DARRYN CAMPBELL

Mobile computing and enterprise software development

Keeping your application running when the device wants to sleep – updated for Android P

9th July 2018 6 By DARRYNCAMPBELL

Back in May 2017 I wrote a post on the Zebra developer portal on "keeping your Android application running when the device wants to sleep", it covered Android Marshmallow and Nougat devices and was designed to address concerns over Doze mode and its impact on applications being moved from Lollipop or KitKat devices. There is a lot of detail in that post but the conclusion was that if you whitelisted your application against Doze mode and acquired both a WiFi and partial wake lock the application would continue run in the background without interference from the Operating System.

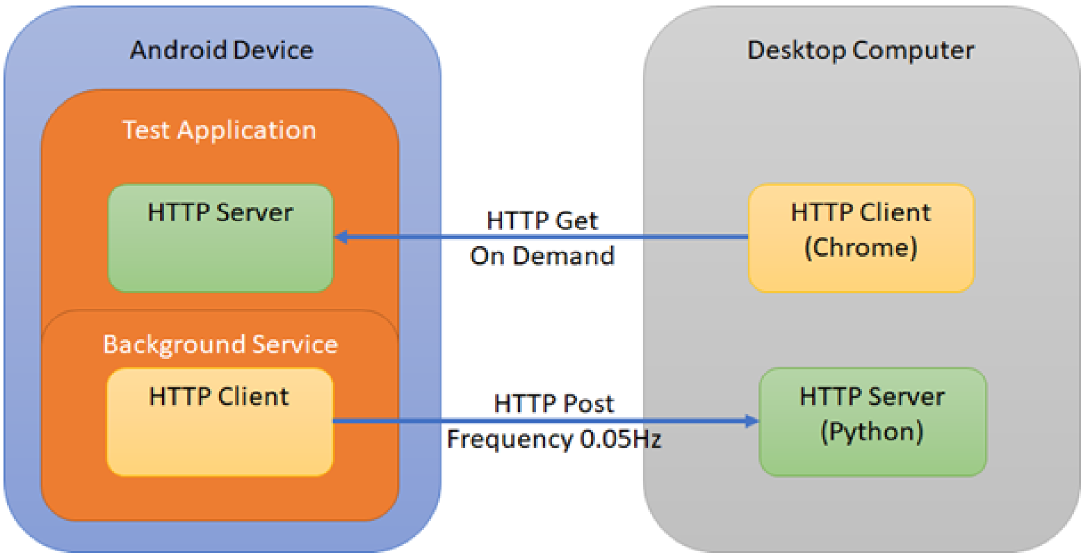
Fast forward over a year and both Android Oreo and Android P have introduced a slew of new restrictions imposed on applications that want to run in the background:

- Oreo background execution limits, specifically the virtual prohibition of background services and limitations on location update frequencies.
- Android P power management features, including App Standby Buckets which makes use of Android Vitals and integration with the Battery saver
    - An appendix summary exists for the power management restrictions applied under Android P. It shows the interactions between all the various power limitations and quantifies some of the time scales involved, though with the caveat that these are subject to change.

Remember that these restrictions are in addition to the restrictions imposed by Doze mode which was introduced in Marshmallow and improved upon in Nougat.

Test setup

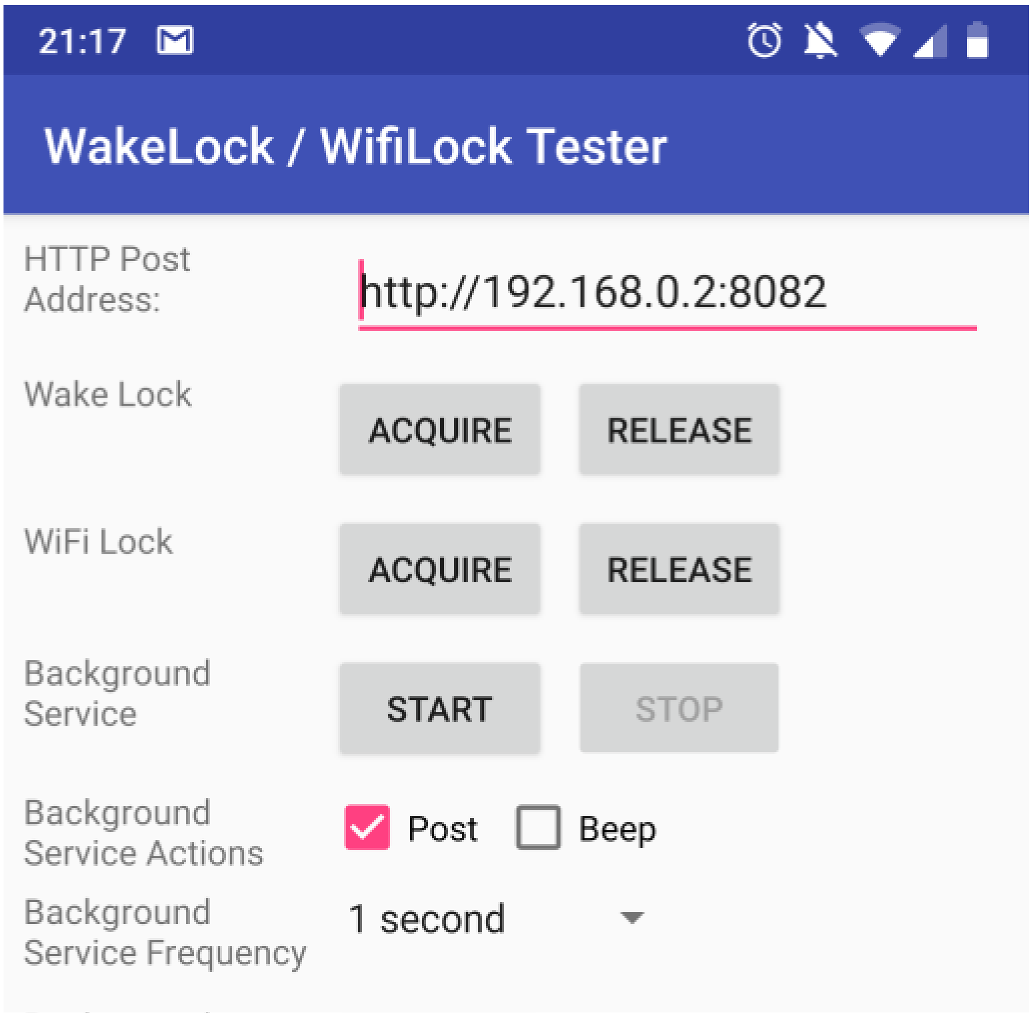
The test setup is as follows (this architecture is unchanged from last year's tests):



A single application on the test device spawns both an HTTP client and an HTTP server.

- The HTTP server can respond to requests from a web browser running on a remote machine, this shows the application is able to respond to network requests and simulates a push message (though in reality push messages are not implemented with HTTP)
- The HTTP client is spawned in a background service (an IntentService) and will continually send HTTP POSTs to a remote server, this tests the applications ability to perform CPU work as well as the longevity of the background service and the ability to establish an outbound network connection.

The application is available from my personal github and is provided without any guarantees or warranties.



HTTP Post Address: http://192.168.0.2:8082

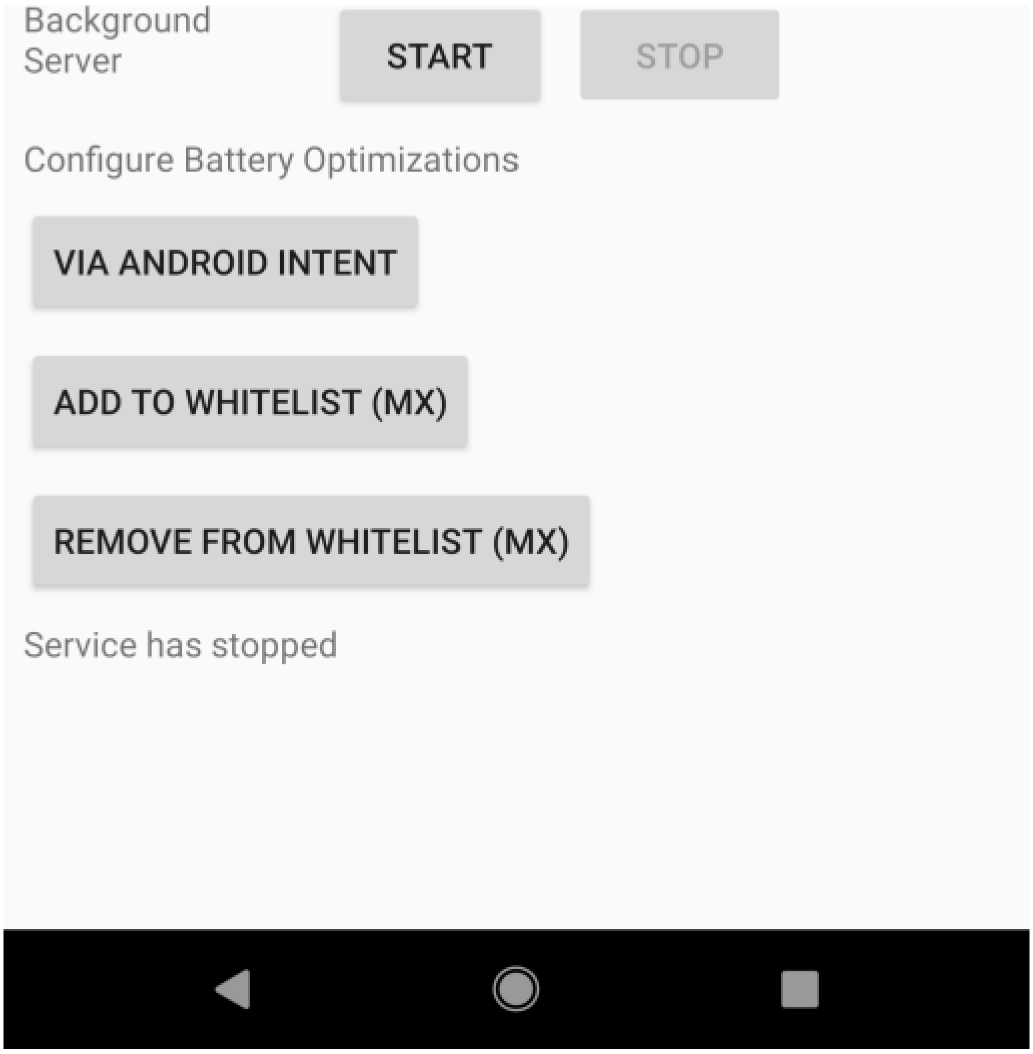
Wake Lock

WiFi Lock

Background Service

Background Service Actions: ☑ Post ☐ Beep

Background Service Frequency: 1 second



Before proceeding, it is worth re-stating the results of testing on an **Android Nougat** device:

Configuration	**Application processing can continue? (HTTP POST)**	**Application can respond to network requests (HTTP GET)**
Wake lock: Not acquired Wifi lock: Not acquired Application whitelisted: No	No. Application will stop processing a few seconds after the screen turns off.	No. Application will stop responding to network requests after a few seconds
Wake lock: Acquired Wifi lock: Not acquired Application whitelisted: No	Until doze. Application will continue processing until doze mode kicks in but will not have WLAN access	No. Application will stop responding to network requests after a few seconds

Configuration	Application processing can continue? (HTTP POST)	Application can respond to network requests (HTTP GET)
Wake lock: Not acquired Wifi lock: Acquired Application whitelisted: No	No. Application will stop processing a few seconds after the screen turns off.	Until doze. Application will respond to network requests until deep doze mode kicks in at which time the application will stop responding to network requests.
Wake lock: Acquired Wifi lock: Acquired Application whitelisted: No	Until doze. Application will continue processing until deep doze mode kicks on.	Until doze. Application will respond to network requests until deep doze mode kicks in at which time the application will stop responding to network requests.
Wake lock: Not acquired Wifi lock: Not acquired Application whitelisted: Yes	No. Application will stop processing a few seconds after the screen turns off.	No. Application will stop responding to network requests after a few seconds
Wake lock: Acquired Wifi lock: Not acquired Application whitelisted: Yes	Yes. Application will continue processing indefinitely but will not have WLAN access	No. Application will stop responding to network requests after a few seconds
Wake lock: Not acquired Wifi lock: Acquired Application whitelisted: Yes	No. Application will stop processing a few seconds after the screen turns off.	Yes. Application will continue responding to network requests indefinitely.

Configuration	Application processing can continue? (HTTP POST)	Application can respond to network requests (HTTP GET)
Wake lock: Acquired Wifi lock: Acquired Application whitelisted: Yes	Yes. Application will continue processing indefinitely.	Yes. Application will continue responding to network requests indefinitely.

Some considerations before re-running the tests under Android P:

- These tests are being run with a **beta version of Android P**, behaviour may change on the release build.
- The tests are being performed on a Google Pixel 2 device, for some reason Google have removed the 'Keep WiFi on during sleep' option (consensus on Reddit seems to be that it is not required). For this reason, any of the tests which involved not acquiring a wake lock cannot be performed, since the WiFi will always remain on when the device sleeps.
- The test application is built to target API 26 (Oreo) with a compileSdkVersion of 28 (P). Minimum SDK version is 21 (Lollipop)
- The P power documentation states, "These [power-management features] apply to all apps, whether or not they target Android P" so by testing on a P platform our target SDK should be inconsequential.
- In relation to App Standby Buckets, introduced in P, the documentation states "Apps that are on the Doze whitelist are exempted from the App Standby Bucket-based restrictions", so we would expect whitelisting to have an effect under P and, when the app is whitelisted, can ignore the earlier documented caveat that bucket assignment can be affected by OEM specific criteria.

Testing results under Android P

Repeating the tests on a Pixel 2 device running the second Android P beta produces the following results:

Configuration	Application processing can continue? (HTTP POST)	Application can respond to network requests (HTTP GET)
Wake lock: Not acquired Wifi lock: Not acquired Application whitelisted: No	Not possible to test – WiFi is always on during sleep	Not possible to test – WiFi is always on during sleep

Configuration	Application processing can continue? (HTTP POST)	Application can respond to network requests (HTTP GET)
Wake lock: Acquired Wifi lock: Not acquired Application whitelisted: No	Not possible to test – WiFi is always on during sleep	Not possible to test – WiFi is always on during sleep
Wake lock: Not acquired Wifi lock: Acquired Application whitelisted: No	No. Application will stop processing a few seconds after the screen turns off (Same as Nougat behaviour).	Server became unresponsive after about 3 minutes. Application is subject to background restrictions which prevent background services (introduced in Oreo)
Wake lock: Acquired Wifi lock: Acquired Application whitelisted: No	No. Application will stop processing about 30 seconds after the screen turns off	Server became unresponsive after about 3 minutes. Application is subject to background restrictions which prevent background services (introduced in Oreo)
Wake lock: Not acquired Wifi lock: Not acquired Application whitelisted: Yes	Not possible to test – WiFi is always on during sleep	Not possible to test – WiFi is always on during sleep
Wake lock: Acquired Wifi lock: Not acquired Application whitelisted: Yes	Not possible to test – WiFi is always on during sleep	Not possible to test – WiFi is always on during sleep

Configuration	Application processing can continue? (HTTP POST)	Application can respond to network requests (HTTP GET)
Wake lock: Not acquired Wifi lock: Acquired Application whitelisted: Yes	No. Application will stop processing about 30 seconds after the screen turns off	Server became unresponsive after about 3 minutes. Presumably related to restrictions on background services
Wake lock: Acquired Wifi lock: Acquired Application whitelisted: Yes	Yes. Application will continue processing indefinitely. (Ran overnight)	Yes. Application will continue processing indefinitely. (Ran overnight)

Conclusions from testing on Android P

- Additional restrictions are observed in some test scenarios, specifically those conditions which under Nougat would run until the device entered doze mode (after about 15 minutes) will now only run for about 3 minutes.
- **It IS still possible to have a background service run continually on an Android P** device with the right combination of wake lock and Doze mode whitelisting. *This is a surprising result* given all the documentation from Google on power saving features in the platform but is encouraging for developers targeting platforms up to and including P who do not want to rework their applications to account for power considerations.

Further tests undertaken on Android P

- **Application bucket**. The test application was augmented to report its current bucket. In all cases where HTTP POST messages were received by the application under test the bucket was reported as STANDBY_BUCKET_ACTIVE. It is not possible to say whether this was specific to the device model under test or an artefact of testing with a beta version of Android P.
    - It is *possible* a less frequently used application would not report as ACTIVE so might give different test results for running in the background than that described above.
- **Effect of battery-saver mode**. Android P documentation states "Android P makes a number of improvements to battery saver mode", the final test scenario with wifi & wake lock acquired as well as the application whitelisted against doze mode was re-run with the device in battery-saver mode. Results showed that battery-saver mode had **NO EFFECT** on whether the application could run in the background under the test conditions.
- **Background restrictions**. The documentation for Android P states under 'Background restrictions' that the user will be notified if a "partial wake lock is held for an hour when the screen is off". I

could not reproduce this and find it a bit confusing since wake locks are cleared when the device enters doze mode.

Share this:

Related

Keeping your Android application running when the device wants to sleep
22nd November 2018
In "Speaking"

Keeping your application running when the device wants to sleep – updated for Android 10
11th October 2019
In "Software"

Keeping your Android application running when the device wants to sleep
5th May 2017
In "Zebra Technologies"

Category *Software*

Tags *android p* *background restrictions* *doze* *doze mode* *wake lock*

6 Comments

Karolis Marksaitis says:

26th February 2019 at 12:48 pm

1. I got wake lock
2. Got wifi lock
3. Whitelisted app over adb

And yet when IDLE mode comes in – all wifi is killed no matter what.

Meizu 15 lite – flymeos 7 – android 7.1.2
All battery optimisations disabled. I give 200 if somebody can solve this for me.

Reply

darryncampbell says:

26th February 2019 at 2:11 pm

If this is Nougat, you might want to try the ‘keep WiFi on during sleep’ setting:
https://android.stackexchange.com/questions/61261/what-does-keep-wifi-on-during-sleep-mean

Reply

asdfasdf says:

27th February 2019 at 2:14 pm

Your tester app worked though! Can't ping phone when deep-idle, but I keep getting POST requests. Do you think same idea can be applied on a VOIP application to make sure it keeps it's connection to server so when somebody calls phone can ring?

Reply

dsfsdfsdf says:

27th February 2019 at 2:44 pm

also not sure if you noticed – but when you minimise your app on the phone, and lock and deep-idle the phone – it looses the wifi lock if you observe over adb. But even if this happens, python dummy service keeps receiving POST requests. Any idea how is that possible?

Reply

darryncampbell says:

28th February 2019 at 9:10 am

You could use this technique to listen for an incoming VOIP call but if I were writing such an app I would probably rely on a high priority Firebase Cloud Message to alert my VOIP app that there is an incoming call – that would likely be a more battery efficient way of achieving the same result. That would also mean you wouldn't have to worry about wake / wifi locks.

Reply

Karolis Marksaitis says:

28th February 2019 at 9:37 am

Sometimes FCM is not an option. Very honestly I did another test with your app. I launched app – did not acquire any locks. Just enabled the service which sends POST. Then put phone in deep-idle – and guess what, your app still keeps sending post messages. I think the http modules you use are excluded from DEEP DOZE, somehow. And it has nothing to do with wifi+wake lock at all.

Reply

Leave a Reply

Your email address will not be published. Required fields are marked *