Android 5.1.1 and above - getRunningAppProcesses() returns my application package only

By Lior Iluz 26.6k1616 gold badges6868 silver badges116116 bronze badges

Published: 2015-06-03 · Archived: 2026-04-05 13:33:21 UTC

It seems Google finally closed all doors for getting the current foreground application package.

After the Lollipop update, which killed `getRunningTasks(int maxNum)` and thanks to this answer, I used this code to get the foreground application package since Lollipop:

```
 final int PROCESS_STATE_TOP = 2;
RunningAppProcessInfo currentInfo = null;
Field field = null;
try {
    field = RunningAppProcessInfo.class.getDeclaredField("processState");
} catch (Exception ignored) {
}
ActivityManager am = (ActivityManager) this.getSystemService(Context.ACTIVITY_SERVICE);
List<RunningAppProcessInfo> appList = am.getRunningAppProcesses();
for (RunningAppProcessInfo app : appList) {
    if (app.importance == RunningAppProcessInfo.IMPORTANCE_FOREGROUND &&
        app.importanceReasonCode == 0 ) {
        Integer state = null;
        try {
            state = field.getInt( app );
        } catch (Exception ignored) {
        }
        if (state != null && state == PROCESS_STATE_TOP) {
            currentInfo = app;
            break;
        }
    }
}
return currentInfo;
```

Android 5.1.1 and above (6.0 Marshmallow), it seems, killed `getRunningAppProcesses()` as well. It now returns a list of your own application package.

UsageStatsManager

We can use the new `UsageStatsManager` API as described here but it doesn't work for all applications. Some system applications will return the same package

```
com.google.android.googlequicksearchbox
```

AccessibilityService (December 2017: Going to be banned for use by Google)

Some applications use `AccessibilityService` (as seen here) but it has some disadvantages.

Is there another way of getting the current running application package?

asked Jun 3, 2015 at 11:52

18

To get a list of running processes on Android 1.6 - Android 6.0 you can use this library I wrote: https://github.com/jaredrummler/AndroidProcesses The library reads /proc to get process info.

Google has significantly restricted access to /proc in Android Nougat. To get a list of running processes on Android Nougat you will need to use UsageStatsManager or have root access.

Click the edit history for previous alternative solutions.

answered Sep 3, 2015 at 3:27

Jared Rummler

38.2k21 gold badges137 silver badges149 bronze badges

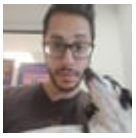
32 Comments

@androiddeveloper no, I only used libsuperuser because it provides an easy way to run a shell command (non-root or root) and get the output. I can re-write it without libsuperuser if there is enough demand.

2015-09-11T14:57:36.97Z+00:00

Sorry for that. Was just being curious. :(

2015-09-12T16:37:21.64Z+00:00



@JaredRummler I'm currently in the process of implementing your solution into my app. Seems to be working fine as far as I can tell

2015-09-21T06:03:08.15Z+00:00

@androiddeveloper, If you want to sort it based on a different attribute, make `Process` implement `Comparable` and override the `compareTo` method. Then, when you use `Collections.sort(processesList)`, it will use the order you specified.

2015-10-11T18:42:39.597Z+00:00

2015-10-21T17:40:30.96Z+00:00

```
private String printForegroundTask() {
  String currentApp = "NULL";
  if(android.os.Build.VERSION.SDK_INT >= android.os.Build.VERSION_CODES.LOLLIPOP) {
      UsageStatsManager usm = (UsageStatsManager)this.getSystemService("usagestats");
      long time = System.currentTimeMillis();
      List<UsageStats> appList = usm.queryUsageStats(UsageStatsManager.INTERVAL_DAILY,  time - 1000*1000, time
      if (appList != null && appList.size() > 0) {
          SortedMap<Long, UsageStats> mySortedMap = new TreeMap<Long, UsageStats>();
          for (UsageStats usageStats : appList) {
              mySortedMap.put(usageStats.getLastTimeUsed(), usageStats);
          }
          if (mySortedMap != null && !mySortedMap.isEmpty()) {
              currentApp = mySortedMap.get(mySortedMap.lastKey()).getPackageName();
          }
```

```
        }
    } else {
        ActivityManager am = (ActivityManager)this.getSystemService(Context.ACTIVITY_SERVICE);
        List<ActivityManager.RunningAppProcessInfo> tasks = am.getRunningAppProcesses();
        currentApp = tasks.get(0).processName;
    }

    Log.e("adapter", "Current App in foreground is: " + currentApp);
    return currentApp;
}
```

Use this method for getting foreground task. U will need an System Permission "android:get_usage_stats"

```
 public static boolean needPermissionForBlocking(Context context){
    try {
        PackageManager packageManager = context.getPackageManager();
        ApplicationInfo applicationInfo = packageManager.getApplicationInfo(context.getPackageName(), 0);
        AppOpsManager appOpsManager = (AppOpsManager) context.getSystemService(Context.APP_OPS_SERVICE);
        int mode = appOpsManager.checkOpNoThrow(AppOpsManager.OPSTR_GET_USAGE_STATS, applicationInfo.uid, applic
        return  (mode != AppOpsManager.MODE_ALLOWED);
    } catch (PackageManager.NameNotFoundException e) {
        return true;
    }
}
```

IF user enable this in setting -> Security-> app with usage access. After that u will get foreground task. Similar process Clean matser by Cheetahamobile [google play link]

answered Sep 1, 2015 at 11:28

Tarun Sharma

8321 gold badge11 silver badges24 bronze badges

11 Comments

As pointed out in the OP and the comments, there are major downsides of using `UsageStatsManager`. Samsung has removed requests and it is quite annoying for a user to grant a system permission.

2015-09-01T11:32:48.363Z+00:00

I have tested in moto e2 device. It is working fine and yes, we need to implement this permission. we all facing same issues. We all really need a better approach. Good luck dude

2015-09-01T12:02:24.72Z+00:00

This approach does not work at least on LG G3 because there is no "Settings -> Security-> App with usage access" menu item

2015-09-01T14:45:21.217Z+00:00

That's not an answer to my question. Moreover, no new info is provided in this answer.

2015-09-05T06:55:14.947Z+00:00

Working fine but not properly in marshmallow. if notification came then current running process will be the package notification received. actually app is not running in foreground or background :(. need solution.

2016-07-01T12:01:01.607Z+00:00

Take a look at https://github.com/ricvalerio/foregroundappchecker, it might be what you need. Provides sample code, and takes away the pain of having to implement cross version foreground detector.

Here are two samples:

```
 AppChecker appChecker = new AppChecker();
String packageName = appChecker.getForegroundApp();
```

Or regularly check:

```
 AppChecker appChecker = new AppChecker();
appChecker
    .when("com.other.app", new AppChecker.Listener() {
        @Override
        public void onForeground(String packageName) {
            // do something
        }
    )
    .when("com.my.app", new AppChecker.Listener() {
        @Override
        public void onForeground(String packageName) {
            // do something
        }
    )
    .other(new AppChecker.Listener() {
        @Override
        public void onForeground(String packageName) {
            // do something
        }
    )
    .timeout(1000)
    .start(this);
```

answered Aug 28, 2016 at 23:06

rvalerio

4975 silver badges5 bronze badges

2 Comments

Thanks but nothing's new here. He just wrapped UsageStatsManager API which is mentioned in OP. The user will need to enable access, as other methods since Android 5.1.1

2016-08-29T04:52:17.22Z+00:00

@Jérémy did you request the required permissions?

2019-01-13T19:26:35.827Z+00:00

Google limited this functionality for system apps only. As been reported in a bug ticket, you will need the **REAL_GET_TASKS** permission to access there.

> Applications must now have ...permission.REAL_GET_TASKS to be able to get process information for all applications. Only the process information for the calling application will be returned if the app doesn't have the permission. Privileges apps will temporarily be able to get process information for all applications if they don't have the new permission, but have deprecated ...permission.GET_TASKS Also,only system apps can acquire the REAL_GET_TASKS permission.

answered Sep 2, 2015 at 15:09

Ilya Gazman

32.6k25 gold badges152 silver badges239 bronze badges

2 Comments

I saw applock is still working on 5.1.1, once the app get the permission in security->"apps with usage access"... This is somewhat surprising

2015-09-22T01:06:49.647Z+00:00

"Permissions with the protection level signature, privileged or signatureOrSystem are only granted to system apps. If an app is a regular non-system app, it will never be able to use these permissions. " say android studio.. Good luck to negociate with Google to be accepted as a "system app".

2019-01-12T18:55:25.35Z+00:00

Just throwing out a potential optimization to what I imagine is a heavily copy-pasted bit of code for detecting the top-most application on Android M.

This

```
 if (android.os.Build.VERSION.SDK_INT >= android.os.Build.VERSION_CODES.LOLLIPOP) {
    UsageStatsManager usm = (UsageStatsManager)this.getSystemService("usagestats");
    long time = System.currentTimeMillis();
    List<UsageStats> appList = usm.queryUsageStats(UsageStatsManager.INTERVAL_DAILY,  time - 1000*1000, time);
    if (appList != null && appList.size() > 0) {
        SortedMap<Long, UsageStats> mySortedMap = new TreeMap<Long, UsageStats>();
        for (UsageStats usageStats : appList) {
            mySortedMap.put(usageStats.getLastTimeUsed(), usageStats);
        }
        if (mySortedMap != null && !mySortedMap.isEmpty()) {
            currentApp = mySortedMap.get(mySortedMap.lastKey()).getPackageName();
        }
    }
}
```

Can be simplified to this

```
 if (android.os.Build.VERSION.SDK_INT >= android.os.Build.VERSION_CODES.LOLLIPOP) {
    UsageStatsManager usm = (UsageStatsManager) context.getSystemService(
        Context.USAGE_STATS_SERVICE);
    long time = System.currentTimeMillis();
    List<UsageStats> appStatsList = usm.queryUsageStats(UsageStatsManager.INTERVAL_DAILY,
            time - 1000 * 1000, time);
    if (appStatsList != null && !appStatsList.isEmpty()) {
        currentApp = Collections.max(appStatsList, (o1, o2) ->
            Long.compare(o1.getLastTimeUsed(), o2.getLastTimeUsed())).getPackageName();
    }
}
```

I found myself using this code in a 2 second loop, and wondered why I was using a complex solution that was O(n*log(n)) when a more simple solution was available in Collections.max() which is O(n).

answered Dec 1, 2017 at 1:07

bstar55

3,6643 gold badges23 silver badges25 bronze badges

Comments

```
 public class AccessibilityDetectingService extends AccessibilityService {

@Override
```

```
protected void onServiceConnected() {
    super.onServiceConnected();

    //Configure these here for compatibility with API 13 and below.

    AccessibilityServiceInfo config = new AccessibilityServiceInfo();
    config.eventTypes = AccessibilityEvent.TYPE_WINDOW_STATE_CHANGED;
    config.feedbackType = AccessibilityServiceInfo.FEEDBACK_GENERIC;

    if (Build.VERSION.SDK_INT >= 16)
        //Just in case this helps
        config.flags = AccessibilityServiceInfo.FLAG_INCLUDE_NOT_IMPORTANT_VIEWS;

    setServiceInfo(config);
}

@Override
public void onAccessibilityEvent(final AccessibilityEvent event) {
        if (event == null ) {
            return;
        } else if(event.getPackageName() == null && event.getClassName() == null){
            return;
        }

            if (activityInfo != null){

                Log.d("CurrentActivity", componentName.flattenToShortString());
        }

}

private ActivityInfo tryGetActivity(ComponentName componentName) {
    try {
        return getPackageManager().getActivityInfo(componentName, 0);
    } catch (PackageManager.NameNotFoundException e) {
        return null;
    }
}
@Override
public void onInterrupt() {
}
}
}//`enter code here`uses-permission android:name="android.permission.BIND_ACCESSIBILITY_SERVICE" />
<uses-permission android:name="android.permission.GET_TASKS" />
```

Then start the service and app accessibility on in your device setting->accessibility->App on that service.

answered Feb 2, 2016 at 11:22

1 Comment

This is already covered in the question, and you've just copy-pasted the code from its original source on StackOverflow.

2019-02-17T10:16:05.827Z+00:00

Please try to use `getRunningServices()` instead of `getRunningAppProcesses()` method.

```
 ActivityManager mActivityManager = (ActivityManager) getSy stemService(Context.ACTIVITY_SERVICE);

List<ActivityManager.RunningServiceInfo> appProcessInfoList = mActivityManager.getRunningServices(Integer.MAX_\
```

soumya

3,8019 gold badges39 silver badges70 bronze badges

answered Oct 9, 2015 at 9:23

1 Comment

Welcome to Stack Overflow! I don't think this will work since the foreground app might not be running a service.

2015-10-23T21:05:31.85Z+00:00

Start asking to get answers

Find the answer to your question by asking.

Ask question

Explore related questions

See similar questions with these tags.
