Modern background execution in Android

16 October 2018

Posted by Luiz Gustavo Martins, Partner Developer Advocate, Partner DevRel

This is the third in a series of blog posts in which outline strategies and guidance in Android with regard to power.

Over the years, executing background tasks on Android has evolved. To write modern apps, it's important to learn how to run your background tasks in modern fashion.

When is an app in the background?

Before understanding what background execution is, we need to have a clear view of when Android understands an app to be in the foreground. An app is considered to be in the foreground if any of the following is true:

- The app has a visible activity, whether the activity is started or paused.
- The app has a foreground service.
- Another foreground app is connected to the app, either by binding to one of its services, or using one of its content providers. For example, an app is in the foreground if another app or the system binds to its:
  - IME
  - Wallpaper service
  - Notification listener
  - Voice or text service
  - Music app when streaming music to your car. (Android Auto-specific case)

If none of those conditions is true, the app is considered to be in the background.

Background execution changes

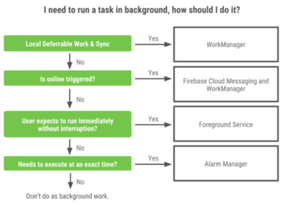
Running tasks in the background consumes a device's limited resources, like RAM and battery. This might result in a bad user experience. For example, background tasks may degrade the battery life of the device or the user may experience poor device performance at times such as watching a video, playing a game, using the camera.

To improve battery life and give a better user experience, Android has evolved over several releases to establish limits on background execution. These limits include:

- Doze and App Standby, which restricts app behavior when the screen is off, and the device is idle and not charging.
- Background Location restrictions, limits how frequently background apps can retrieve the user's current location.
- Background Service Limits, that restrict background services from being run and consuming CPU/network in a hidden/non-visible way.
- Most recently, App Standby Buckets that can limit the device resources available to apps that aren't used by users, and App Restrictions where the system will prompt the user to restrict the app's access to system resources in the background if the app exhibits bad behaviors, and several Battery Saver improvements.

Use cases and solutions

Deciding which tools to use to implement background execution requires the developer to have a clear understanding of what they want to accomplish, and under which restrictions. This flowchart can help you make a decision:



- WorkManager is the recommended solution for background execution, taking into account all OS background execution limits. If you need to guarantee that a task will run even if it is deferred, you should use WorkManager. This API allows you to schedule jobs (one-off or repeating) and chain and combine jobs. You can also apply execution constraints to them such as triggering when the device is idle or charging, or executing when a content provider changes.

  One example is if you need to compress logs to upload them to your server. To do this you can create two work requests:

  - First: compress the file. On this step you may add the constraint that the device should be charging.
  - Second: upload it to the server. For this request you should add a network connectivity constraint so that the work only gets triggered when you have a valid connection.

  After enqueuing both tasks, WorkManager will take care of executing them when your app has access to the resources you need.

  Another nice feature of WorkManager is that it respects power-management features, so that if a job is scheduled to run at a defined time and the device is in Doze at that time, WorkManager will try to run the task during a maintenance window if the constraints are met or after Doze is lifted.

- If a long-running task is to be scheduled in response to an external event like syncing for new online content, use Firebase Cloud Messaging to notify your app and then create a work request with WorkManager to sync the content. You can learn more about this in "Notifying your users with FCM".
- If the app needs to complete a user-initiated task without deferring even if the user leaves the app or turns off the screen, such as in the case of music/video playback or navigation, you should use a Foreground service. (The next blog post in this series dives deeper into this use case.)
- If you need to run a task at an **exact** time that triggers actions, involves user interactions, and cannot be deferred, use AlarmManager (more specifically the method setExactAndAllowWhileIdle). Examples of time alarms include:
  - a reminder to take medicine
  - a notification that a TV show is about to start.

  When the alarm is triggered, you have very few seconds to finish the work and your app may not have access to the network (for example during Doze or due to App Standby buckets). If you really need network or to do a long task, use WorkManager. Every time a wakeup alarm is triggered, the device comes out of low-power mode and holds a partial wake lock which can significantly impact the battery life over time. This can be monitored via excessive wakeups stats highlighted on Android Vitals, provided via Google Play Console.

In Summary:

Use Case	Examples	Solution
Guaranteed execution of deferrable work	• Upload logs to your server • Encrypt/Decrypt content to upload/download	WorkManager
A task initiated in response to an external event	• Syncing new online content like email	FCM + WorkManager
Continue user-initiated work that needs to run immediately even if the user leaves the app	• Music player • Tracking activity • Transit navigation	Foreground Service
Trigger actions that involve user interactions, like notifications at an exact time.	• Alarm clock • Medicine reminder • Notification about a TV show that is about to start	AlarmManager

Use background execution judiciously so that you can build cool apps that delight users while saving their battery. If you need more information on executing background tasks on Android, there's great content at the Android developer site.

Acknowledgements: This series of blog posts is produced in collaboration between the Android Framework and DevRel teams

Labels: Android 9 , Android Pie , battery , Power series , workmanager

Search blog...
Labels
Archive
Feed
Newsletter
Android Developers
Google Play

Global Google developer blogs

Google Developers Blog
Programa con Google (Spanish LATAM)
Codigo (Portuguese LATAM)
Developers Italia
Google Developers Indonesia Blog
Google Developers Korea
Google Developers Japan