Error when trying to debug android test of library from AS

Fixed Bug P1 + Add Hotlist

STATUS UPDATE No update yet. Edit

DESCRIPTION sp...@google.com created issue #1

When I try to debug a library module's android test in AS, I see the following pop up error (image attached):

```
Error running 'ExampleInstrumentedTest':
Facet is not found for com.example.mylibrary.test
```

To repro:

1. Create a new project in AS (Giraffe alpha02)
2. Add a new Android library module
3. Go to library's `ExampleInstrumentedTest.kt` source file
4. Right click in gutter by class name, and click `Debug 'ExampleInstrumentedT...'`

Expected behavior: test runs successfully

Actual behavior: test fails and error message appears

Observation: debugging the app's android test in a similar way is working.

Emre, I assigned this bug to you because I know you have some context here, but please feel free to reassign it.

Screen Shot 2023-02-06 at 4.57.59 PM.png
49 KB View Download

✓ Mentioned issues (2) ✓ Links (3)

Mentioned issues (2)

P3 Debug test instrumentation run failed due to Process crashed "https://issuetracker.google.com/264388220"

P3 Unable to debug JNI since upgrade to Flamingo "https://issuetracker.google.com/278621689"

Links (3)

"http://cs/studio-main/tools/adt/idea/execution/common/src/com/android/tools/idea/execution/common/debug/utils/FacetFinder.kt;l=44;rcl=baf40178..."
"http://cs/studio-main/tools/adt/idea/project-system-gradle/src/com/android/tools/idea/projectsystem/gradle/GradleModuleSystem.kt;l=387;rcl=baf4017... "
"Something like: http://ag/21325527 (WIP)" em..

COMMENTS All comments

sp...@google.com <sp...@google.com>

Reassigned to sm...@google.com.

sm...@google.com <sm...@google.com>

Accepted by sm...@google.com.

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

These are all `FacetFinder.findFacetForProcess()` issues that I've known to be problematic for a while.

http://cs/studio-main/tools/adt/idea/execution/common/src/com/android/tools/idea/execution/common/debug/utils/FacetFinder.kt;l=44;rcl=baf401786db1df18324048800ffec1bee07264

Here's my take on this issue:

1. In the first part of `FacetFinder.findFacetForProcess()` where we check for package name match:

```
val facets = project.getAndroidFacets()
```

...returns the facets for holder modules. This means we can only compare the `processName` against the `packageName`.

=> We should also check the facet that corresponds to the `androidTest` module. This will enable us to compare against `testPackageName`.

2. We call:

```
facet.getModuleSystem().getApplicationIdProvider().packageName
```

But, we won't be able to call the equivalent:

```
androidTestFacet.getModuleSystem().getApplicationIdProvider().testPackageName
```

because `GradleModuleSystem.getApplicationIdProvider` always passes `forTests=false`.

http://cs/studio-main/tools/adt/idea/project-system-gradle/src/com/android/tools/idea/projectsystem/gradle/GradleModuleSystem.kt;l=387;rcl=baf401786db1df18324048800ffec1bee072

=> It should be made more flexible such that, if the given Module is an `androidTest` Module, then it should automatically pass `forTests=true`.

3. In the second part of `FacetFinder.findFacetForProcess()` where we traverse the dependency graph:

```
ModuleUtilCore.getDependencies(facet.mainModule, dependentModules)
```

...uses only the `mainModule`, so it only checks for `implementation` dependencies.

=> We should traverse the dependency graph for both `mainModule` and `androidTestModule` so that we check both `implementation` and `androidTestImplementation` dependencies.

3. In `FacetFinder.findFacetForProcess()`, we always return the `facet` that we iterate over:

```
return facet
```

...which are the facets for the "holder modules".

=> If processName matches an `androidFacet`, then we should return the `androidFacet`. This way, downstream calls to `getPackageName()` will automatically return the "test application id

---

em...@google.com <em...@google.com> #3

Feel free to assign it back to me if you'd like.

---

em...@google.com <em...@google.com> #4

Something like: http://ag/21325527 (WIP)

Message last modified on Feb 8, 2023 04:48PM

---

sm...@google.com <sm...@google.com>

Assigned to em...@google.com.

---

em...@google.com <em...@google.com> #5

Reassigned to so...@google.com.

solodkyy@ will update this bug per http://ag/21325527

---

so...@google.com <so...@google.com>

Accepted by so...@google.com.

---

ga...@google.com <ga...@google.com>

Status: New

---

cm...@google.com <cm...@google.com>

Accepted by cm...@google.com.

---

yo...@gmail.com <yo...@gmail.com> #6

still hitting this on Giraffe Alpha 8, when I try to debug a test. Works fine in Flamingo Beta 4.

on macOS `12.6.2` using AGP `7.4.2`

from `idea.log`:

```
com.intellij.execution.ExecutionException: Facet is not found for com.example.mylib.test
        at com.android.tools.ndk.run.editor.AutoAndroidDebugger.getDebugProcessStarterForNewProcess(AutoAndroidDebugger.java:118)
        at com.android.tools.ndk.run.editor.AutoAndroidDebugger.getDebugProcessStarterForNewProcess(AutoAndroidDebugger.java:44)
        at com.android.tools.idea.execution.common.debug.DebugSessionStarter$attachDebuggerToStartedProcess$1.invokeSuspend(DebugSessionStarter.kt:69)
        at kotlin.coroutines.jvm.internal.BaseContinuationImpl.resumeWith(ContinuationImpl.kt:33)
        at kotlinx.coroutines.DispatchedTask.run(DispatchedTask.kt:106)
        at kotlinx.coroutines.EventLoopImplBase.processNextEvent(EventLoop.common.kt:284)
        at kotlinx.coroutines.BlockingCoroutine.joinBlocking(Builders.kt:85)
        at kotlinx.coroutines.BuildersKt__BuildersKt.runBlocking(Builders.kt:59)
```

```
at kotlinx.coroutines.BuildersKt.runBlocking(Unknown Source)
at com.intellij.openapi.progress.CoroutinesKt$runBlockingCancellable$2.invoke(coroutines.kt:112)
at com.intellij.openapi.progress.CoroutinesKt$runBlockingCancellable$2.invoke(coroutines.kt:106)
at com.intellij.openapi.progress.CancellationKt$ensureCurrentJob$1$1.invoke(cancellation.kt:84)
at com.intellij.openapi.progress.CancellationKt.withCurrentJob$lambda$0(cancellation.kt:17)
at com.intellij.openapi.progress.Cancellation.withCurrentJob(Cancellation.java:60)
at com.intellij.openapi.progress.CancellationKt.withCurrentJob(cancellation.kt:17)
at com.intellij.openapi.progress.CancellationKt.executeWithJobAndCompleteIt(cancellation.kt:125)
at com.intellij.openapi.progress.CancellationKt.ensureCurrentJob$lambda$1(cancellation.kt:82)
at com.intellij.openapi.progress.impl.CoreProgressManager.silenceGlobalIndicator(CoreProgressManager.java:964)
at com.intellij.openapi.progress.CancellationKt.ensureCurrentJob(cancellation.kt:81)
at com.intellij.openapi.progress.CoroutinesKt.runBlockingCancellable(coroutines.kt:106)
at com.android.tools.idea.execution.common.debug.DebugSessionStarter.attachDebuggerToStartedProcess(DebugSessionStarter.kt:66)
at com.android.tools.idea.run.tasks.DefaultConnectDebuggerTask.perform(DefaultConnectDebuggerTask.kt:52)
at com.android.tools.idea.testartifacts.instrumented.GradleAndroidTestRunConfigurationExecutor$debug$1.invokeSuspend(GradleAndroidTestRunConfigura
at kotlin.coroutines.jvm.internal.BaseContinuationImpl.resumeWith(ContinuationImpl.kt:33)
at kotlinx.coroutines.DispatchedTask.run(DispatchedTask.kt:106)
at kotlinx.coroutines.EventLoopImplBase.processNextEvent(EventLoop.common.kt:284)
at kotlinx.coroutines.BlockingCoroutine.joinBlocking(Builders.kt:85)
at kotlinx.coroutines.BuildersKt__BuildersKt.runBlocking(Builders.kt:59)
at kotlinx.coroutines.BuildersKt.runBlocking(Unknown Source)
at com.intellij.openapi.progress.CoroutinesKt$runBlockingCancellable$2.invoke(coroutines.kt:112)
at com.intellij.openapi.progress.CoroutinesKt$runBlockingCancellable$2.invoke(coroutines.kt:106)
at com.intellij.openapi.progress.CancellationKt$ensureCurrentJob$1$1.invoke(cancellation.kt:84)
at com.intellij.openapi.progress.CancellationKt.withCurrentJob$lambda$0(cancellation.kt:17)
at com.intellij.openapi.progress.Cancellation.withCurrentJob(Cancellation.java:60)
at com.intellij.openapi.progress.CancellationKt.withCurrentJob(cancellation.kt:17)
at com.intellij.openapi.progress.CancellationKt.executeWithJobAndCompleteIt(cancellation.kt:125)
at com.intellij.openapi.progress.CancellationKt.ensureCurrentJob$lambda$1(cancellation.kt:82)
at com.intellij.openapi.progress.impl.CoreProgressManager.silenceGlobalIndicator(CoreProgressManager.java:964)
at com.intellij.openapi.progress.CancellationKt.ensureCurrentJob(cancellation.kt:81)
at com.intellij.openapi.progress.CoroutinesKt.runBlockingCancellable(coroutines.kt:106)
at com.android.tools.idea.testartifacts.instrumented.GradleAndroidTestRunConfigurationExecutor.debug(GradleAndroidTestRunConfigurationExecutor.kt:
at com.android.tools.idea.run.configuration.execution.AndroidConfigurationExecutorRunProfileState.debug(AndroidConfigurationExecutor.kt)
at com.android.tools.idea.run.DefaultStudioProgramRunner.run(DefaultStudioProgramRunner.kt:104)
at com.android.tools.idea.run.configuration.AndroidConfigurationProgramRunner$execute$4.run(AndroidConfigurationProgramRunner.kt:123)
at com.intellij.openapi.progress.impl.CoreProgressManager.startTask(CoreProgressManager.java:429)
at com.intellij.openapi.progress.impl.ProgressManagerImpl.startTask(ProgressManagerImpl.java:114)
at com.intellij.openapi.progress.impl.CoreProgressManager.lambda$runProcessWithProgressAsynchronously$6(CoreProgressManager.java:480)
at com.intellij.openapi.progress.impl.ProgressRunner.lambda$submit$3(ProgressRunner.java:252)
at com.intellij.openapi.progress.impl.CoreProgressManager.lambda$runProcess$2(CoreProgressManager.java:186)
at com.intellij.openapi.progress.impl.CoreProgressManager.lambda$executeProcessUnderProgress$13(CoreProgressManager.java:604)
at com.intellij.openapi.progress.impl.CoreProgressManager.registerIndicatorAndRun(CoreProgressManager.java:679)
at com.intellij.openapi.progress.impl.CoreProgressManager.computeUnderProgress(CoreProgressManager.java:635)
at com.intellij.openapi.progress.impl.CoreProgressManager.executeProcessUnderProgress(CoreProgressManager.java:603)
at com.intellij.openapi.progress.impl.ProgressManagerImpl.executeProcessUnderProgress(ProgressManagerImpl.java:60)
at com.intellij.openapi.progress.impl.CoreProgressManager.runProcess(CoreProgressManager.java:173)
at com.intellij.openapi.progress.impl.ProgressRunner.lambda$submit$4(ProgressRunner.java:252)
at java.base/java.util.concurrent.CompletableFuture$AsyncSupply.run(Unknown Source)
at java.base/java.util.concurrent.ThreadPoolExecutor.runWorker(Unknown Source)
at java.base/java.util.concurrent.ThreadPoolExecutor$Worker.run(Unknown Source)
at java.base/java.util.concurrent.Executors$PrivilegedThreadFactory$1$1.run(Unknown Source)
at java.base/java.util.concurrent.Executors$PrivilegedThreadFactory$1$1.run(Unknown Source)
at java.base/java.security.AccessController.doPrivileged(Unknown Source)
at java.base/java.util.concurrent.Executors$PrivilegedThreadFactory$1.run(Unknown Source)
at java.base/java.lang.Thread.run(Unknown Source)
```

Message last modified on Mar 8, 2023 06:56AM

ls...@gmail.com <ls...@gmail.com> #7

Stumbled upon this bug on Android Studio Giraffe Canary 9. Workaround that worked for me: edit generated run configuration and change debugger type from "Detect Automatically" to "Java

au...@google.com <au...@google.com> #8

Is this being looked at?

lp...@google.com <lp...@google.com> #9

Just upgraded to Giraffe Canary 11 - issue still exists, but now this time it seems to hang indefinitely instead of crashing. The fix of setting to java only still works for me

au...@google.com <au...@google.com> #10

Assigned to je...@google.com.

assigning to jedo@ since cmw is OOO

je...@google.com <je...@google.com>

Reassigned to ga...@google.com.

---

ga...@google.com <ga...@google.com> #11

Reassigned to cm...@google.com.

We want to fix this in Giraffe Betas. Chris has already started looking into this and once he's back there is enough time to address this bug, so I'm reassigning it back to him.

---

au...@google.com <au...@google.com> #12

Using a debugger is a critical part of an IDE experience. It doesn't seem great that we knew about this since February and we are assigning this to a person who is OOO for another 2 weeks.

---

rp...@google.com <rp...@google.com> #13

Reassigned to em...@google.com.

Emre, this bug is marked as blocking RC1. Is there anything we can do to work around the issue until it is fixed in the project system? It feels to me it is way too late anyways in "G" to start wo
we would have to cherry pick it anyways (I am assuming the "real" fix is not trivial and not low risk).

---

em...@google.com <em...@google.com> #14

We can still use http://ag/21325527 from comment#4.

---

em...@google.com <em...@google.com> #15

Marked as fixed.

A fix is submitted for Android Studio Giraffe beta 2 and Android Studio H canary 1.

---

rp...@google.com <rp...@google.com> #16

Thank you Emre!

---

be...@citymapper.com <be...@citymapper.com> #17

As noted in https://issuetracker.google.com/issues/264388220 and https://issuetracker.google.com/issues/278621689 this also affects Flamingo. There's no popup in studio, but the test st
debugger type from "Detect Automatically" to "Java Only" works around it.

Since it was considered P1/S0 for G, is there any chance of the fix also being cherry-picked into an F patch?

---

jf...@block.xyz <jf...@block.xyz> #18

I have reports from someone who duplicated this on both Electric Eel and Flamingo, with general debugging, not specific to instrumentation tests. Suggestion in #17 fixes it

---

nu...@gmail.com <nu...@gmail.com> #19

Are there any work-arounds for this issue? Why does it say "FIXED" when it clearly isn't in Flamingo 2022.2.1 Patch 1? I have an Android library module with a large C++ codebase that builds a
several JNI bindings. The androidTest cannot be debugged because the instrumentation process crashes! And it runs in non-debug mode, but what good is that if a test does not pass becaus
and it is an IMPOSSIBLE task to debug and fix the native code? It does not help to just run a debug session in Java mode. I am using AS since it was pre-1.0 (Eclipse was still around) and I ca
later it still takes months and months to fix critical things, or having to wait some Googler to come back from holiday and take a look! Us, actual developers that use your product 24/7, have t
looking for work-arounds to make the IDE work, only to hit a wall with issues that "won't fix" or pray that the next update doesn't (again) break the build or make something not work, or worse,
we did something wrong. Maybe just 0.5% of your users actually need to debug low-level C++ code inside an Android library but I can guarantee to you that those users are not the newbies th
zero knowledge into a Hello world Kotlin app, they expect things to work. WE are your product, so show us some respect, or else we can always switch the camp.

```
Could not get applicationId for (my module name). Project type: PROJECT_TYPE_LIBRARY
java.lang.Throwable
        at com.android.tools.idea.run.GradleApplicationIdProvider.getPackageName(GradleApplicationIdProvider.kt:111)
        at com.android.tools.idea.execution.common.debug.utils.FacetFinder.findFacetForProcess(FacetFinder.kt:48)
```

---

em...@google.com <em...@google.com> #20

> is there any chance of the fix also being cherry-picked into an F patch?

The patch that was applied to Giraffe Beta and Hedgehog Canary (in comment#14) was a small, tactical native debugger fix (as requested in comment#13). It simply added the following:

```
holderFacet.androidTestModule?.androidFacet
    ?.getModuleSystem().getApplicationIdProvider().testPackageName
```

...as a potential debug target when looking for a test package name. In words: For a library that has an instrumented test module (i.e., `androidTest`), we now also check its "test package na

Unfortunately, this fix cannot be applied to Flamingo, because in Flamingo the project system does not return a valid "test package name" for libraries (i.e., returns `null` for this statement).

Applying a project system fix (and then a debugger fix backport) to Flamingo is considered too complex/risky to be applied at a "Patch 2" stage.

> Why does it say "FIXED" when it clearly isn't in Flamingo 2022.2.1 Patch 1?

It is Android Studio team practice to mark bugs as fixed when the root cause is fixed in the `main` branch. We cherry-picked the fix into Android Studio Giraffe starting from Beta 2 (available t back-ported to Flamingo. I tried to explain the rationale above.

> I can't believe that 10 years later it still takes months and months to fix critical things

I am really sorry for this poor IDE experience you are observing. We had big changes in the parts of Android Studio that are involved here: both on the project system side, and on the debugge side, for instance, we refactored the debugger startup infrastructure to better support more devices (e.g., Wear OS). However, we unfortunately introduced this regression which was not caug infrastructure. In order to prevent/catch such regressions in the future, I am currently adding new integration tests to cover these instrumented test scenarios. Once again, I apologize for the

---

Please try debugging using Android Studio Giraffe Beta 2 or newer. I hope you will be able to debug your instrumented tests using it.