to the last start request and the service stops, the other is false. Protect Methods to Protect Void AttachBaseContext (Context newBase) Set the primary context of this ContextWrapper. All calls will then be forwarded to the main context. An IllegalStateException occurs if the primary context is already set. Parameters newBase context: A new basic context for this package. protected by an empty dump (FileDescriptor fd, PrintWriter Writer, String [args) print service status to the specified stream. This will be called if you run the ADB shell dumpsys activity &lt;yourservicename&gt;service (note that this command is running, the service must be started, and you must specify a fully qualified service This means that &lt;/yourservicename&gt; &lt;/yourservicename&gt; of dumpsys, &lt;servicename&gt;which only runs on named system services and which relies on the IBinder # dump method for the IBinder interface registered servicemanager. Parameters fd FileDescriptor: Raw file description to which the dump file is sent. writer printwriter: A printwriter to which you need to dump your state. It will close for you when you return. args string: Additional arguments for the dump request. Request. &lt;/servicename&gt;

normal_5f94aa6f29059.pdf
normal_5f8c62dbd065d.pdf
normal_5f8729f468acf.pdf
normal_5f89b5b13fcdf.pdf
manual lego batman 4526
footej camera old version apk
ecss threaded fasteners handbook pdf
howard miller wall clock winding instructions
0 pagador de promessas pdf
abc letter writing practice book pdf
yamaha clavinova cvp 103 manual
guinness book of world record 2020 pdf
remote desktop for android download
supersu pro apkpure
lovecraft rats in the walls pdf
best android phone for spoofing pokemon go
como funciona una bobina de tesla pdf
counter assault forces mod apk android 1
d1d3c9182.pdf
c04dc84919bdfb5.pdf