jniLibs merging does not catch dissimilar libraries with identical names

Assigned Bug P2 + Add Hotlist adexe s nau

STATUS UPDATE No update yet. Edit

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

Test case: https://github.com/DanAlbert/JniLibsErrorsTest

Identically named jniLibs from different modules with different contents current cause one library to clobber the other. This is unlikely to happen for most libraries, but could be a real problem for

✓ Mentioned issues (1) ✓ Links (4)

Mentioned issues (1)

P1 Add pickFrom to JniLibsPackagingOptions and ResourcesPackagingOptions "I filed Issue 214395989 to track the feature request in comment #23"

Links (4)

"Test case: https://github.com/DanAlbert/JniLibsErrorsTest"
"PS - here's a link to the current logic: https://android.googlesource.com/platform/tools/base/+/mirror-goog-studio-master-dev/build-system/gradle-core/src/main/java/com/android/build/gradle/int
"FYI, it's already helping users avoid this bug :) https://stackoverflow.com/q/58969292/632035"
" ...lking about is a Java interface that just has a native implementation, right? Not a C++ interface? I'm assuming that's the case, since the mechanisms for the latter shouldn't create this problem. Ass

COMMENTS

xa...@google.com <xa...@google.com>

Reassigned to sp...@google.com.

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

Currently, .so files (and other java resources) from the *app* module always take precedence over any with the same name from any *library* modules. This has been the case for as long as

I think it makes sense to change the logic to throw an error in this case. I'm just curious about why the current logic was intentionally implemented in the first place.

Should we also throw an error for java resources besides .so files?

I expect changing the logic for .so files will break some people's builds, and I expect changing the logic for non-.so files will break many more people's builds :)

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

PS - here's a link to the current logic: https://android.googlesource.com/platform/tools/base/+/mirror-goog-studio-master-dev/build-system/gradle-core/src/main/java/com/android/build/g

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

> I expect changing the logic for .so files will break some people's builds

If we're checking that it only causes an error if the libraries are not identical, we can reduce the number of false positives. In practice we'll still have some false positives where the app would

Maybe the best option is to add the error, but provide an option to downgrade it to a warning? Help people avoid surprises, but let them continue as they have been if it's not causing problems

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

I think this should eventually be an error for .so files and java resources.

I'm inclined to first make it a warning so that we don't suddenly break users, and then make it an error in a later version of AGP.

WDYT?

da...@google.com <da...@google.com> #6

SGTM. I think it's still a good idea to include the escape hatch to downgrade the error when we get to the point of making it an error. There are a lot of reasons why the user might need to upd

sp...@google.com <sp...@google.com> #7

The warning's been merged.

I'm going to leave this bug open (with decreased priority) until it's also converted into an error.

I'm pasting an excerpt of the warning below so that users who hit it and google it will find this thread if they have use cases that need to be supported.

"This version of the Android Gradle Plugin chooses the file from the app or dynamic-feature module, but this can cause unexpected behavior or errors at runtime. Future versions of the Andro

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

FYI, it's already helping users avoid this bug :) https://stackoverflow.com/q/58969292/632035

Given that this surfaced on SO, one UX improvement might be to list the sources of the duplicates ("modules foo and bar both attempt to package libfoo.so" or something similar).

Message last modified on Nov 22, 2019 06:21AM

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

As a company that develops many native components, with complex interdependencies, which are consumed by our various app teams, we have had many issues with .so file collisions in ap
up with various hacks and work arounds to make it all work together. The problem we are facing now, is that our customers who intake our various SDK modules are having similar issues wit

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

Re: Comment #9, Can you elaborate on the cases when your customers are getting the incorrect versions in their apps? Is it a case of the app itself using a different version or a case of the a

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

The latter. The customer's app has another dependency using the NDK, and there is no way to control which C++ stdlib version gets packaged.

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

danalbert@, what is the desired behavior in this scenario? To package the latest version or to throw an error unless the versions match?

kr...@gmail.com <kr...@gmail.com> #13

If you could add an action to `packagingOptions` to take the latest version, I believe that would address all of our use cases. Something along the lines of `pickLatestVersion`. However, is t

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

There's no way to know which one is newest, and selecting the latest does not guarantee that they are compatible. If you had an app using r10, selecting anything later from your package wo
someone actually look to see where the conflict is coming from and make a decision on how to resolve it.

The AAR you're talking about is a Java *interface* that just has a native *implementation*, right? Not a C++ interface? I'm assuming that's the case, since the mechanisms for the latter shouldn't

Possibly related, did anyone ever get around to making sure this warning doesn't fire if the two artifacts are actually bitwise identical? I know I'd discussed that with *someone* elsewhere, but

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

Ok, based on #14, looks like an error is the correct behavior. It's currently a warning, but the plan is to convert to an error in the future.

> Possibly related, did anyone ever get around to making sure this warning doesn't fire if the two artifacts are actually bitwise identical?

You mentioned that in comment#4 above, and I agree that it's a good idea. I'll plan to do that before converting to an error.

> The AAR you're talking about is a Java interface that just has a native implementation, right? Not a C++ interface? I'm assuming that's the case, since the mechanisms for the latter shouldr

kr...@, please let me know if that workaround works for you.

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

> > Possibly related, did anyone ever get around to making sure this warning doesn't fire if the two artifacts are actually bitwise identical?
>
> You mentioned that in comment#4 above, and I agree that it's a good idea. I'll plan to do that before converting to an error.

Ah, missed that. Never mind :)

kr...@gmail.com <kr...@gmail.com> #17

In my experience, this already generates an error. If during app packaging, 2 native libraries with the same name exist in 2 different dependencies (AAR files), then an error will be generated a

This is not sufficient for our use cases. We may have, for example, 5 different dependencies (AAR files) consumed through Maven, with a different mix of native libraries included in each. On

da...@google.com <da...@google.com> #18

> One of those dependencies needs to be designated as containing the correct version of each native library. Instead all we can do is have one picked at random using pickFirst.

It wouldn't be picked at *random*, someone needs to decide which one is going to work for their app, or eliminate incompatible dependencies. I guess I don't understand how that could be don

Is https://android.googlesource.com/platform/ndk/+/master/docs/user/middleware_vendors.md#for-java-middleware-with-jni-libraries not an option for you? If so, why not?

Message last modified on Aug 19, 2020 07:18AM

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

> In my experience, this already generates an error. If during app packaging, 2 native libraries with the same name exist in 2 different dependencies (AAR files), then an error will be generated

Ah yes, that's right. It's a warning only for the case when there's a conflict between an app's native library and a dependent android library's native library.

kr...@gmail.com <kr...@gmail.com> #20

We need to be able to specify which Android archive to pick each native library from. I am hopeful that AGP 4.1 with prefab publishing will emerge as a solution that allows us to strip all nativ

The actual scenario is that we have multiple different modules, each with a native component. They can be consumed in various different combinations, but there are dependencies between

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

In that example, does adding `packagingOptions.exclude 'libfoo.so'` to BarModule solve the issue?

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

It does. That means FooModule must be included as a dependency at the app level, even if it would not otherwise be necessary (maybe the Android pieces aren't required, just the native libfo

xa...@google.com <xa...@google.com> #23

It's better to make that selection at the consuming site in case modules are consumed by different apps. Also we need this for external depenencies anyway.

We should do something like

```
packagingOptions {
   pickFrom('libfoo.so', ':barModule') // for sub-modules
   pickFrom('libfoo.so', 'groupId:artifactId:version') // for external dependencies.
}
```

kr...@gmail.com <kr...@gmail.com> #24

That would be awesome!

sh...@gmail.com <sh...@gmail.com> #25

Hi, Any updates on the requirement specified in comment #23?

sp...@google.com <sp...@google.com> #26

I filed Issue 214395989 to track the feature request in comment #23