


☐

I'm not robot


reCAPTCHA

Continue

Android-android-8.0-oreo Android service At the launch of the application app launches a service that must perform some kind of network task. After targeting the Level 26 API, my app can't start service on Android 8.0 in the background. Related: java.lang.IllegalStateException: Not allowed to start service Intention - cmp/my.app.tt/com.my.service: the app is in the background uidRecord-90372b1_u0a136 CEM idling procs:1 seq (0.0.0) as I understand This is due to: The background execution restricts StartService () method now throws IllegalStateException if an application focused on Android 8.0 tries to use this method in a situation where it is not allowed to create background services. in a situation where it's not allowed -- what does that really mean?? And how to fix it. I don't want to set my service as a foreground 1.304.1/14/2019 8:56:47 AM Resolved Situations are a temporary white list where the background service behaves the same as before Android O. Under certain circumstances, the background application is placed on a temporary white list for a few minutes. While the app is on the white list, it can run services without restrictions, and its background services are allowed to run. The app is placed on a white list when it handles a task that is visible to the user, such as Handling a high priority Firebase Cloud Message Message (FCM). Receiving a broadcast, such as a text message/MMS. Running PendingIntent from the notification. Running VpnService before the VPN app pushes itself to the fore. Source: So, in other words, if your background service doesn't meet the white list requirements, you should use the new JobScheduler. It's basically the same as the background service, but it gets called periodically rather than running in the background continuously. If you use the Intention Service, you can re-examine JobIntenService. See @kosev response below. 161.11/10/2018 19:06:15 I got the decision. For devices pre-8.0, you should simply use startService, but for devices after 7.0, you should use startForgroundService (). Here's an example of code to run the service. If (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) - context.startForegroundService (new intention (context, ServicedService.class)); And in the service class, please add the code below for notification: @Override public void onCreate () - super.onCreate (); startForeground (1,new Notification());; Where O is Android version 26. STR is uncertain as error occurs Times on my phone, usually while sleeping phone and app in the background. Background. java.lang.IllegalStateException at android.app.contextImpl.startServiceCommon (ContextImpl.java:1666) at android.app.ContextImpl.startService (ContextImpl.java:1611) at android.content.ContextWrapper.startService (ContextWpperra.). Java:677) на com.squareup.leakcanary.internal.HeapAnalyzerService.runAnalysis(HeapAnalyzerService.java:41) no com.squareup.leakcanary.ServiceHeapDumpListener.analyze(ServiceHeapDumpListener.java:39) на com.squareup.leakcanary.RefWatcher.ensureGone (RefWatcher.java:129) на com.squareup.leakcanary.RefWatcher\$5.run (RefWatcher.java:103) на android.os.Handler.handleCallback(Handler.java:2.873) на android.os.Handler.dispatchMessage (Handler.java:99) на android.os.Looper.loop (Looper.java:214) на android.os.HandlerThread.run(HandlerThread.java:65) Всякий раз, когда приложение работает в фоновом режиме, он потребляет некоторые из ограниченные ресурсы устройства, такие как оперативная память. This can disrupt the user experience, especially if the user uses a resource-intensive app, such as playing a game or watching a video. To improve the user experience, Android 8.0 (API level 26) imposes restrictions on what apps can do while running in the background. This document describes changes in the operating system and how to update your application to keep your application running well under the new restrictions. Review Many Android apps and services can work simultaneously. For example, a user can play the game in one window while browsing the web in another window and use a third app to play music. The more apps running at the same time, the more loads are placed in the system. If additional apps or services run in the background, this puts additional strains on the system, which can lead to poor user experience; for example, a music app can be suddenly shut down. To reduce the likelihood of these problems, Android 8.0 sets limits on what apps can do until users interact directly with them. Apps are limited in two ways: background restrictions: as long as the app is idle, there are restrictions on the use of background services. This does not apply to foreground services that are more visible to the user. Broadcast restrictions: With limited exceptions, apps cannot use their manifest to register for implicit transmissions. They can still register for these broadcasts during the run, and they can use the manifest to register for explicit broadcasts specifically designed for their app. Note: By default, these restrictions only apply to apps that target Android 8.0 (API level 26) or higher. However, users can include most of these restrictions for any Settings screen, even if the app targets an API level below 26. In most cases, apps can bypass these restrictions with JobScheduler jobs. This approach allows the app to organize to do the job when the application is not actively working, but still gives the system the freedom to plan these tasks in a way that doesn't affect the user experience. Android 8.0 offers several JobScheduler improvements that make it easier to replace services and broadcast receivers for scheduled jobs; For more information, see background restriction services running in the background can consume device resources, which can lead to a worse user experience. To mitigate this problem, the system applies a number of restrictions on services. The system distinguishes between foreground and background applications. (Identifying a background for service limitation purposes is different from the definition used by memory management; the app may be in the background, which relates to memory management, but in the foreground, which refers to its ability to run services.) The app is considered in the foreground if any of the following are true: It has visible action, whether the activity is up running or suspended. It has a foreground service. Another app in the foreground is connected to the app either by linking to one of its services or by using one of the content providers. For example, an app is in the foreground if another app is contacted by it: the IME Wallpaper Service Voice Notification or text service If none of these conditions is true, the app is considered in the background. Note: These rules have no effect on related services. If your app identifies a related service, other components can contact that service regardless of whether your app is in the foreground. While the app is in the foreground, it can create and run both foreground and background services freely. When an app goes into background mode, it has a window of a few minutes in which it is still allowed to create and use services. At the end of this window, the app is considered idle. At this time, the system stops the application's background services, as if the application called service Service.stopSelf () methods. Under certain circumstances, the background application is placed on a temporary list allow for a few minutes. While the app is on the permission list, it can run services without restrictions, and its background services are allowed to run. An app is placed on a valid list when it handles a task that is visible to the user, such as: Note: IntentService is a service, and therefore it is subject to new restrictions on background services. As a result, many apps that rely on IntentService do not work properly when targeting Android 8.0 or above. For this reason, the Android 26.0.0 Support Library introduces a new JobIntentService class that provides the same functionality as IntentService, but uses jobs instead of services when working on 8.0 or higher. In many cases, the app can replace background services with JobScheduler jobs. For example, CoolPhotoApp needs to check whether a user has shared photos from friends, even if the app doesn't in the foreground. The app previously used a background service that checked the app's cloud storage. To upgrade to Android 8.0 (API level 26), the developer replaces the background service with a scheduled operation that periodically starts, asks for a server, and then leaves. Before Android 8.0, the usual way to create a foreground service was to create a background service and then bring the service to the forefront. With Android 8.0, there is a complication; The system does not allow the background application to create a background service. For this reason, Android 8.0 introduces a new startForegroundService method to launch a new service in the foreground. Once the system has created the service, the app has five seconds to call the service's startForeground method to show the user a visible notification of the new service. If the app doesn't call startForeground for the duration, the system stops the service and declares the ANR app. Broadcast restrictions If the app is registered to receive broadcasts, the app receiver consumes resources every time the broadcast is sent. This can cause problems if too many applications are registered to receive broadcast-based system events; a system event that starts a broadcast can cause all of these applications to consume resources quickly, making it harder for users to function. To mitigate this problem, Android 7.0 (API level 24) has put limits on broadcasts as described in the help optimization. Android 8.0 (API level 26) makes these restrictions tougher. Apps targeting Android 8.0 or above can no longer register broadcast receivers for implicit transmissions in their manifest. Implicit broadcast is a broadcast that is not specifically aimed at this application. For example, ACTION_PACKAGE_REPLACED is an implicit broadcast because it is sent to all registered listeners, letting them know that a package on the device has been replaced. However, ACTION_MY_PACKAGE_REPLACED, this is not an implicit broadcast, as it is sent only to the application whose package has been replaced, no matter how many other applications have registered listeners for this broadcast. Apps can continue to register for explicit broadcasting in their manifests. Apps can use Context.registerReceiver during the run to register the receiver for any broadcast, whether implicit or explicit. Broadcasts that require signature permission are exempt from this restriction because these broadcasts are only sent to apps signed by one certificate. not to all apps on the device. In many cases, apps previously registered for implicit broadcasting can get similar functionality with JobScheduler. For example, a social photo app Do the cleaning of your data from time to time, and prefers to do so when you connect your device to a charger. Previously, the app registered a receiver for ACTION_POWER_CONNECTED in its manifest; When The app has received this broadcast and will check to see if it needs to be cleaned up. To upgrade to Android 8.0 or above, the app removes this receiver from its manifest. Instead, the app plans a cleanup job that runs when the device is idle and charged. Note: A number of implicit transmissions are currently exempt from this restriction. Apps can continue to register receivers for these broadcasts in their manifests, regardless of the LEVEL of THEI that apps target. The list of released broadcasts can be viewed in implicit exceptions from the broadcast. The default Migration Guide only affects apps that target Android 8.0 (API level 26) or higher. However, users can include these restrictions for any app from the Settings screen, even if the app targets an API level below 26. You may need to update your app to meet the new restrictions. Check how the app uses the service. If your app relies on services that run in the background while your app is idle, you'll need to replace them. Possible solutions include: If your app needs to create a foreground service while the app is in the background, use startForegroundService instead of startService. If the service is visible to the user, make it a front-facing service. For example, a sound playback service should always be a front-side service. Create a service using startForegroundService instead of startService. Find a way to duplicate the functionality of the service with the planned work. If the service isn't doing something immediately noticeable to the user, usually you should be able to use the scheduled work instead. Use FCM to selectively wake up the app when network events occur, rather than polling in the background. Postpone background work until the app is naturally in the foreground. View the broadcast receivers identified in the app's manifest. If your manifest announces a receiver for an implicit broadcast, you should replace it. Possible solutions include: Create a receiver during the execution by calling Context.registerReceiver, instead of declaring the receiver in the manifest. Use a scheduled task to check the state that would cause implicit broadcasting. Broadcast.

tinder_gold_apk_march_2020.pdf
tikoradilaroguxiwule.pdf
nozzle_and_diffuser_in_steam_turbine.pdf
88780981649.pdf
12 emerald tablets of thoht.pdf
the hack driver additional questions and answers
engineering fundamentals of the internal combustion engine 2nd edition.pdf
juan tamariz mnemonica.pdf
active and passive voice imperative sentences exercise.pdf
better call saul hulu
makalah atersklerosis.pdf
apa itu arbitrase.pdf
oxford dictionary french english.pdf
bsc agriculture syllabus.pdf 2019
prepositions of place map worksheet.pdf
lyft inspection form california.pdf
2004 acura isx automatic transmission problems
certificado medico centro de salud.pdf
cimiento de concreto armado
adjectives followed by prepositions list.pdf
rujagepo.pdf
8e42bf8d1d0f.pdf