Android Gradle Plugin v3.6.0 fails to report C++ sources to Android Studio

STATUS UPDATE No update yet. Edit

DESCRIPTION je...@gmail.com created issue #1

I have an Android project containing an app and a library module which contains both Java and C++ sources and uses CMake.

When I use Android Studio 3.6.0 with v3.6.0 of the Android Gradle Plugin:
1) Navigate to Class shows none of my C++ classes
2) Navigate to File shows .cpp.o files, but none of my .cpp or .hpp files
3) The Project -> Android tree version shows Xxx.dir entries from within .externalNativeBuild rather than showing appropriate C++ source directories

My C++ sources are outside my project directory, so this leaves me really hamstrung in terms of working on the C++ sources with Android Studio!

This project works fine with Android Studio 3.5.3 with the Android Gradle Plugin 3.5.3 and with Android Studio 3.6.0 with the Android Gradle Plugin 3.5.3.

I have another Android app project using CMake which works fine with the 3.6.0 plugin.

It is unclear what the critical difference between these 2 projects is but:
1) The broken project uses CMake 3.16.4 (and requires a newer CMake than any available via SDK Manager), whereas the one the working one does not specify and presumably uses the default,
2) The broken project follows the normal top-level build.gradle, app, and library module pattern. The working project jams everything into a single build.gradle.
3) The broken project generates a warning on startup that the working project does not (see https://issuetracker.google.com/issues/150274431):
"Unsupported Modules Detected: Compilation is not supported for following modules: Android. Unfortunately you can't have non-Gradle Java modules and Android-Gradle modules in one project
4) The broken project uses NDK r21, whereas the working project uses NDK r20.

Build: AI-192.7142.36.36.6200805, 202002120909,

AI-192.7142.36.36.6200805, JRE 1.8.0_212-release-1586-b04x64 JetBrains s.r.o, OS Windows 10(amd64) v10.0 , screens 1920x1080

AS: 3.6; Kotlin plugin: 1.3.61; Android Gradle Plugin: 3.6.0; Gradle: 5.6.4; NDK: from local.properties: (not specified), latest from SDK: (not found); LLDB: LLDB 3.1 (revision: 3.1.4508709); CMake: f
PATH: (not found)

✓ Mentioned issues (3) ✓ Links (1)

Mentioned issues (3)

P3	Getting presumably inaccurate "Unsupported Modules Detected" warning	"https://issuetracker.google.com/150274431"
P2	Android Studio doesn't support CMake's OBJECT library	"https://issuetracker.google.com/144938511"
P2	When debugging app library module's native symbols are not found	"The fix I mentioned in comment#21 was to make sure the debugger can find your debug symbols. I think I mixed up this bug

Links (1)

"But nevertheless, symbol stripping can be disabled using packagingOptions.doNotStrip in Android DSL."

COMMENTS

je...@gmail.com <je...@gmail.com> #2

This issue is forcing my team to downgrade to v3.5.3 of the Android Gradle Plugin until this issue is resolved!

uc...@google.com <uc...@google.com>

Assigned to an...@google.com.

ar...@google.com <ar...@google.com>

Reassigned to tg...@google.com.

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

Can you please delete the .externalNativeBuild directory (for the library module that has the C++ code), and then do a clean build from scratch?

I'm unable to reproduce this issue on my end, here are the steps I took:
1. Using Android Studio 3.6 and Gradle Plugin 3.6; Ndk 21; CMake 3.16.4
2. Created a new application with "app" and "library" modules.

3. Added dependency from app to library module.
4. Added C++ code and CMakeLists.txt outside the app directory.
5. Referred to the CMakeLists.txt from the library/build.gradle as ../../cpp/CMakeLists.txt
6. Added a Java class to library module with a native method implemented in C++.
7. Instantiate that java class from app module, and call the native method.
8. Sync.

Then, I can use the IDE to navigate from C++/Java for the native method; from C++ class header to C++ implementation, etc. And, I also see the file tree in Android view, including C++ files. D
different from you?

je...@gmail.com <je...@gmail.com> #4

I've done multiple attempts with `git clean -xfd`, reloading the project and rebuilding from scratch -- to no avail.

I don't see anything obvious in what you've done that is different. And, yes, I've tried again with AS and Android Gradle Plugin 3.6.1.

je...@gmail.com <je...@gmail.com> #5

I am using Conan and the `conanInstall` task, though I suspect that is not the cause of this -- as it is only installing dependencies, not any of my sources.

Message last modified on Mar 14, 2020 11:01PM

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

Note that the broken project has no native compilation at the app layer, only at the library module layer -- in case that makes a difference. I'm struggling to understand what other differences
works...

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

Does the Android Gradle Plugin care about which type of CMake libraries are involved?

I ask as all but the top-level linkage in our failing case are via "object" libraries to get JNI/SWIG symbol visibility to propagate properly. (This also requires that we create/touch an empty file t

Due to the use of SWIG we also have to do:

```
project.afterEvaluate {
    javaPreCompileDebug.dependsOn externalNativeBuildDebug
    javaPreCompileRelease.dependsOn externalNativeBuildRelease
}
```

to ensure Java compilation is done after native compilation completes.

With the 3.5.3 Android Gradle Plugin (i.e. where things are working ok), under `cpp` in the `Android` view of the `Project` panel I see a `CMakeModuleName` node for each CMake library involved
and the final shared library. Each of these nodes contains the appropriate C++ and SWIG sources. I see these immediate after project sync completes -- and see no warnings in the sync outpu
CMakeLists.txt in each node, but that's not the case. This seems to depend on how the sources and CMakeLists.txt dependencies are structured.)

With the 3.6.1 Android Gradle Plugin, however, after under synchronize completes I only see the `includes` and the sources for the top-level shared library module directly under the `cpp` node
no use as there's only 1 dummy source file just to get the linker to link all of the other object libraries and one version file.)

I also see:

```
C:\git-repos\daltonplayer\CMakeLists.txt : C/C++ debug|arm64-v8a : CMake Deprecation Warning:
  The 'cmake-server(7)' is deprecated.  Please port clients to use the
  'cmake-file-api(7)' instead.
```

in the project sync build log.

After I build, under `cpp` I then see `includes` and a slew of `SomeDir.dir` nodes, one for each CMake library (both object libraries and the final shared library) with `SomeDir` being the top-leve
of these nodes points to a corresponding subdirectory of `.externalNativeBuild`, thus showing .o files, not source files!

To make matters even more fun, I note that I now get a build failure at the app level as well (where all my native compilation and linking is at the library module level):

```
> Task :app:stripDebugDebugSymbols FAILED

FAILURE: Build failed with an exception.

* What went wrong:
Execution failed for task ':app:stripDebugDebugSymbols'.
> No version of NDK matched the requested version 20.0.5594570. Versions available locally: 21.0.6113669
```

I'd apparently not noticed this before as I previously had both NDK r20 and r21 installed.

I'm building with NDK r21 (explicitly via `ndkVersion`), but I am including some shared libraries that are built elsewhere using NDK r20. With Android Gradle Plugin 3.5.3 I don't need NDK r20 i
attempt to strip shared libraries I'm including from elsewhere??? Or what gives here?

Message last modified on Mar 15, 2020 01:33AM

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

Status: Duplicate of 144938511

Hi, object library is probably why. In that case this seems to be a duplicate of https://issuetracker.google.com/issues/144938511. Could you try Android Studio 4.1 Canary 2 or later and see i
version of Android Gradle plugin.

je...@gmail.com <je...@gmail.com> #9

So... I just tried this (after doing `git clean -xfd` on my Git repo to be sure nothing untoward was hanging around)...

Unfortunately, when I did so I ran into another issue when attempting to build the project, I get

```
> Task :dpComp:mergeDebugNativeLibs FAILED

FAILURE: Build failed with an exception.

* What went wrong:
Execution failed for task ':dpComp:mergeDebugNativeLibs'.
> A failure occurred while executing com.android.build.gradle.internal.tasks.Workers$ActionFacade
   > More than one file was found with OS independent path 'lib/armeabi-v7a/libcollab_services.so'
```

I search for this file in the repository and see nothing untoward from my perspective. I see a copy in build\intermediates\cmake\debug\obj\armeabi-v7a\libcollab_services.so, but I don't see

This is with Android Studio 4.1 Canary 3 -- and corresponding Android Gradle Plugin and Gradle versions.

je...@gmail.com <je...@gmail.com> #10

To be clear it is hard to say this is a duplicate of an issue fixed in Canary 2 when Canary 3 fails for me...

je...@gmail.com <je...@gmail.com> #11

I also discovered that applying `com.jaredsburrows:gradle-license-plugin:0.8.70` as a build plugin seems to produce some of the same issues even on the Android Gradle Plugin 3.5.3

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

Assigned to em...@google.com.

Hi, do you have multiple Android modules that include native dependencies? For example, multiple modules with `externalNativeBuild` block in your build.gradle files. Or, do you use `jniLi`

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

If you happen to have a section like this in your `build.gradle` file:

```
    sourceSets {
        main {
            jniLibs.srcDirs = ["build/intermediates/cmake/debug/obj/"]
        }
    }
```

Then it would instruct AGP to put two same-named binaries into the same location in the APK, and AGP will give this as an error.

If this indeed is the case, then I am guessing you are trying to include "debug" version of your libraries in the APK, which is something we do not encourage. Android Studio debugger should r
not working for you).

But nevertheless, symbol stripping can be disabled using packagingOptions.doNotStrip in Android DSL.

je...@gmail.com <je...@gmail.com> #14

We only have 1 library module in our project.

And we do use jniLibs srcDir as follows:

```
        debug {
            jniLibs {
                // pull shared libraries for each ABI variant from conan debug packages
                srcDir '.externalNativeBuild/conan/debug'
            }
        }
        release {
            jniLibs {
                // pull shared libraries for each ABI variant from conan release packages
                srcDir '.externalNativeBuild/conan/release'
            }
        }
```

As noted, we need this to include shared libraries from Conan dependencies we've installed via our conanInstall() task.

This seems like the right way to do this, no? It seems like it has been necessary and correct to date...

Message last modified on Apr 10, 2020 09:42PM

je...@gmail.com <je...@gmail.com> #15

So... I tried commenting out these lines...

With AS 4.0b4, this allows my build to complete and app to run! (I have to assume that at some point AS started automatically including shared library dependencies for me?) But the Android
my C++ sources. (It shows the CMake output dirs containing the .o files and my includes directories, which is about useless.) Ctrl-N also does not seem to understand anything about my C++

With AS 4.1c5, commenting out these lines doesn't quite produce a working build. I get a Java compilation error where it can't find BuildConfig for my library package (!). (This code works fin
then, yes, the build completes and the app runs. I still don't see my CMake modules or C++ sources in the Android view of the Project tree, though. (Unlike AS 4.0b4, the tree view shows only
essentially empty in my case, nothing for my object libraries.) Ctrl-N does know about my C++ classes, though, so that's an improvement over 4.0b4!

In both AS 4 and 4.1, I get

```
Tests model errors
    Warning:Warning:<i><b>project ':dpComp'</b>
Details: java.util.ConcurrentModificationException: null</i>
```

during project sync.

je...@gmail.com <je...@gmail.com> #16

Also, I did a bit of investigation. With the Android Gradle Plugin 3.5.3, those lines are 100% necessary. Commenting them out results in the libraries in question not being included in the AAR.

It's great that they're being automatically included for me in later versions. But some heads up to this change, e.g. "if you see this error this is probably what's going on and why..." would be re
somewhere???

em...@google.com <em...@google.com> #17

Regarding comment#14, The directories you listed are surprising to me, because:

- Starting with Android Gradle Plugin 3.5, the `<module>/.externalNativeBuild` directory was renamed to `<module>/.cxx`. That `.externalNativeBuild` directory is no longer created
- Where does the `conan/` subdirectory come from? Are you explicitly telling Conan to install your libraries into the `.externalNativeBuild` folder?

Regarding comment#7 Dependency:

- The `(javaPreCompile<>, externalNativeBuild<>)` dependency you added is harmless. It should not affect the final result.

Regarding comment#7 Symbol Stripping:

- You said your main `app` module has no native compilation. However, it seems like you added your `jniLibs` section to this `app` module.
- When Gradle tries to strip symbols from those `jniLibs`, it needs to use the `strip` tool from Android NDK.
- I am guessing you only added `ndkVersion` to the `build.gradle` for your `library` module (after all that's the only one that builds native code). Please make sure you also add the sam
  Otherwise, it will try to search for the "default" NDK version for that Android Gradle Plugin (which is why you are seeing r20 vs r21 error).
- Alternatively, you can move your `jniLibs` section from the `build.gradle` of your app module to your library module (which is probably a cleaner solution).

If you could provide us a small project that reproduces your issues (e.g., has a similar project structure, uses Conan in the same way you do, builds native code the same style, and reproduce
fixing those issues.

je...@gmail.com <je...@gmail.com> #18

The `.externalNativeBuild` bit is because we have:

```
buildStagingDirectory '.externalNativeBuild'  // be explicit about this in case default changes as we rely on this
```

in our build.gradle, because we rely upon this location as noted. I didn't realize the default had changed, but this is precisely why we were explicit here.

Yes, we're explicitly telling Conan to install shared libraries and jars from Conan-provided dependencies via `.externalNativeBuild/conan`. If we don't explicitly point these out to the Androi

I'm glad the `javaPreCompile / externalNativeBuild` dependency is deemed unrelated to this issue -- as it is absolutely necessary here due to the use of SWIG and the presence of Java c
classes.

If I remove the `jniLibs` section of my library's build.gradle, then I won't have these .so's in my final APK or app bundle -- at least with the 3.5.3 plugin -- and I do need these .so's at runtime. Y
need to know. (Why would the Android Gradle Plugin simply fail when there's only 1 NDK on the system and it is the one I intend for it to use?) I publish this library to an internal Maven reposi
project is just a test app. So all of those downstream apps need the .so's and shouldn't have to know to spell out a specific NDK to use the AAR!

Producing a small project that reproduces this error is a non-trivial exercise to say the least.

I am *guessing* the central issue is actually the nuances of our CMake. But I am not sure which nuances -- or if that's the key part.

em...@google.com <em...@google.com> #19

When you use Conan, do you have a preamble like this:

```
include(${CMAKE_CURRENT_SOURCE_DIR}/../../../conan_build/conanbuildinfo.cmake)
conan_basic_setup(TARGETS)
```

which seems quite typical. I noticed that this Conan setup call invokes `conan_output_dirs_setup`, which has this section:

```
set(CMAKE_LIBRARY_OUTPUT_DIRECTORY ${CMAKE_CURRENT_BINARY_DIR}/lib)
set(CMAKE_LIBRARY_OUTPUT_DIRECTORY_RELEASE ${CMAKE_LIBRARY_OUTPUT_DIRECTORY})
set(CMAKE_LIBRARY_OUTPUT_DIRECTORY_RELWITHDEBINFO ${CMAKE_LIBRARY_OUTPUT_DIRECTORY})
set(CMAKE_LIBRARY_OUTPUT_DIRECTORY_MINSIZEREL ${CMAKE_LIBRARY_OUTPUT_DIRECTORY})
set(CMAKE_LIBRARY_OUTPUT_DIRECTORY_DEBUG ${CMAKE_LIBRARY_OUTPUT_DIRECTORY})
```

Only the first line actually matters. That line ends up confusing the debugger, making it unable to find the symbol files. Commenting out these five lines was enough for the debugger to pick u

I'll further investigate the interaction between this cmake flag and the debugger.

+++

Note to myself: For clean, deterministic repro, do a full "Clean Project + Refresh Linked C++ Projects" between edits to `conanbuildinfo.cmake`.

+++

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

Continuing from comment#19:

The problem is actually quite simple. The native debugger assumes that all symbol directories end with `/${ABI}/`, but the path used by Conan overrides this and puts them at `${module}/.c`
skips the symbol file.

em...@google.com <em...@google.com> #21

And, one workaround on the `CMakeLists.txt` side is to change the setup call to:

```
conan_basic_setup(TARGETS NO_OUTPUT_DIRS)
```

je...@gmail.com <je...@gmail.com> #22

Yes, we have

```
...
conan_basic_setup(TARGETS)
```

in our top-level CMakeLists.txt.

So I'll try changing this to `conan_basic_setup(TARGETS NO_OUTPUTDIRS)` and seeing how things work in 4.1 Canary 5 with corresponding Gradle Plugin.

(This all did work fine with v3.5.3 of the plugin...)

je...@gmail.com <je...@gmail.com> #23

So I tried that with 4.1c5 -- to no avail. I still don't see the C++ sources from my object libraries in my Android Project tree.

em...@google.com <em...@google.com> #24

Reassigned to tg...@google.com.

The fix I mentioned in comment#21 was to make sure the debugger can find your debug symbols. I think I mixed up this bug and b/139933953. I'll cross-post there to explain this.

The issue about object libraries in the Android view: I'll assign this to my teammate, I think he already is working on a fix for that, and he can update you.

tg...@google.com <tg...@google.com> #25

Marked as fixed.

Hi, the fix for object library in Android View will be in Android Studio 4.1 Canary 7. Please reopen this bug if you still have further issues. Thank you!