App hibernation

Archived: 2026-04-02 11:55:24 UTC

If your app targets Android 11 (API level 30) or higher, and the user doesn't interact with your app for a few months, the system places your app in a *hibernation* state. The system optimizes for storage space instead of performance, and the system protects user data. This system behavior is similar to what occurs when the user manually force-stops your app from system settings.

Effects of hibernation

As shown in table 1, the effects of hibernation depend on your app's target SDK version, as well as the device on which your app is running:

Table 1. Effects of hibernation on your app

Target SDK version	Characteristics of device	Hibernation effects
Android 12 or higher	Runs Android 12 or higher	Your app's runtime permissions are reset. This action has the same effect as if the user viewed a permission in system settings and changed your app's access level to **Deny**. Your app can't run jobs or alerts from the background. Your app can't receive push notifications, including high-priority messages that are sent through Firebase Cloud Messaging. Any files in your app's cache are removed.
Android 11	Runs Android 11	Your app's runtime permissions are reset.
Android 11	Runs Android 6.0 (API level 23) to Android 10 (API level 29), inclusive, and is powered by Google Play services	Your app's runtime permissions are reset. This behavior takes effect in December 2021. Learn more in this blog post about making permissions auto-reset available to billions more devices.

System behavior when an app leaves hibernation

When the user next interacts with your app, your app exits hibernation, and it can create jobs, alerts, and notifications again.

However, the system doesn't do the following for your app:

1. Re-grant your app's runtime permissions.

   The user must re-grant these permissions for your app.

2. Reschedule any jobs, alerts, and notifications that were scheduled before your app went into hibernation.

   To support this workflow more easily, use WorkManager. You can also add rescheduling logic in the `ACTION_BOOT_COMPLETED` broadcast receiver, which is invoked when your app leaves hibernation and after the device boots up.

App usage

The following sections provide examples of app usage, as well as examples of actions that the system doesn't consider to be app usage.

Examples of app usage

When an activity in your app is resumed, the system considers this event to be a user interaction. Therefore, the system extends the amount of time before your app enters hibernation.

On Android 11 and higher, the following behaviors are also considered to be user interactions:

- The user interacts with a widget.
- The user interacts with a notification, except for dismissing the notification.

It should be noted that app usage for hibernation doesn't explicitly require user interaction. As long as a component of the package is invoked, it is still considered app usage. Some examples of this include:

- Apps that have a service or content provider bound by another app on the device or the OS. For example, Input Method Editors (IMEs) or password managers.
- Broadcast receivers in the package receiving an explicit broadcast from an external package.

Non-examples

If your app only ever exhibits the behaviors described in the following list, your app enters hibernation after a few months:

- Runs a scheduled job using `JobScheduler`.
- Receives an implicit broadcast.

- Schedules alarms.

System exemptions from hibernation

Android grants system-level exemptions from app hibernation in certain use cases. If your app falls into one of the following categories, it is exempt from the app usage standards and will not hibernate.

Apps not displayed on the launcher
: Any app that doesn't have an active shortcut tile on the launcher.

Work profile apps
: Any app that a user installs on a work profile. Note that if the same app also resides on a personal profile, only the work profile app is exempt.

Device policy controllers
: Apps that control local device policies and system applications on devices.

Carrier privileged apps
: Any app that mobile phone carriers pre-load on devices and deem necessary for contractual service obligations, for example, voicemail or customer service apps.

3p installer apps
: Third-party app stores for automatic updates of their installed apps when necessary.

User exemptions from hibernation

If you anticipate that a core use case in your app is affected by hibernation, you can request an exemption from app hibernation from the user. This exemption is useful for situations in which the user expects your app to work primarily in the background, even without the user interacting with your app, such as when your app does any of the following:

- Provide family safety by periodically reporting the location of family members.
- Sync data between a device and your app's server.
- Communicate with smart devices, such as a TV.
- Pair to companion devices, such as a watch.

To request an exemption, complete the steps in the following sections.

Check whether the user has already disabled hibernation for your app

To check whether the user has already disabled hibernation for your app, use the `getUnusedAppRestrictionsStatus()` API.

For additional details on how to use this API in your app, see the API code example on this page.

Ask the user to disable hibernation for your app

If the user hasn't already disabled hibernation for your app, you can send a request to the user. To do so, complete these steps:

1. Display a UI that explains to the user why they need to disable hibernation for your app.

2. Invoke the `createManageUnusedAppRestrictionsIntent()` API, as shown in the API code example. This API creates an intent that loads the **App info** screen in Settings. From here, the user can turn off hibernation for your app.

   It is important that you call `startActivityForResult()`, not `startActivity()`, when sending this intent.

   As shown in table 2, the option's location and name depends on the characteristics of the device on which your app is installed:

Table 2. Option that disables hibernation for your app

Characteristics of device	**Page where the option appears**	**Name of the option to turn off**
Runs Android 13 or higher	**App info**	**Pause app activity if unused**
Runs Android 12	**App info**	**Remove permissions and free up space**
Runs Android 11	**App info > Permissions**	**Remove permissions if app isn't used**
Runs Android 6.0 to Android 10, inclusive, and is powered by Google Play services	**Play app > Menu > Play Protect > Permissions for Unused Apps**	**Remove permissions if app isn't used**

API code example

This code example shows how to check whether hibernation is enabled for your app, and the correct way to ask users to disable hibernation for your app.

```
val future: ListenableFuture<Int> =
    PackageManagerCompat.getUnusedAppRestrictionsStatus(context)
future.addListener({ onResult(future.get()) }, ContextCompat.getMainExecutor(context))

fun onResult(appRestrictionsStatus: Int) {
  when (appRestrictionsStatus) {
    // Couldn't fetch status. Check logs for details.
    ERROR -> { }

    // Restrictions don't apply to your app on this device.
    FEATURE_NOT_AVAILABLE -> { }

    // The user has disabled restrictions for your app.
    DISABLED -> { }
```

```
    // If the user doesn't start your app for a few months, the system will
    // place restrictions on it. See the API_* constants for details.
    API_30_BACKPORT, API_30, API_31 -> handleRestrictions(appRestrictionsStatus)
  }
}

fun handleRestrictions(appRestrictionsStatus: Int) {
  // If your app works primarily in the background, you can ask the user
  // to disable these restrictions. Check if you have already asked the
  // user to disable these restrictions. If not, you can show a message to
  // the user explaining why permission auto-reset or app hibernation should be
  // disabled. Then, redirect the user to the page in system settings where they
  // can disable the feature.
  val intent = IntentCompat.createManageUnusedAppRestrictionsIntent(context, packageName)

  // You must use startActivityForResult(), not startActivity(), even if
  // you don't use the result code returned in onActivityResult().
  startActivityForResult(intent, REQUEST_CODE)
}
```

Legacy platform API

The operating system also includes an API to interact with the hibernation feature. However, the API only works on devices that run Android 11 or higher; the API doesn't handle the hibernation features that are backported to earlier Android versions. Therefore, we don't recommend using the API.

If you need to continue using the API temporarily for compatibility purposes, the following list shows how to use it:

- To check if hibernation is disabled for your app: `isAutoRevokeWhitelisted()`
- To send the user to the hibernation settings page: create an Intent using `ACTION_APPLICATION_DETAILS_SETTINGS`

Manually invoke hibernation behavior

To test how your app behaves after the system places your app in a hibernation state, complete the following steps:

1. *(Android 12 and higher only)* Enable the hibernation behavior on your device:

   ```
   adb shell device_config put app_hibernation app_hibernation_enabled true
   ```

2. Set the default amount of time that the system waits to enter hibernation. That way, you can restore it after testing:

```
threshold=$(adb shell device_config get permissions \
  auto_revoke_unused_threshold_millis2)
```

3. Reduce the amount of time that the system waits. In the following example, the system is modified such that your app enters hibernation only one second after you stop interacting with the app:

```
adb shell device_config put permissions \
  auto_revoke_unused_threshold_millis2 1000
```

4. Wait for any boot-time broadcasts to finish on your test device by running the following command:

```
adb shell am wait-for-broadcast-idle
```

   When the broadcasts are finished, this command returns the message: `All broadcast queues are idle!`

5. Invoke the app hibernation process manually:

```
adb shell cmd jobscheduler run -u 0 -f \
  com.google.android.permissioncontroller 2
```

6. *(Android 12 and higher only)* Confirm that the app is hibernated, using one of the following methods:

   - Observe that the test device now shows a notification, indicating that unused apps are hibernated.
   - Run the following command:

     ```
     adb shell cmd app_hibernation get-state PACKAGE-NAME
     ```

7. Restore the default amount of time that the system waits before it places your app into hibernation:

```
adb shell device_config put permissions \
  auto_revoke_unused_threshold_millis2 $threshold
```

Source: https://developer.android.com/topic/performance/app-hibernation