

← ↻ ☆

Fix flaking postsubmit test:
androidx.appcompat.app.NightModeRotateDoesNotRecreateActivityTestCase

+1²

Hotlists (2)

Mark as Duplicate

Comments (35)DependenciesDuplicates (3)BlockingResources (7)

Fixed

Bug

P2

+ Add Hotlist

STATUS UPDATE No update yet.

Edit

DESCRIPTION ti...@google.com created issue on behalf of sj...@google.com #1

* Originally From: sjgilbert@google.com *

androidx.appcompat.app.NightModeRotateDoesNotRecreateActivityTestCase has been flaking on products:
API 28: Pixel 2016

Here's the definition of 'flake' in this context: Consider the last 30 test runs broken into 3 intervals of 10. It is a flaky test if the it displays a <fail, pass, fail> pattern across these intervals.An interval is considered a 'fail' if a single test run within the interval has a failing status. Similarly, an interval is considered a 'pass' if the test passes at least once in that interval.

The flaky bug filing bot runs daily, and will update this bug if it continues to meet the criteria. Failure to meet this criteria doesn't mean the bug is fixed. If you already have an open bug for this, please duplicate this bug against it to prevent this bot from filing another.

You can view the failures by visiting go/androidx-tests, opening the hamburger menu on the top left, selecting the appropriate product, locating your tests, and following the link to the failures. To locate the last good/first breaking build, you can either scroll to the right until you find the first green build, or you can use the tool at go/find-a-break-androidx.

[priority: P1]
[hotlists: 2230173]

✓ Mentioned issues (1)

✓ Links (6)

Mentioned issues (1)

-- --

"I think [b/155379937](#) exposed the root cause -- if there are entries in sActiveDelegates with duplicate Context, Resources, or Configuration objects, then we may end up flipping the nig

Links (6)

"<https://android-review.googlesource.com/1299229>"

"<https://goto.google.com/android-sha1/cd849d2fb6de91cff24523889...>"

"Looking at a [recent instance](#) of the flake on SDK 28:"

"<https://android-review.googlesource.com/1308513>"

"On SC: <https://android-build.googleplex.com/builds/tests/view?invocationId=18300006486129606&testResultId=TR...>"

"<https://android-review.googlesource.com/1530721>"

COMMENTS

ti...@google.com <ti...@google.com> #2

* Originally From: sjgilbert@google.com *

androidx.appcompat.app.NightModeRotateDoesNotRecreateActivityTestCase has been flaking on products:
API 28: Pixel 2016

Here's the definition of 'flake' in this context: Consider the last 30 test runs broken into 3 intervals of 10. It is a flaky test if the it displays a <fail, pass, fail> pattern across these intervals.An interval is considered a 'fail' if a single test run within the interval has a failing status. Similarly, an interval is considered a 'pass' if the test passes at least once in that interval.

The flaky bug filing bot runs daily, and will update this bug if it continues to meet the criteria. Failure to meet this criteria doesn't mean

the bug is fixed. If you already have an open bug for this, please duplicate this bug against it to prevent this bot from filing another.

You can view the failures by visiting [go/androidx-tests](#), opening the hamburger menu on the top left, selecting the appropriate product, locating your tests, and following the link to the failures. To locate the last good/first breaking build, you can either scroll to the right until you find the first green build, or you can use the tool at [go/find-a-break-androidx](#).

[priority: P1]
[hotlists: 2230173]

ki...@google.com <ki...@google.com> [#3](#)

Reassigned to al...@google.com.

Alan - is this yours or Chris's?

ki...@google.com <ki...@google.com> [#4](#)

```
java.lang.AssertionError: Expected DESTROYED never happened to androidx.appcompat.app.NightModeRotateDoesNotRecreateActivity@5a75c5a. Current state:RESUMED
Expected: is <true>
but: was <false>
    at org.hamcrest.MatcherAssert.assertThat(MatcherAssert.java:20)
    at androidx.testutils.LifecycleOwnerUtils.waitForRecreation(LifecycleOwnerUtils.java:135)
    at androidx.appcompat.testutils.NightModeUtils.setNightModeAndWaitForRecreate(NightModeUtils.kt:114)
    at androidx.appcompat.testutils.NightModeUtils.setNightModeAndWaitForRecreate(NightModeUtils.kt:97)
    at androidx.appcompat.app.NightModeRotateDoesNotRecreateActivityTestCase.testRotateDoesNotRecreateActivity(NightModeRotateDoesNotRecreateActivityTestCase.kt:57)
    at java.lang.reflect.Method.invoke(Native Method)
    at org.junit.runners.model.FrameworkMethod$1.runReflectiveCall(FrameworkMethod.java:50)
    at org.junit.internal.runners.model.ReflectiveCallable.run(ReflectiveCallable.java:12)
    at org.junit.runners.model.FrameworkMethod.invokeExplosively(FrameworkMethod.java:47)
```

al...@google.com <al...@google.com> [#5](#)

Hard to tell! I added the `DESTROYED` check as a canary -- it probably indicates something is wrong with the device or adjacent tests. Previously that was the dialogs.

It's my test and my check, so I'll try to look into this when I have a free hour. Likely Thursday.

Chris or Kirill -- if you happen to have time before then and would like to help diagnose, I'd appreciate it.

ti...@google.com <ti...@google.com> [#6](#)

* Originally From: sjgilbert@google.com *

androidx.appcompat.app.NightModeRotateDoesNotRecreateActivityTestCase has been flaking on products:
API 28: Pixel 2016

Here's the definition of 'flake' in this context: Consider the last 30 test runs broken into 3 intervals of 10. It is a flaky test if the it displays a <fail, pass, fail> pattern across these intervals. An interval is considered a 'fail' if a single test run within the interval has a failing status. Similarly, an interval is considered a 'pass' if the test passes at least once in that interval.

The flaky bug filing bot runs daily, and will update this bug if it continues to meet the criteria. Failure to meet this criteria doesn't mean the bug is fixed. If you already have an open bug for this, please duplicate this bug against it to prevent this bot from filing another.

You can view the failures by visiting [go/androidx-tests](#), opening the hamburger menu on the top left, selecting the appropriate product, locating your tests, and following the link to the failures. To locate the last good/first breaking build, you can either scroll to the right until you find the first green build, or you can use the tool at [go/find-a-break-androidx](#).

[priority: P1]
[hotlists: 2230173]

ti...@google.com <ti...@google.com> [#7](#)

* Originally From: sjgilbert@google.com *

androidx.appcompat.app.NightModeRotateDoesNotRecreateActivityTestCase has been flaking on products:
API 28: Pixel 2016

Here's the definition of 'flake' in this context: Consider the last 30 test

runs broken into 3 intervals of 10. It is a flaky test if the it displays a <fail, pass, fail> pattern across these intervals. An interval is considered a 'fail' if a single test run within the interval has a failing status. Similarly, an interval is considered a 'pass' if the test passes at least once in that interval.

The flaky bug filing bot runs daily, and will update this bug if it continues to meet the criteria. Failure to meet this criteria doesn't mean the bug is fixed. If you already have an open bug for this, please duplicate this bug against it to prevent this bot from filing another.

You can view the failures by visiting go/androidx-tests, opening the hamburger menu on the top left, selecting the appropriate product, locating your tests, and following the link to the failures. To locate the last good/first breaking build, you can either scroll to the right until you find the first green build, or you can use the tool at go/find-a-break-androidx.

[priority: P1]
[hotlists: 2230173]

ti...@google.com <ti...@google.com> [#8](#)

* Originally From: sgilbert@google.com *

androidx.appcompat.app.NightModeRotateDoesNotRecreateActivityTestCase has been flaking on products:
API 28: Pixel 2016

Here's the definition of 'flake' in this context: Consider the last 30 test runs broken into 3 intervals of 10. It is a flaky test if the it displays a <fail, pass, fail> pattern across these intervals. An interval is considered a 'fail' if a single test run within the interval has a failing status. Similarly, an interval is considered a 'pass' if the test passes at least once in that interval.

The flaky bug filing bot runs daily, and will update this bug if it continues to meet the criteria. Failure to meet this criteria doesn't mean the bug is fixed. If you already have an open bug for this, please duplicate this bug against it to prevent this bot from filing another.

You can view the failures by visiting go/androidx-tests, opening the hamburger menu on the top left, selecting the appropriate product, locating your tests, and following the link to the failures. To locate the last good/first breaking build, you can either scroll to the right until you find the first green build, or you can use the tool at go/find-a-break-androidx.

[priority: P1]
[hotlists: 2230173]

al...@google.com <al...@google.com> [#9](#)

Accepted by al...@google.com.

```
java.lang.AssertionError: Expected DESTROYED never happened to androidx.appcompat.app.NightModeRotateDoesNotRecreateActivity@f362459. Current state:RESUME
Expected: is <true>
but: was <false>
    at org.hamcrest.MatcherAssert.assertThat(MatcherAssert.java:20)
    at androidx.testutils.LifecycleOwnerUtils.waitForRecreation(LifecycleOwnerUtils.java:135)
    at androidx.appcompat.testutils.NightModeUtils.setNightModeAndWaitForRecreate(NightModeUtils.kt:115)
    at androidx.appcompat.testutils.NightModeUtils.setNightModeAndWaitForRecreate(NightModeUtils.kt:98)
    at androidx.appcompat.app.NightModeRotateDoesNotRecreateActivityTestCase.testRotateDoesNotRecreateActivity(NightModeRotateDoesNotRecreateActivityT
    at java.lang.reflect.Method.invoke(Native Method)
```

Failure seems to co-occur with some other tests.

Occasionally:

- androidx.appcompat.app.NightModeDefaultOnlyTestCase
- androidx.appcompat.app.NightModePreventOverrideConfigTestCase (only the [0] config)
- androidx.appcompat.app.NightModeTestCase (only the [0] config)

Always:

- androidx.appcompat.app.NightModeRotateDoesNotRecreateActivityTestCase (only the [0] config)
- androidx.appcompat.app.NightModeRotateRecreatesActivityWithConfigTestCase (only the [0] config)

al...@google.com <al...@google.com> [#10](#)

Normal execution. Note how the lifecycle event occurs immediately (~4ms) after the mode change:

```
01-01 00:10:37.978 10801 10816 D NightModeUtils: setNightModeAndWaitForRecreate on Activity: androidx.appcompat.app.NightModeCustomAttachBaseContextActivi
01-01 00:10:37.978 671 671 D QCOM PowerHAL: LAUNCH HINT: OFF
01-01 00:10:37.982 10801 10801 D LifecycleMonitor: Lifecycle status change: androidx.appcompat.app.NightModeCustomAttachBaseContextActivity@ddad2e9 in: PA
01-01 00:10:37.982 10801 10801 D LifecycleMonitor: running callback: androidx.test.rule.ActivityTestRule$LifecycleCallback@2698764
01-01 00:10:37.982 10801 10801 D LifecycleMonitor: callback completes: androidx.test.rule.ActivityTestRule$LifecycleCallback@2698764
01-01 00:10:37.983 10801 10801 D LifecycleMonitor: Lifecycle status change: androidx.appcompat.app.NightModeCustomAttachBaseContextActivity@ddad2e9 in: SI
01-01 00:10:37.984 10801 10801 D LifecycleMonitor: running callback: androidx.test.rule.ActivityTestRule$LifecycleCallback@2698764
01-01 00:10:37.984 10801 10801 D LifecycleMonitor: callback completes: androidx.test.rule.ActivityTestRule$LifecycleCallback@2698764
01-01 00:10:37.985 10801 10801 D LifecycleMonitor: Lifecycle status change: androidx.appcompat.app.NightModeCustomAttachBaseContextActivity@ddad2e9 in: DE
01-01 00:10:37.985 10801 10801 D LifecycleMonitor: running callback: androidx.test.rule.ActivityTestRule$LifecycleCallback@2698764
01-01 00:10:37.985 10801 10801 D LifecycleMonitor: callback completes: androidx.test.rule.ActivityTestRule$LifecycleCallback@2698764
```

Failed executions.

```
01-01 00:10:43.385 10801 10816 D NightModeUtils: setNightModeAndWaitForRecreate on Activity: androidx.appcompat.app.NightModeRotateDoesNotRecreateActivity
01-01 00:10:43.394 653 933 W DeviceHAL: Error from HAL Device in function open_input_stream: Invalid argument
01-01 00:10:43.395 2262 2262 I MicroDetector: Keeping mic open: false
01-01 00:10:43.396 2262 4273 I MicroRecognitionRunner: Stopping hotword detection.
01-01 00:10:43.400 2262 10959 I AudioController: internalShutdown
01-01 00:10:43.400 494 1880 W SurfaceFlinger: Attempting to set client state on removed layer: Splash Screen androidx.appcompat.test#0
01-01 00:10:43.400 494 1880 W SurfaceFlinger: Attempting to destroy on removed layer: Splash Screen androidx.appcompat.test#0
01-01 00:10:43.400 2262 10965 I DeviceStateChecker: DeviceStateChecker cancelled
01-01 00:10:43.404 893 12038 I AudioFlinger: AudioFlinger's thread 0x1c09b00 tid=12038 ready to run
01-01 00:10:43.420 893 1200 I SoundTriggerHwService::Module: onCallbackEvent no clients
01-01 00:10:43.422 2262 10958 I MicrophoneInputStream: mic_started SR : 16000 CC : 16 SO : 1999
```

```
01-01 00:11:00.084 10801 10816 D NightModeUtils: setNightModeAndWaitForRecreate on Activity: androidx.appcompat.app.NightModeActivity@9ec425 to mode: 2 us
01-01 00:11:00.085 2190 3418 I PlaceInferenceEngine: [anon] Changed inference mode: 1
01-01 00:11:00.087 2190 12056 I Places : ?: Couldn't find platform key file.
01-01 00:11:00.088 2190 11669 I Places : ?: Couldn't find platform key file.
01-01 00:11:00.092 2262 10965 E PlaceStateUpdater: Received no places
01-01 00:11:00.094 2190 11552 I Places : ?: Couldn't find platform key file.
01-01 00:11:00.094 671 671 D QCOM PowerHAL: LAUNCH HINT: OFF
01-01 00:11:00.100 494 537 W SurfaceFlinger: Attempting to destroy on removed layer: dadd87d AssistPreviewPanel#0
01-01 00:11:00.101 2190 3418 I PlaceInferenceEngine: Too few candidate results: percentageOfMissingArea 100
01-01 00:11:00.103 2262 2262 I MicroDetectionWorker: onReady
01-01 00:11:00.103 909 1185 I WindowManager: Screen frozen for +575ms due to Window{a52960d u0 StatusBar}
```

Nothing obviously similar between the two failures. Screenshots look normal-ish and show the "before" activity state, e.g. the failures for `NightModeTestCase` are day mode.

ap...@google.com <ap...@google.com> [#11](#)

Project: platform/frameworks/support
Branch: androidx-master-dev

commit cd849d2fb6de91cff245238898ade23678c73fb5
Author: Alan Viverette <alanv@google.com>
Date: Thu Apr 30 12:28:58 2020

Enable verbose debugging for AppCompatActivity night mode

Hopefully we can catch the flake.

Bug: 152673431
Test: androidx.appcompat.app.NightModeRotateDoesNotRecreateActivityTestCase
Change-Id: I879f238f3fe5e881d9c896548e08f10e3674ca56

M appcompat/appcompat/src/main/java/androidx/appcompat/app/AppCompatDelegate.java
M appcompat/appcompat/src/main/java/androidx/appcompat/app/AppCompatDelegateImpl.java

<https://android-review.googlesource.com/1299229>
<https://goto.google.com/android-sha1/cd849d2fb6de91cff245238898ade23678c73fb5>

al...@google.com <al...@google.com> [#12](#)

I think b/155379937 exposed the root cause -- if there are entries in `sActiveDelegates` with duplicate `Context`, `Resources`, or `Configuration` objects, then we may end up flipping the ni
Working hypothesis is that we have two entries representing two different `Activity` objects with the same `Configuration`. We may or may not attempt to recreate the first (dead) activity, l

al...@google.com <al...@google.com> [#13](#)

Tentatively, we could work around in a couple of ways:

1. Freeze the initial delegate configuration state before traversing `sActiveDelegates`
2. Always re-apply `uiMode` and `recreate()` even if the mode is already set correctly
3. Throw an exception if a new delegate shares a `Configuration` with an existing delegate

I think freezing the state would be the most reliable option. Shared configurations seem WAI on certain API levels and re-applying doesn't seem consistent with the platform behavior.

This is still all theoretical -- I haven't gotten a stable repro or any verbose debug logs from this flake.

ch...@google.com <ch...@google.com> [#14](#)

Agreed, #1 seems the most logical to me.

al...@google.com <al...@google.com> [#15](#)

Looking at a [recent instance](#) of the flake on SDK 28:

```
01-01 00:28:07.343 16481 16481 D AppCompatDelegate: setDefaultNightMode. New:1, Current:2
01-01 00:28:07.343 16481 16481 D AppCompatDelegate: applyDayNightToActiveDelegates. Applying to androidx.appcompat.app.AppCompatActivityImpl@a9f5249
01-01 00:28:07.344 16481 16481 D AppCompatDelegate: updateForNightMode [allowRecreation:true, currentNightMode:32, newNightMode:16, activityHandlingUiMode:1]
01-01 00:28:07.344 16481 16481 D AppCompatDelegate: updateForNightMode not recreating Activity: androidx.appcompat.app.AlertDialog@9e60f79
01-01 00:28:07.344 16481 16481 D AppCompatDelegate: updateForNightMode. Updating resources config on host: androidx.appcompat.app.AlertDialog@9e60f79
01-01 00:28:07.349 16481 16481 D AppCompatDelegate: applyDayNightToActiveDelegates. Applying to androidx.appcompat.app.AppCompatActivityImpl@ae6c06f
01-01 00:28:07.350 16481 16481 D AppCompatDelegate: updateForNightMode [allowRecreation:true, currentNightMode:16, newNightMode:16, activityHandlingUiMode:1]
01-01 00:28:07.350 16481 16481 D AppCompatDelegate: updateForNightMode not recreating Activity: androidx.appcompat.app.AppCompatActivity@1d80087
01-01 00:28:07.350 16481 16481 D AppCompatDelegate: updateForNightMode. Skipping. Night mode: 1 for host:androidx.appcompat.app.AppCompatActivity@1d80087
01-01 00:28:07.350 16481 16481 D AppCompatDelegate: applyDayNightToActiveDelegates. Applying to androidx.appcompat.app.AppCompatActivityImpl@7bbf17c
01-01 00:28:07.351 16481 16481 D AppCompatDelegate: updateForNightMode [allowRecreation:true, currentNightMode:16, newNightMode:16, activityHandlingUiMode:1]
01-01 00:28:07.351 16481 16481 D AppCompatDelegate: updateForNightMode not recreating Activity: androidx.appcompat.app.AlertDialog@464e0fd
01-01 00:28:07.351 16481 16481 D AppCompatDelegate: updateForNightMode. Skipping. Night mode: 1 for host:androidx.appcompat.app.AlertDialog@464e0fd
01-01 00:28:07.353 16481 16497 E TestRunner: failed: testNightModeChangeWhenInBackground(androidx.appcompat.app.NightModeDefaultOnlyTestCase)
01-01 00:28:07.353 16481 16497 E TestRunner: ----- begin exception -----
01-01 00:28:07.355 16481 16497 E TestRunner: java.lang.AssertionError: Expected DESTROYED never happened to androidx.appcompat.app.AppCompatActivity@1d80087
```

AlertDialog gets the configuration change first, which prevents the host activity from running recreate(). Bingo!

al...@google.com <al...@google.com> [#16](#)

Still unable to submit.

```
07-24 14:55:43.466 11170 11192 I TestRunner: started: testRotateDoesNotRecreateActivity[1](androidx.appcompat.app.NightModeRotateDoesNotRecreateActivityTest)
... (app starts) ...
07-24 14:55:43.991 11170 11170 D LifecycleMonitor: Lifecycle status change: androidx.appcompat.app.NightModeRotateDoesNotRecreateActivity@8b32b5d in: RESUMING
... (what appears to be an orientation change) ...
07-24 14:55:44.341 522 542 V WindowManager: Orientation start waiting for draw, mDrawState=DRAW_PENDING in Window{3303fcd u0 com.android.systemui.ImageBackground}
07-24 14:55:44.443 6821 6821 W GoogleInputMethodService: GoogleInputMethodService.onConfigurationChanged():1401 onConfigurationChanged() : NewConfig = {
... (a LOT of spam while IME handles config change) ...
07-24 14:55:44.628 522 547 I ActivityTaskManager: Displayed androidx.appcompat.test/androidx.appcompat.app.NightModeRotateDoesNotRecreateActivity: +97ms
07-24 14:55:44.631 522 542 D EventSequenceValidator: Transition from ACTIVITY_LAUNCHED to ACTIVITY_FINISHED
07-24 14:55:44.634 390 390 W SurfaceFlinger: couldn't log to binary event log: overflow.
```

And then we run the test.

```
07-24 14:55:44.636 11170 11192 D NightModeUtils: setNightModeAndWaitForRecreate on Activity: androidx.appcompat.app.NightModeRotateDoesNotRecreateActivityTest
07-24 14:55:44.638 390 576 I BpBinder: onLastStrongRef automatically unlinking death recipients: <uncached descriptor>
07-24 14:55:44.749 1725 1725 W MonetClient: Could not collect restore state: The DisplayCoordinator's controller has not taken over yet.
07-24 14:55:44.751 1725 2298 I AppFlowTracker: AppFlow[Type: 236 - Status: 1 - Events[1673 at 0ms, 1675 at -585ms, 1665 at -608ms, 1668 at -611ms, 1671 at -614ms]]
07-24 14:55:44.753 11170 11170 D LifecycleMonitor: Lifecycle status change: androidx.appcompat.app.NightModeRotateDoesNotRecreateActivity@8b32b5d in: PAUSED
07-24 14:55:44.753 11170 11170 D LifecycleMonitor: running callback: androidx.test.rule.ActivityTestRule$LifecycleCallback@a8aa6e8
07-24 14:55:44.753 11170 11170 D LifecycleMonitor: callback completes: androidx.test.rule.ActivityTestRule$LifecycleCallback@a8aa6e8
07-24 14:55:44.760 11170 11192 E TestRunner: failed: testRotateDoesNotRecreateActivity[1](androidx.appcompat.app.NightModeRotateDoesNotRecreateActivityTest)
07-24 14:55:44.760 11170 11192 E TestRunner: ----- begin exception -----
07-24 14:55:44.761 11170 11192 E TestRunner: java.lang.AssertionError
07-24 14:55:44.761 11170 11192 E TestRunner: at org.junit.Assert.fail(Assert.java:86)
07-24 14:55:44.761 11170 11192 E TestRunner: at org.junit.Assert.assertTrue(Assert.java:41)
07-24 14:55:44.761 11170 11192 E TestRunner: at org.junit.Assert.assertTrue(Assert.java:52)
07-24 14:55:44.761 11170 11192 E TestRunner: at androidx.appcompat.testutils.NightModeUtils.setNightModeAndWaitForRecreate(NightModeUtils.kt:115)
07-24 14:55:44.761 11170 11192 E TestRunner: at androidx.appcompat.app.NightModeRotateDoesNotRecreateActivityTest.testRotateDoesNotRecreateActivity()
```

al...@google.com <al...@google.com> [#17](#)

My best guess right now is that the IME is taking up system resources, so the app starts up and meets the criteria for LifecycleOwnerUtils.waitForState(activity, Lifecycle.State.RESUMING). The failure screenshot is empty, so there's a decent chance the app never gets displayed. Seems like a recurring theme in flaky tests.

al...@google.com <al...@google.com> [#18](#)

I'm going try waiting until the window has focus, rather than checking and immediately aborting.

ap...@google.com <ap...@google.com> [#19](#)

Project: platform/frameworks/support
Branch: androidx-master-dev

commit 8e9ee0661573d9eff50b99b43af92252c09f478e
Author: Alan Viverette <alanv@google.com>
Date: Mon May 11 12:47:01 2020

Fix NightModeRotateDoesNotRecreateActivity failure on emulator

Second parameterized run was catching the old activity, so now we restart the activity and are more careful to clear the default night mode on failure. Also added some sanity checks when we're changing night mode.

Bug: 152673431
Test: NightModeRotateDoesNotRecreateActivity
Change-Id: Icd4bac8473aec243bd9912690b7f1f715fc1bc11

M appcompat/appcompat/src/androidTest/java/androidx/appcompat/app/NightModeRotateDoesNotRecreateActivityTestCase.kt
M appcompat/appcompat/src/androidTest/java/androidx/appcompat/testutils/NightModeUtils.kt

<https://android-review.googlesource.com/1308513>

al...@google.com <al...@google.com>

Marked as fixed.

sj...@google.com <sj...@google.com> [#20](#)

Status: Assigned (reopened)

Still appears to be flaking.

sj...@google.com <sj...@google.com> [#21](#)

On SC: <https://android-build.googleplex.com/builds/tests/view?invocationId=118300006486129606&testResultId=TR33914834901748643>

al...@google.com <al...@google.com> [#22](#)

```
java.lang.AssertionError: Expected RESUMED never happened to androidx.appcompat.app.NightModeActivity@89bccbf. Current state:DESTROYED
Expected: is <true>
but: was <false>
    at org.hamcrest.MatcherAssert.assertThat(MatcherAssert.java:20)
    at androidx.testutils.LifecycleOwnerUtils.waitForState(LifecycleOwnerUtils.java:82)
    at androidx.appcompat.testutils.NightModeUtils.setNightModeAndWaitForRecreate(NightModeUtils.kt:114)
    at androidx.appcompat.testutils.NightModeUtils.setNightModeAndWaitForRecreate(NightModeUtils.kt:100)
    at androidx.appcompat.app.NightModeRotateRecreatesActivityWithConfigTestCase.testRotateRecreatesActivityWithConfig(NightModeRotateRecreatesActivit
    at java.lang.reflect.Method.invoke(Native Method)
```

From logcat, some background service is failing over and over:

```
11-08 11:54:37.663 544 564 E IorapForwardingService: Transaction failed on small parcel; remote process probably died
11-08 11:54:37.663 544 564 E IorapForwardingService: android.os.DeathObjectException: Transaction failed on small parcel; remote process probably died
11-08 11:54:37.663 544 564 E IorapForwardingService: at android.os.BinderProxy.transactNative(Native Method)
11-08 11:54:37.663 544 564 E IorapForwardingService: at android.os.BinderProxy.transact(BinderProxy.java:550)
11-08 11:54:37.663 544 564 E IorapForwardingService: at com.google.android.startup.iorap.Iorap$Stub$Proxy.onAppLaunchEvent(Iorap.java:427)
11-08 11:54:37.663 544 564 E IorapForwardingService: at com.google.android.startup.iorap.IorapForwardingService$AppLaunchObserver.lambda$onActiv
11-08 11:54:37.663 544 564 E IorapForwardingService: at com.google.android.startup.iorap.~$$Lambda$IorapForwardingService$AppLaunchObserver$0L4ql
11-08 11:54:37.663 544 564 E IorapForwardingService: at com.google.android.startup.iorap.IorapForwardingService.invokeRemote(IorapForwardingServ
11-08 11:54:37.663 544 564 E IorapForwardingService: at com.google.android.startup.iorap.IorapForwardingService.access$500(IorapForwardingService
```

But near the test failure there's just:

```
11-08 11:58:22.043 4965 4987 D NightModeUtils: setNightModeAndWaitForRecreate on Activity: androidx.appcompat.app.NightModeActivity@89bccbf to mode: 2 u
11-08 11:58:22.925 958 986 E MmTelFeatureConn [0]: ImsRegistrationCallbackAdapter: ImsRegistration is null
11-08 11:58:27.983 958 986 E MmTelFeatureConn [0]: ImsRegistrationCallbackAdapter: ImsRegistration is null
11-08 11:58:32.655 544 5976 W TelephonyManager: requestModemActivityInfo: Received an invalid ModemActivityInfo
11-08 11:58:32.655 544 5976 W BatteryExternalStatsWorker: error reading modem stats:ERROR_INVALID_INFO_RECEIVED
11-08 11:58:33.042 958 986 E MmTelFeatureConn [0]: ImsRegistrationCallbackAdapter: ImsRegistration is null
11-08 11:58:37.061 4965 4965 D LifecycleMonitor: Lifecycle status change: androidx.appcompat.app.NightModeActivity@dae00f3 in: PAUSED
11-08 11:58:37.061 4965 4965 D LifecycleMonitor: running callback: androidx.test.rule.ActivityTestRule$LifecycleCallback@658e5a4
11-08 11:58:37.061 4965 4965 D LifecycleMonitor: callback completes: androidx.test.rule.ActivityTestRule$LifecycleCallback@658e5a4
11-08 11:58:37.066 4965 4987 E TestRunner: failed: testRotateRecreatesActivityWithConfig[0] (androidx.appcompat.app.NightModeRotateRecreatesActivityWithC
11-08 11:58:37.066 4965 4987 E TestRunner: ----- begin exception -----
11-08 11:58:37.067 4965 4987 E TestRunner: java.lang.AssertionError: Expected RESUMED never happened to androidx.appcompat.app.NightModeActivity@89bccbf
```

```
11-08 11:58:37.067 4965 4987 E TestRunner: Expected: is <true>
11-08 11:58:37.067 4965 4987 E TestRunner:      but: was <false>
```

So we request the change and then nothing happens for 15 seconds, then it fails.

al...@google.com <al...@google.com> [#23](#)

Ah, except that window ID is from way earlier where it seems to hit a Window Session Crash on... possibly some other window? And then there are a ton of "resource failed to call release" me

```
11-08 11:58:16.669 4965 4965 D LifecycleMonitor: Lifecycle status change: androidx.appcompat.app.NightModeActivity@89bccbf in: RESUMED
11-08 11:58:16.824 544 5575 W InputManager-JNI: Input channel object '7f829bd com.google.android.apps.nexuslauncher/com.google.android.apps.nexuslaunch
11-08 11:58:16.824 544 5575 W InputDispatcher: Attempted to unregister already unregistered input channel
11-08 11:58:16.848 4965 4979 I .appcompat.tes: Background concurrent copying GC freed 83477(3643KB) AllocSpace objects, 1(20KB) LOS objects, 49% free, 1
11-08 11:58:16.851 408 408 W SurfaceFlinger: couldn't log to binary event log: overflow.
11-08 11:58:16.874 544 5729 E WindowManager: Window Session Crash
11-08 11:58:16.874 544 5729 E WindowManager: java.lang.IllegalArgumentException: Requested window android.os.BinderProxy@f459d14 does not exist
11-08 11:58:16.874 544 5729 E WindowManager:     at com.android.server.wm.WindowManagerService.windowForClientLocked(WindowManagerService.java:5565)
11-08 11:58:16.874 544 5729 E WindowManager:     at com.android.server.wm.WindowManagerService.windowForClientLocked(WindowManagerService.java:5557)
11-08 11:58:16.874 544 5729 E WindowManager:     at com.android.server.wm.WindowManagerService.reportSystemGestureExclusionChanged(WindowManagerServ
11-08 11:58:16.874 544 5729 E WindowManager:     at com.android.server.wm.Session.reportSystemGestureExclusionChanged(Session.java:480)
11-08 11:58:16.874 544 5729 E WindowManager:     at android.view.IWindowSession$Stub.onTransact(IWindowSession.java:1240)
11-08 11:58:16.874 544 5729 E WindowManager:     at com.android.server.wm.Session.onTransact(Session.java:163)
11-08 11:58:16.874 544 5729 E WindowManager:     at android.os.Binder.execTransactInternal(Binder.java:1182)
11-08 11:58:16.874 544 5729 E WindowManager:     at android.os.Binder.execTransact(Binder.java:1141)
11-08 11:58:16.875 544 5729 W Binder : Caught a RuntimeException from the binder stub implementation.
11-08 11:58:16.875 544 5729 W Binder : java.lang.IllegalArgumentException: Requested window android.os.BinderProxy@f459d14 does not exist
11-08 11:58:16.875 544 5729 W Binder :     at com.android.server.wm.WindowManagerService.windowForClientLocked(WindowManagerService.java:5565)
11-08 11:58:16.875 544 5729 W Binder :     at com.android.server.wm.WindowManagerService.windowForClientLocked(WindowManagerService.java:5557)
11-08 11:58:16.875 544 5729 W Binder :     at com.android.server.wm.WindowManagerService.reportSystemGestureExclusionChanged(WindowManagerService.java
11-08 11:58:16.875 544 5729 W Binder :     at com.android.server.wm.Session.reportSystemGestureExclusionChanged(Session.java:480)
11-08 11:58:16.875 544 5729 W Binder :     at android.view.IWindowSession$Stub.onTransact(IWindowSession.java:1240)
11-08 11:58:16.875 544 5729 W Binder :     at com.android.server.wm.Session.onTransact(Session.java:163)
11-08 11:58:16.875 544 5729 W Binder :     at android.os.Binder.execTransactInternal(Binder.java:1182)
11-08 11:58:16.875 544 5729 W Binder :     at android.os.Binder.execTransact(Binder.java:1141)
11-08 11:58:17.006 4965 4981 W System : A resource failed to call release.
11-08 11:58:17.006 4965 4981 W System : A resource failed to call release.
... way more of those ...
11-08 11:58:17.071 4965 4981 W System : A resource failed to call release.
11-08 11:58:17.071 4965 4981 W System : A resource failed to call release.
11-08 11:58:17.073 544 573 I ActivityTaskManager: Displayed androidx.appcompat.test/androidx.appcompat.app.NightModeActivity: +571ms
11-08 11:58:17.076 4965 4965 D LifecycleMonitor: Lifecycle status change: androidx.appcompat.app.NightModeActivity@89bccbf in: PAUSED
11-08 11:58:17.077 4965 4965 D LifecycleMonitor: Lifecycle status change: androidx.appcompat.app.NightModeActivity@89bccbf in: STOPPED
11-08 11:58:17.078 4965 4965 D LifecycleMonitor: Lifecycle status change: androidx.appcompat.app.NightModeActivity@89bccbf in: DESTROYED
```

Then we start another activity, but the recreate seems to happen on the one that was destroyed a long time ago (object 89bccbf):

```
11-08 11:58:22.043 4965 4987 D NightModeUtils: setNightModeAndWaitForRecreate on Activity: androidx.appcompat.app.NightModeActivity@89bccbf to mode: 2 u
```

So we must be hitting a race condition or stale data or something.

al...@google.com <al...@google.com> [#24](#)

Yeah, the `ActivityTestRule` is giving us a stale activity for some reason.

al...@google.com <al...@google.com> [#25](#)

+cc brettchabot since (1) he owns this file and (2) we were just talking about things that are hard to test. We're simulating screen rotation here.

al...@google.com <al...@google.com> [#26](#)

Rather, owns `ActivityTestRule`. I'm going to see if we can bypass it and rely on callbacks or something.

al...@google.com <al...@google.com> [#27](#)

This feels awfully familiar.

We have this in the other `DoesNotRecreate` test:

```
// Set local night mode to MODE_NIGHT_YES and wait for state RESUMED.
val initialActivity = activityRule.launchActivity(null)
LifecycleOwnerUtils.waitForUntilState(initialActivity, Lifecycle.State.RESUMED)
setNightModeAndWaitForRecreate(initialActivity, MODE_NIGHT_YES, setMode)
```

But this in the flaky one:

```
// Set local night mode to YES
setNightModeAndWaitForRecreate(activityRule, MODE_NIGHT_YES, setMode)

val activity = activityRule.activity
```

So we're probably hitting a race condition where the activity is *actively* restarting when the test begins. I'll update the code to let it settle first.

al...@google.com <al...@google.com> [#28](#)

Brett, if you do happen to check in here, we're launching an activity that in many cases will immediately call `Activity.recreate()` on itself. This will happen in g3 as well, since most apps a

br...@google.com <br...@google.com> [#29](#)

I haven't had a chance to digest all the commentary here yet, but IIUC there are known issues with `ActivityTestRule` and activity recreation, because it keeps a reference to an `Activity`.

`ActivityScenario` should not have this problem.

al...@google.com <al...@google.com> [#30](#)

IIRC there was an issue with `ActivityScenario` not supporting this use case, but I didn't file a bug so shame on me. I'll attempt to migrate and file one if I can't.

al...@google.com <al...@google.com> [#31](#)

`ActivityScenario` does not appear to be able to handle `Activity.recreate()` being called by anything other than `ActivityScenario.recreate()`, which means it's not going to work fo

al...@google.com <al...@google.com> [#32](#)

And, generally, it doesn't look like `ActivityScenario` is a good choice for testing the interaction between `AppCompatActivity`/Core and that platform's actual lifecycle events, given that it drives its

al...@google.com <al...@google.com> [#33](#)

Fix CL currently failing on CF 29 where it fails to detect activity `recreate()`.


Does that mean the activity is not recreated? Unclear.

al...@google.com <al...@google.com> [#34](#)

Accepted by al...@google.com.

Hah, the failure screenshot is mid-rotation animation. I think what may be happening is that either (a) it's getting stuck or (b) it's destroying the activity in the background but the animation is

What we *actually* want is a sync point for activity creation.

 **androidx.appcompat.app.NightModeRotateRecreatesActivityWithConfigTestCase#testRotateRecreatesActivityWithConfig[0]-127.0.0.1_35847-screenshot-on-failure_708999618605**
43 KB [View](#) [Download](#)

ap...@google.com <ap...@google.com> [#35](#)

Marked as fixed.

Project: platform/frameworks/support
Branch: androidx-main

commit b45f113d7ca003b28b65fa7fd005622d6b61ad00
Author: Alan Viverette <alanv@google.com>
Date: Fri Nov 13 17:59:49 2020

Fix race condition in Recreates test by mirroring DoesNotRecreate setup

Also updates DoesNotRecreate to fix Kotlin lint warnings.

Test: NightModeRotate*ActivityTestCase
Fixes: 152673431
Bug: 174742601
Change-Id: I64dc2006534bd39d8da1f4fe8f37cd8814cb3a50

M `appcompat/appcompat/src/androidTest/java/androidx/appcompat/app/NightModeActivity.java`
M `appcompat/appcompat/src/androidTest/java/androidx/appcompat/app/NightModeRotateDoesNotRecreateActivityTestCase.kt`
M `appcompat/appcompat/src/androidTest/java/androidx/appcompat/app/NightModeRotateRecreatesActivityWithConfigTestCase.kt`

<https://android-review.googlesource.com/1530721>

