


Assigned

Feature Request

P1


+

Reviewed (L1)

 STATUS UPDATE

No update yet.

Edit

 DESCRIPTION

to...@gmail.com created issue [#1](#)

Gradle version: 4.0-milestone-1

Android Plugin Version: 3.0.0-alpha1

Module Compile Sdk Version: n/a

Module Build Tools Version: n/a

Android SDK Tools version: n/a

This issue (or actually a feature request) is for those of us that need to create library projects that produce AAR library as the output.

Please provide a way to create an AAR output that has configured dependencies built in.

There's apparently a lot of Android projects that need this, and so far there has not been an official way to do this with Android Gradle plugin.

There are some workarounds such as Android Fat AAR:
<https://github.com/adwiv/android-fat-aar>

Still, since this is a very basic level feature I think this should be covered by core Android tools.


For example a library project that uses three sub projects that should be all bundled into one AAR output could be declared like this ("api" keyword is just suggestion, it could be also "bundle", "inc fit):

```
apply plugin: 'com.android.library'

android {
    // Normal project definitions
}

dependencies {
    api project(":subproject1")
    api project(":subproject2")
    api project(":subproject3")
}
```

✓ Links (16)

 Links (16)

"<https://github.com/adwiv/android-fat-aar>"

"<https://stackoverflow.com/questions/40405465/android-studio-create-aar-using-another-aar-file-and-jar-i...> "

"<https://stackoverflow.com/questions/41994462/creating-fat-aar-file...> "


"<https://stackoverflow.com/questions/38988416/creating-aar-of-android-library-containing-jar-and-...> "

"<https://stackoverflow.com/questions/37270905/build-a-fat-aar-which-will-include-other-sub-modul...> "

See all related links


COMMENTS

All comments

 to...@gmail.com


<to...@gmail.com> [#2](#)

Examples of users requesting this feature in Stack Overflow:
<https://stackoverflow.com/questions/40405465/android-studio-create-aar-using-another-aar-file-and-jar-inside/40522657#40522657>
<https://stackoverflow.com/questions/41994462/creating-fat-aar-file-in-android-studio>
<https://stackoverflow.com/questions/38988416/creating-aar-of-android-library-containing-jar-and-aar/40522706#40522706>
<https://stackoverflow.com/questions/37270905/build-a-fat-aar-which-will-include-other-sub-modules-as-jar-aar-using-gradle>

 uc...@google.com

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

Thank you for suggesting this enhancement. We value the feedback from our community and hope to review your suggestion in an upcoming sprint.

 wa...@gmail.com

<wa...@gmail.com> [#4](#)

Hello, any update on this?

I am also waiting for this, and till now was using <https://github.com/adwiv/android-fat-aar>, as mentioned in original issue. But this tend to break with each plugin update. So can you please g we expect it?

The above mentioned script breaks again with new android plugin.
androidGradlePlugin: '3.0.0-beta6',
buildTools : '26.0.1',
distributionUrl=https\://services.gradle.org/distributions/gradle-4.1-all.zip

to...@gmail.com <to...@gmail.com> [#5](#)

Now that Android Gradle plugin 3.0.0 has been out for some time, its time to sync the status of this issue.
The Android Fat AAR plugin has not been updated and it seems to be incompatible with Android Gradle plugin 3.0.0.
Now "api" dependencies are supported but they behave like the old "compile" dependencies in library projects.

We still need a way to force subproject libraries to be bundled into using library project output AAR.
Are there any news on getting this officially supported in Android Gradle plugin?

pr...@gmail.com <pr...@gmail.com> [#6](#)

Same here, because of no official support still stuck with gradle plugin version 2.3.3

[Deleted User] <[Deleted User]> [#7](#)

our use case is we'd want to share an AAR w/ another project, but the AAR is composed of multiple submodules. Workaround is to share each submodule, but that's not as easy for users to artifact.

ss...@gmail.com <ss...@gmail.com> [#8](#)

We have 6 modules that we want to proguard in order to create a release. Some of the modules have inter-dependencies.

Because of the inter-dependencies, there is no way to Proguard the entire project at once. If we proguard each module individually, then we are getting class not found errors at runtime for s

ss...@gmail.com <ss...@gmail.com> [#9](#)

Additionally, we would also have the option to generate a single AAR file from multiple android library modules to provide to clients to avoid confusion when providing our SDK. This is not po
current build tooling. We have to generate all of the modules independently, and provide multiple AAR files to a client.

da...@capitalone.com <da...@capitalone.com> [#10](#)

+1 I agree we need an easy way to Proguard a multi-module project and produce a single artifact for others to consume. It's particularly important for scenarios like producing SDKs.

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

Thanks all for the comments. We will take this for consideration for 3.3. It's too late for 3.2 at this time unfortunately.

wa...@gmail.com <wa...@gmail.com> [#12](#)

When can we expect 3.3?

ma...@gmail.com <ma...@gmail.com> [#13](#)

I was able to get this working in one of our libraries with multiple AARs doing this, using the maven publishing plugin for gradle.

```
pom.withXml {  
    def dependenciesNode = asNode().appendNode('dependencies')  
  
    configurations.implementation.allDependencies.each {  
        def dependencyNode = dependenciesNode.appendNode('dependency')  
        dependencyNode.appendNode('groupId', it.group)  
        dependencyNode.appendNode('artifactId', it.name)  
  
        if(it.group == 'com.mattruno.android') {  
            dependencyNode.appendNode('version', rootProject.localVersion())  
        } else {  
            dependencyNode.appendNode('version', it.version)  
        }  
    }  
}
```

The trick here is that I had to "manually" configure the versions of the POMs based on the version from my project. By overwriting everything this way, I was able to get it to work.

localVersion() is...

```
def localVersion() {  
    isSnapshot() == "true" ? "${rootProject.gitLastTag()}-SNAPSHOT" : "${rootProject.gitLastTagWithCommitCount()}-${rootProject.gitSha(8)}"  
}
```

ma...@gmail.com <ma...@gmail.com> [#14](#)

I was able to get this working in one of our libraries with multiple AARs doing this, using the maven publishing plugin for gradle.

```
pom.withXml {
    def dependenciesNode = asNode().appendNode('dependencies')

    configurations.implementation.allDependencies.each {
        def dependencyNode = dependenciesNode.appendNode('dependency')
        dependencyNode.appendNode('groupId', it.group)
        dependencyNode.appendNode('artifactId', it.name)

        if(it.group == 'com.matttruno.android') {
            dependencyNode.appendNode('version', rootProject.localVersion())
        } else {
            dependencyNode.appendNode('version', it.version)
        }
    }
}
```

The trick here is that I had to "manually" configure the versions of the POMs based on the version from my project. By overwriting everything this way, I was able to get it to work.

localVersion() is...

```
def localVersion() {
    isSnapshot() == "true" ? "${rootProject.gitLastTag()}-SNAPSHOT" : "${rootProject.gitLastTagWithCommitCount()}-${rootProject.gitSha(8)}"
}
```

pr...@gmail.com <pr...@gmail.com> [#15](#)

Can you please post a sample on github with complete code? Thanks a lot..

ss...@gmail.com <ss...@gmail.com> [#16](#)

I wrote a blog post detailing why this is important: "Why We Need "fat" AARs for Android Libraries" <https://handstandsam.com/2018/07/13/why-we-need-fat-aars-for-android-libraries/> I hope also attached a GIF that attempts to visualize how this is different than current methods.

 **fat-aar.gif**
463 KB [View](#) [Download](#)

je...@jedrivisser.com <je...@jedrivisser.com> [#17](#)

This works for me in some cases, so hopefully it is useful to some other people as well: <https://medium.com/@jedri/creating-aars-with-private-dependencies-b17de89c6b7c>

ss...@gmail.com <ss...@gmail.com> [#18](#)

Thanks for sharing, but the solution you provided is limited to code only and not any Android resources, Manifest and other AAR specific things.

Also, it doesn't allow you to obfuscate the code all at once because the Jar is precompiled.

It does solve the case of zipping multiple precompiled Jars into a single AAR file.

ur...@gmail.com <ur...@gmail.com> [#19](#)

+1 , Waiting for tooling support to publish SDK along-with internal dependencies as a single fat-aar.

ra...@gtempaccount.com <ra...@gtempaccount.com> [#20](#)

Any updates for this issue? Is it in any roadmap? It would be nice to have a feedback because this is becoming critical to our solution as it grows. So we are considering an in-house plugin implementation but we would prefer an official solution, if it is a viable date.

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

It is something we are considering but it is not something we are actively working on right now.

I would be surprised if it made it into 3.3 at this point :(

ma...@gmail.com <ma...@gmail.com> [#22](#)

Urgently needed this feature

ka...@waltzapp.com <ka...@waltzapp.com> [#23](#)

+1. Such a great feature !

om...@gmail.com <om...@gmail.com> [#24](#)

	+1 ! We really needed it for our SDK as well!	/
li...@163.com <li...@163.com> #25	+1!!!	/
li...@gmail.com <li...@gmail.com> #26	+1 ! Urgently needed this feature !	/
it...@gmail.com <it...@gmail.com> #27	+1 Also need this feature or a work around	/
ah...@gmail.com <ah...@gmail.com> #28	+1!!!!	/
[Deleted User] <[Deleted User]> #29	Any chance we could hack apk packaging mechanism to achive the goal? We can easily make an apk from any AARs we want, and I might be wrong on this one but since apk and aar look sir dexed classes, we could try hooking into apk build process and somehow avoid dexing or something? Could that work, at least for some cases?	/
li...@mingchao.com <li...@mingchao.com> #30	+1!!!!	/
sa...@gmail.com <sa...@gmail.com> #31	+1 on this, we're making SDKs and it would be great if we could package multiple modules in one SDK that could then be optimized with proguard/R8. Otherwise we need mono-module (ugh) kotlin doesn't support package protected, it's easy for people to introduce dependencies where there shouldn't be any.	
to...@gmail.com <to...@gmail.com> #32	Google, please take this feature request into account and implement this into the next major Android Gradle Plugin version (3.3.0). Pretty much every Android developer who wants to create and offer an SDK and use android library modules to arrange code & architecture will bump into this same issue. There's already 135 fellows that have starred this issue and there are bound to be lots more out there who would really benefit from this.	§
fx...@gmail.com <fx...@gmail.com> #33	Any Android SDK developer has been forced to avoid/go-around/re-arrange/hack its code to achieve the common task of modularised and pack code. +1	§
zh...@gmail.com <zh...@gmail.com> #34	+1! We have to use a self-define merger gradle task which try to copy source code files, jni libraries and more to include submodules into release aar. I believe many other SDK developers would benefit from this feature.	§
ma...@gmail.com <ma...@gmail.com> #35	@zh...@gmail.com Is it possible for you to share what you have done with the community? :-) Thanks.	§
gm...@163.com <gm...@163.com> #36	+1!!!! Thanks.	
ss...@gmail.com <ss...@gmail.com> #37	We've been trying to manually come up with a workaround for this. We've hacked together a Python script that hides our ~10 modules out of settings.gradle, and adds a new, single "merged" single "merged" library module has sourceSets for each existing module so that all the source code appears to just be owned by this new module. We then have to run a script to manually r collisions like strings.xml and AndroidManifest.xml by doing some XML Parsing hacks. We also have to do a find/replace for our all of our `com.myapp.R` references, and replace it with our `com.myapp.merged.R` version so that the resources can be located. Then finally we can run Proguard on all of it. This is super fragile, and really hard to edit our existing code. We've been it really blows up your local workspace. Any other tips? We have tried looking at the existing Fat AAR plugins on GitHub, but a lot of the Gradle Tasks are no longer available. This method is hack. I'd love to see some more guidance/advice if the tooling is not going to be available soon.	
li...@gmail.com <li...@gmail.com> #38		

<div></div>	<div>ku...@gmail.com <ku...@gmail.com> #38</div> <div><div>+1!!!!</div><div>Thanks.</div></div>	
<div></div>	<div>to...@gmail.com <to...@gmail.com> #39</div> <div><div></div><div>Dear Google. Please try to prioritize this since this is a blocker for pretty much all SDK and/or library type of developers. First step would be to support bundling of Java code only (i.e. resource file & manifest merging could be skipped and conflicts could just throw a Gradle error/exception). This should be an e would nicely allow hierarchical library & SDK code development.</div></div>	
<div></div>	<div>om...@gmail.com <om...@gmail.com> #40</div> <div><div></div><div>@xa...@google.com any update? We are still relying on the unofficial Fat-AAR library and this doesn't allow us to upgrade our gradle version due to compatibility issues</div></div>	
<div></div>	<div>ma...@gmail.com <ma...@gmail.com> #41</div> <div><div></div><div><div>+1</div><div>We think this feature is very useful to reuse modules across different projects.</div></div></div>	<div></div>
<div></div>	<div>ra...@gmail.com <ra...@gmail.com> #42</div> <div><div></div><div><div>+1!!!</div><div>We really need this feature in out project. Unofficial lib fat-aar does not support new gradle versions.</div></div><div>Message last modified on Nov 21, 2018 12:42PM</div></div>	<div></div>
<div></div>	<div>[Deleted User] <[Deleted User]> #43</div> <div><div></div><div><div>+1!</div><div>We really needed it too.</div></div></div>	<div></div>
<div></div>	<div>dm...@gmail.com <dm...@gmail.com> #44</div> <div><div></div><div><div>We can't make aar library written in kotlin and use it in java app without import kotlin library (ノ 益 ㊦)ノ 彡 ㊦</div><div>Our team voted for this feature</div><div>Message last modified on Nov 30, 2018 10:44PM</div></div></div>	<div></div>
<div></div>	<div>bi...@outlook.com <bi...@outlook.com> #45</div> <div><div></div><div><div>+1000!!</div><div>We need this feature urgently :(</div></div></div>	<div></div>
<div></div>	<div>ja...@gmail.com <ja...@gmail.com> #46</div> <div><div></div><div><div>+111111111</div><div>We need this feature desperately!</div></div></div>	<div></div>
<div></div>	<div>ss...@gmail.com <ss...@gmail.com> #47</div> <div><div></div><div><div>From my understanding, this issue is to be able to create a single AAR from multiple sub-modules source code. If you have "model", "network", "interface", and "impl" modules, you want to ex the end.</div><div>For your "bundling kotlin with AAR" use case, you can include the kotlin JAR file in a `libs` directory and this will work. Bundling JARs in an AAR has been supported. You cannot bundle AARs: good practice though, if you don't own the source code, I wouldn't bundle it in and just give the gradle dependencies to the apps that use your SDK. This is important because you will run into use Kotlin and want to upgrade the version version of Kotlin for an unrelated reason. In this situation, there will be two versions of kotlin at runtime causing errors.</div></div></div>	
<div></div>	<div>dm...@gmail.com <dm...@gmail.com> #48</div> <div><div></div><div><div>#47 Thank you! This is what I needed. However fat-aar is also a very desired feature.</div></div></div>	
<div></div>	<div>[Deleted User] <[Deleted User]> #49</div> <div><div></div><div><div>+1</div><div>We require this feature since we need to bundle a third party sdk (from a private maven) with our sdk. While we have a license agreement with this third party and sufficient DRM, there's no their library without bundling it. We have been stuck using an extremely old version of gradle which is the last version that the fat aar script supported well.</div></div></div>	<div></div>
<div></div>	<div>fn...@163.com <fn...@163.com> #50</div> <div><div></div><div><div>+10086 !!!!!!!!!!!!!</div></div></div>	<div></div>



de...@gmail.com <de...@gmail.com> [#51](#)

+1 We are eagerly waiting for this fix. As we are a SDK developer, our clients are also a SDK developers, they are unable to add our AAR and asking their clients to add our AAR to their app. If we are the most happier than other.. Plz Google.

Message last modified on Jan 3, 2019 06:58PM



ep...@gmail.com <ep...@gmail.com> [#52](#)

+1 ! Urgently needed this feature !



[Deleted User] <[Deleted User]> [#53](#)

This might help some of you: <https://github.com/Mobbeel/fataar-gradle-plugin>



li...@gmail.com <li...@gmail.com> [#54](#)

Is there any progress, please?

Message last modified on Jan 5, 2019 05:48PM



ze...@gmail.com <ze...@gmail.com> [#55](#)

+1 Yes, we need an official solution.



yu...@gmail.com <yu...@gmail.com> [#56](#)

+1 we need this feature!



ma...@gmail.com <ma...@gmail.com> [#57](#)

Message to up-coming people that are willing to comment on this issue:

Please, stop polluting this thread by adding "+1", "we need this feature", "Urgently need this", ...

You have a star button on the top left corner that help Google see how important this issue/feature is important. So ensure the star is enabled (yellow) and stop commenting with non-helping. Actually, those message will have the complete reverse effect, where people will get sick of receiving email that are unhelping, and will unstar from this thread, decreasing from Google eyes, issue.

Btw, thanks [to...@indoo.rs](#) for your [comment #53](#).

Thanks.

Message last modified on Jan 12, 2019 02:55AM



dm...@gmail.com <dm...@gmail.com> [#58](#)

We develop a library which is divided into sub-modules internally. The library is not public so it's being distributed by zip files with sdk which includes single fat aar file. We had to modify fata to combine our modules into single file.

Fat aar in build tools will be very helpful as it makes no sense to provide details of internal modules division to the clients. You have also no way to provide the list of dependencies if you are not

Thanks!



nx...@gmail.com <nx...@gmail.com> [#59](#)

+1



to...@gmail.com <to...@gmail.com> [#60](#)

Dear Google. Please check whether you could actually implement this.
There's already 215 stars in this issue which means loads of developers really need this feature.
If you start now, this could be implemented before two years has passed from when I filed this issue :)



mi...@gmail.com <mi...@gmail.com> [#61](#)

I'm being faced with the same problem. Having tried unofficial solutions for several weeks, I can claim there's no solid way to implement a single bundled aar. We need it, Google!



al...@gmail.com <al...@gmail.com> [#62](#)

I know this kind of reply is frowned upon ...

... But I want to show my support for this feature request. Please prioritize this Google (or perhaps Gradle?)! :-)

ra...@gmail.com <ra...@gmail.com> [#63](#)

+1 ! Urgently needed this feature !

se...@gmail.com <se...@gmail.com> [#64](#)

Wish I am still alive when official support comes

Message last modified on Apr 15, 2019 08:03PM

zs...@gmail.com <zs...@gmail.com> [#65](#)

It can cost me my job :(I made a lot of modules on updated react-native that uses build tool 3.3.0 and now our library is not published for 1 week. Any time lines on this?

lo...@gmail.com <lo...@gmail.com> [#66](#)

[@comment#65](#)

You better find another solution like fat AAR library, or find another job. There's seemingly no progress on that issue, only people complaining.

[Deleted User] <[Deleted User]> [#67](#)

Start merging those modules ASAP, nobody is working on this, nothing changed in 2 years since issue was opened.

[Deleted User] <[Deleted User]> [#68](#)

I don't know if this can work for all cases but we were in the same situation and we ended up using this solution: <https://stackoverflow.com/a/55650560/2910520>
With this we ended up having our .aar library compiled and bundled with other 4 different aars (all in the same package).

Hope it will work for some of you without manifest merging strategies and similar.

an...@gmail.com <an...@gmail.com> [#69](#)

Looking forward

an...@gmail.com <an...@gmail.com> [#70](#)

Any updates on this?

cl...@gmail.com <cl...@gmail.com> [#71](#)

Just stumbled upon <https://github.com/kezong/fat-aar-android> It follows the, in my opinion preferable, approach of <https://github.com/Vigi0303/fat-aar-plugin> to allow you just to explicit ma
embedded. Have not challenged deselecting transitive dependencies, manifest merging, obfuscation, etc., but at least it initially just bundled the libs up.

wa...@gmail.com <wa...@gmail.com> [#72](#)

+1

co...@gmail.com <co...@gmail.com> [#73](#)

+1!
Thanks.

xi...@gmail.com <xi...@gmail.com> [#74](#)

+100000!
Thanks

et...@gmail.com <et...@gmail.com> [#75](#)

This would be very useful for distributing SDKs that include other dependencies. The client would only have to include one dependency instead of many (5 in our case ;()

[Deleted User] <[Deleted User]> [#76](#)

^ If you're talking about maven dependencies and such, then that's horrible usecase and it break dependencies. If your SDK shares dependencies with application using it, build will fail due to have no option but to remove it's own explicit dependency and be locked into your version of hidden dependency.

[Deleted User] <[Deleted User]> [#77](#)

We could have a custom configuration for such type of dependencies which should be capable of repackaging the dependencies under another name. Something like this:

```
dependencies {  
    //this will not be bundled into the aar and the consumer will download the dependency from maven/jcenter etc... as it is now  
    implementation "com.thirdparty.dep:lib:1.0.0"  
    //this will be bundled into our lib and by default all the :my_lib dependencies will be bundled as they are into our aar lib.  
    //To prevent duplicate conflict with third party open source lib, you can specify a custom repackaging behaviour  
    bundleAar project(":my_lib") {  
        dep: "com.thirdparty.dep:lib2", repackagingAs: "com.mylib.lib2"  
    }  
}
```

In this way we would prevent conflict and duplicate classes error. The only thing is that the Consumer will have the same class name for different packages and will need to know which one it thinks it's still fine because it is something we already do every day when there is a conflict)

Message last modified on Jul 30, 2019 09:33PM

[Deleted User] <[Deleted User]> [#78](#)

That only introduces more questions and issues and would delay release of actually useful feature for a year or so.

If you have lots of dependencies, make an (empty) companion library with all your maven dependencies and publish it to online repo. That way there's no need to modify standard tooling and add just one dependency.

[Deleted User] <[Deleted User]> [#79](#)

Integrating app would have 1 line dependency in both cases and if you push third party dependencies on maven you still need to repackage them to prevent conflicts

[Deleted User] <[Deleted User]> [#80](#)

I didn't mean pushing actual libraries you depend on to maven, only symbolic/link dependencies as in your gradle files. So, a companion library without any code and dependencies listed as in gradle file.

ma...@gmail.com <ma...@gmail.com> [#81](#)

Hi,

Is there any update on this.

I also need to bundle multiple AAR files inside my library projects. These aar files are having third party property rights with whom we already have contract signed but we don't want to expose our clients. Our team is creating wrapper around them and deciding which library need to be called based on the requirement.

Due to FAT aar issue, we facing crashes. Resource files of the embedded AAR files are not getting merged and final aar file don't have all string resources of the embedded AAR files in it.

Message last modified on Aug 2, 2019 04:24PM

ra...@gmail.com <ra...@gmail.com> [#82](#)

Hello, teams.

Any updates about this feature request?

It would be awesome to bundle multiple AAR inside our SDK library. Due to we don't want to expose a few libraries dependencies as well inside our SDK library.

ra...@gmail.com <ra...@gmail.com> [#83](#)

For anyone have read Sam Edwards article.
<https://handstandsam.com/2018/07/13/why-we-need-fat-aars-for-android-libraries/>

Looks like kezon already done it to support gradle plugin version 3 above, here's the link:
<https://github.com/kezon/gradle-plugin/blob/master/src/main/java/com/kezon/android/gradle/plugin/AndroidLibraryPlugin.java>

But currently, doesn't support publish to maven for adding our dependencies.

no...@gmail.com <no...@gmail.com> [#84](#)

any update?

da...@gmail.com <da...@gmail.com> [#85](#)

Any updates about this feature request?

ka...@gmail.com <ka...@gmail.com> [#86](#)

We also need this feature since we are using different module for our SDK project.



zs...@plaid.com <zs...@plaid.com> [#87](#)

I'm also using the fat aar plugin by kezong but an official solution would be great



in...@gmail.com <in...@gmail.com> [#88](#)

+1



bu...@gmail.com <bu...@gmail.com> [#89](#)

+1



mi...@gmail.com <mi...@gmail.com> [#90](#)

+1



ad...@gmail.com <ad...@gmail.com> [#91](#)

any update?



ss...@gmail.com <ss...@gmail.com> [#92](#)

I talked to Chris Warrington on the AGP team and he said they are aware of this, but it still isn't prioritized compared to other features.

There is a Fat AAR plugin that works for most use cases, but isn't an official solution. Check it out here <https://github.com/kezong/fat-aar-android>



d....@infotech.team <d....@infotech.team> [#93](#)

I think it should be a priority with more and more people adopting modularization in their android repos. It's almost 5 years since AGP exists and there is still no native way to build something
External implementations will always break with updates and be suboptimal to AGP implementation(if it existed), since only people that are working on it know the internals and possible pit
For example: how does <https://github.com/kezong/fat-aar-android> work with composite builds? Multiple flavor dimensions? Does it caches everything that is cachable?



[Deleted User] <[Deleted User]> [#94](#)

^ Amen



ty...@gmail.com <ty...@gmail.com> [#95](#)

god, after five years passed...



aa...@gmail.com <aa...@gmail.com> [#96](#)

5 years and counting...



zs...@plaid.com <zs...@plaid.com> [#97](#)

Not sure how this is a P2, S2 with this being a topic at google io 2019 <https://www.youtube.com/watch?v=PZBg5DIzNww>



d....@infotech.team <d....@infotech.team> [#98](#)

@xa...@google.com If you guys are not plannig to implement it in 4.0, can you at least elaborate why?

AGP is already heavily granularizes all possible artifacts which are produced for building AAR. Why is it so hard to take the same logic that you already have for apk(and maybe bundles/feat artifacts?



xu...@gmail.com <xu...@gmail.com> [#99](#)

Any updates about this feature request?



xa...@google.com <xa...@google.com> [#100](#)

Building an APK is very different from merging AARs. The main difficulty is resource handling. In APKs, we package it, keep each library's R file (which gets re-created with the final ID of the

For AAR there are 2 main difficulties:
- We need to unify the package/application ID of the different AARs, as AAR can only have a single manifest and consumer will create a single R class for it. This means rewriting the compile libraries (this is not technically challenging but work that needs to happen).
- We need to rewrite the dependencies of the published AAR. If you merge everything this is easy, if you only merge your local modules and keep your external dependencies out this is more c

offer ways to do this but we have not looked into this yet.

This is not happening in 4.0

rj...@gmail.com <rj...@gmail.com> [#101](#)

When is this happening? With the new architecture practices and everyone doing modularization we are not able to bundle single aar. Due to this, I have to rollback whole modularization of lil request is not at priority. Will try to make sub modules in a single library (Bad approach)

es...@gmail.com <es...@gmail.com> [#102](#)

+1

Any updates?

ro...@gmail.com <ro...@gmail.com> [#103](#)

+1

I need this! please!!

va...@gmail.com <va...@gmail.com> [#104](#)

+1

er...@gmail.com <er...@gmail.com> [#105](#)

+1

ji...@gmail.com <ji...@gmail.com> [#106](#)

+1

Google team, please add this feature.

ch...@gmail.com <ch...@gmail.com> [#107](#)

+1

oa...@gmail.com <oa...@gmail.com> [#108](#)

+1

ka...@gmail.com <ka...@gmail.com> [#109](#)

+1

lo...@gmail.com <lo...@gmail.com> [#110](#)

+1

lo...@gmail.com <lo...@gmail.com> [#111](#)

+10000

ye...@gmail.com <ye...@gmail.com> [#112](#)

+1

ok...@gmail.com <ok...@gmail.com> [#113](#)

It seems that there is a bit of a disconnect between how android libraries are developed at Google and how proprietary libraries are developed in the wild. It's super common that proprietary libraries are published to a public maven repository (that would basically circumvent this problem altogether). So every time a proprietary library becomes non-trivial and is modularized, people run into the sad levels of desperation in the comments of this issue.

I'm aware this is not a trivial issue. I'm just hoping the the priority of this issue would be re-evaluated.

ok...@gmail.com <ok...@gmail.com> [#114](#)

Also it is very unintuitive that currently the dependencies work differently depending on whether library is a precompiled AAR dependency or a subproject in the same gradle project.

For example: I have the dependency chain: 'app' -> 'lib_a' -> 'lib_b'

If all the projects are subprojects in the same gradle project it's enough for app to say

```
dependencies {  
    implementation project(':lib_a')  
}
```

and also resources from lib_b will be included in the APK.

However, if I use the AAR generated for lib_a and make an app that uses it in a similar way (e.g. using new module -> import JAR/AAR Package)

From the app point of view the dependency looks exactly the same:

```
dependencies {  
    implementation project(':lib_a')  
}
```

but resources from lib_b are no longer included in the APK as they are not part of the AAR generated by lib_a



ag...@skipit.com <ag...@skipit.com> [#115](#)

I completely agree with you #114. In one case, the only solution that worked for me was to unpack the .aar to a .jar, and then include the .jar in my gradle file. I had to add proguard rules to exclude from the final build. Using this method, I'm able to include additional libraries in my final output, but this is not ideal and could potentially fail.



[Deleted User] <[Deleted User]> [#116](#)

+1



ra...@gmail.com <ra...@gmail.com> [#117](#)

+1



as...@hoyosintegrity.com <as...@hoyosintegrity.com> [#118](#)

+1



p....@criteo.com <p....@criteo.com> [#119](#)

+1



ba...@gmail.com <ba...@gmail.com> [#120](#)

+1



ao...@gmail.com <ao...@gmail.com> [#121](#)

+1



ma...@gmail.com <ma...@gmail.com> [#122](#)

+1



oz...@gmail.com <oz...@gmail.com> [#123](#)

+1



va...@gmail.com <va...@gmail.com> [#124](#)

+1



kr...@gmail.com <kr...@gmail.com> [#125](#)

+1
(pity, apparently so many people in need of this super important feature, and no luck)



eg...@gmail.com <eg...@gmail.com> [#126](#)

+1



dm...@quix.com <dm...@quix.com> [#127](#)

<div><div></div><div>um...@wix.com <um...@wix.com> #127</div></div>	<div>+1</div>
<div><div></div><div>pr...@gmail.com <pr...@gmail.com> #128</div></div>	<div>+1</div>
<div><div></div><div>ko...@gmail.com <ko...@gmail.com> #129</div></div>	<div>+1</div>
<div><div></div><div>se...@demant.com <se...@demant.com> #130</div></div>	<div>+1</div>
<div><div></div><div>am...@gini-apps.com <am...@gini-apps.com> #131</div></div>	<div>+1</div>
<div><div></div><div>at...@gmail.com <at...@gmail.com> #132</div></div>	<div>+1</div>
<div><div></div><div>ma...@gmail.com <ma...@gmail.com> #133</div></div> <div><div>DO NOT WRITE "+1" COMMENTS</div><div>It sends emails to hundreds of people. If you want to bring attention to this issue, just click the star icon at the top. If you spam the others with unhelpful comments, they might unstar this issue and it will never be resolved.</div></div>	
<div><div></div><div>ga...@google.com <ga...@google.com> #134</div></div> <div><div>At the moment we are talking with Gradle about being able to merge dependencies from multiple projects (that are being bundled), as that part is a problem in the JVM ecosystem as well. On the Android-specific features (Android resources, manifest etc.).</div></div>	
<div><div></div><div>ne...@gmail.com <ne...@gmail.com> #135</div></div>	<div>+1</div>
<div><div></div><div>ga...@google.com <ga...@google.com></div><div>Assigned to bi...@google.com.</div></div>	
<div><div></div><div>[Deleted User] <[Deleted User]> #136</div></div>	<div>+1</div>
<div><div></div><div>pa...@gmail.com <pa...@gmail.com> #137</div></div> <div><div>Comment has been deleted.</div><div>Message last modified on Apr 27, 2021 05:35PM</div></div>	
<div><div></div><div>ch...@gmail.com <ch...@gmail.com> #138</div></div>	<div>+1</div>
<div><div></div><div>cm...@google.com <cm...@google.com> #139</div></div> <div><div>Please don't comment with +1 - as #comment133 states - use the star button.</div></div>	
<div><div></div><div>mu...@gmail.com <mu...@gmail.com> #140</div></div> <div><div>is there any update on this request from google?</div></div>	
<div><div></div><div>ww...@gmail.com <ww...@gmail.com> #141</div></div>	

nope... it sucks but it looks like this P1 priority is not a priority at this point in time.

ve...@gmail.com <ve...@gmail.com> [#142](#)

I'm leaving below a "workaround" solution that I put together for my particular project.

My main requirement was to ship a closed-source library that was obfuscated as much as possible, except for a very limited set of API methods.

The main idea is to replace each internal module with a corresponding sourceSet in the main library:

```
sourceSets {  
    getByName("main") {  
        java.srcDir("../module1/src/main/java")  
        java.srcDir("../module2/src/main/java")  
    }  
}
```

By using a relative path with `..`, I was able to version control `module1` and `module2` in their own git repository.

Note that this solution has to be expanded if your modules contribute additional dependencies, additional android resources, additional manifest entries, etc.

I realize this is hacky as hell, but it serves my purpose, while we wait for a proper fix...

wa...@bytedance.com <wa...@bytedance.com> [#143](#)

+1, while no response

al...@appodeal.com <al...@appodeal.com> [#144](#)

+ 1

si...@gmail.com <si...@gmail.com> [#145](#)

+1

cm...@google.com <cm...@google.com> [#146](#)

Please don't comment with +1 - as [#comment133](#) states - use the star button.

jr...@gmail.com <jr...@gmail.com> [#147](#)

+1

ww...@gmail.com <ww...@gmail.com> [#148](#)

This issue bugs me so much. Google for one reason (compilation speed) or another (code structure) is/was actively suggesting splitting logic into separate modules and creating multi-module packages. "package access modifier" people are actively pointing at a modules as an alternative.

But you can't even do it when writing a 3rd-party library because of this bug. You'll either end up with a huge number of nonsensical module dependencies that way or your code will not even

lu...@google.com <lu...@google.com> [#149](#)

↪ [www.matrix.dev](#)@ sorry for your frustration. This is clearly a significant painpoint for developers and we're now actively working on writing a plugin for easily bundling libraries modules. We know when you can expect a release and/or usable version in the alpha branch.

id...@gmail.com <id...@gmail.com> [#150](#)

The lack of this very fundamental modular construction feature causes so much aggravation across so many areas of Android development. How is it that such a basic issue was identified and yet here we are 5 years later with no timeline for a solution? It suggests that these priorities are just markers that serve no purpose other than to placate developers into not complaining demonstrates that clearly Google must have a significant list of priorities that far supersede those of actual developer needs. In the meantime, we'll all just swim around in Google's mire because choice but to endure Google's incompetence.

go...@jakewharton.com <go...@jakewharton.com> [#151](#)

They probably have to spend their time dealing with entitled assholes like you here on the issue tracker. I'm sure they're hiring, by the way, if you need it that badly.

je...@google.com <je...@google.com> [#152](#)

Reassigned to [je...@google.com](#).

As [#149](#) stated, we are actively working towards providing a solution.

- qi...@gmail.com** <qi...@gmail.com> [#153](#)

+1 star and the total number of stars hit 520, a special sign of love number which sounds like 我爱你(I Love You) in Chinese.
- ga...@ril.com** <ga...@ril.com> [#154](#)

+1
- vr...@ril.com** <vr...@ril.com> [#155](#)

+1
- mu...@ril.com** <mu...@ril.com> [#156](#)

+1
- pr...@ril.com** <pr...@ril.com> [#157](#)

+1
- ja...@gmail.com** <ja...@gmail.com> [#158](#)

Hello, do you have an update about the alpha release timing? 🙏
- vi...@mobills.com.br** <vi...@mobills.com.br> [#159](#)

+1
- pa...@mobills.com.br** <pa...@mobills.com.br> [#160](#)

+1
- il...@scandit.com** <il...@scandit.com> [#161](#)

+1

Message last modified on Jul 27, 2022 05:12PM
- cl...@portto.com** <cl...@portto.com> [#162](#)

+2
- kb...@gmail.com** <kb...@gmail.com> [#163](#)

Writing "+1" won't change anything, it just sends spam to people watching this issue. Please star the issue instead of sending "+1".
- un...@gmail.com** <un...@gmail.com> [#164](#)

it is 2022, is there an official way to do such thing?
- ko...@gmail.com** <ko...@gmail.com> [#165](#)

+100 our team's working on our sdk as a service, ones use our sdk need to import like thousands of aars to make things work, which is really unnecessary and annoying for them.
- de...@gmail.com** <de...@gmail.com> [#166](#)

Any update on this?
- mm...@gmail.com** <mm...@gmail.com> [#167](#)

Developing library, that has 2 artifacts, for Android Views system and Jetpack Compose UI.
Both of them use some shared modules, but I'm unable to pack them with artifacts.
Really need this feature
- e....@gmail.com** <e....@gmail.com> [#168](#)

Re: #167

If I used both Android Views and Compose with your library in my project I'd get duplicated classes errors because both your libraries contain the same common classes making it unusable. Producing an additional common AAR already solves this case. Publishing support improved so much in AGP 7.1+.

e....@gmail.com <e....@gmail.com> [#169](#)

Re: #165

You don't provide your SDK using maven coordinates? Copying even a single one AAR is what's annoying. You can use raw Amazon S3 as a maven repo.

go...@jakewharton.com <go...@jakewharton.com> [#170](#)

@ #168

With Java-based projects embedding local project dependencies inside your jar also affords the ability to repackaging them into separate package names thus avoiding that problem. Wanting to create a monolithic AAR from a suite of local project dependencies is one use case. Wanting to repackage a dependency (whether local or remote) into a new package is a separate, however, useful and related.

mr...@gmail.com <mr...@gmail.com> [#171](#)

Only combine local libs into single AAR is enough.

go...@jakewharton.com <go...@jakewharton.com> [#172](#)

I think you dropped a "for me".

vu...@gmail.com <vu...@gmail.com> [#173](#)

I have the issue that my library depended on another lib (example: lib_A) then after obfuscating my code throws an error 'NoClassDefFoundError', so let the application that uses the library aware of the library (lib_A), so I don't need to create something like fatAAR anymore.

But if your dependent library (lib_A) is private then you have to create fatAAR or open the dependent library (public lib_A)
good luck

be...@livevoice.io <be...@livevoice.io> [#174](#)

Is there any update on this?

#149 stated work on this was in progress almost a year ago, is this still true?

Can we expect *something* to happen anytime soon?

Message last modified on Apr 7, 2023 12:33AM

ga...@linecorp.com <ga...@linecorp.com> [#175](#)

It seems that [com.android.fused-library](#) is supported since Android Gradle Plugin 7.3

But when I applied this plugin, I got errors like this.

```
plugins {
    id "com.android.fused-library"
}

android {
    buildToolsVersion "33.0.2"
}

dependencies {
    include project(":xxx")
}
```

```
$ ./gradlew tasks
...
* What went wrong:
An exception occurred applying plugin request [id: 'com.android.fused-library']
> Failed to apply plugin 'com.android.internal.fused-library'.
    > Could not register artifact transform AarResourcesCompilerTransform (from {artifactType=android-exploded-aar} to {artifactType=android-compiled-dependencies}).
        > Service com.android.build.gradle.internal.services.Aapt2DaemonBuildService_fe3852e3-dd9f-49d6-ae5c-b2ea20b92660 is not registered.
```



xa...@google.com <xa...@google.com> [#176](#)

The `fused-library` plugin is a work in progress and not usable at the moment.



be...@livevoice.io <be...@livevoice.io> [#177](#)

Any info on when the `fused-library` *might* be available? A date far out would still be more helpful than being left in the dark



lu...@google.com <lu...@google.com> [#178](#)

Unfortunately, the team can't provide a solid release date for the release of the fused library. Although there was active development on the fused library in 2022, our team priorities changed development on hold. We are still committed to fused library especially given this is a highly requested feature and this feature request has been around since 2017.

Once development resumes, it may take up to 6 months for the plugin to land in a stable release.

I'll update this feature request, once we are able to continue work.



cy...@gmail.com <cy...@gmail.com> [#179](#)

Comment has been deleted.

Message last modified on Jun 2, 2023 11:55AM



iv...@gmail.com <iv...@gmail.com> [#180](#)

We would greatly appreciate any information regarding the potential resumption of development for the fused-library, as it is currently hindering our ability to upgrade AGP to version 8+. Your consideration.



ne...@gmail.com <ne...@gmail.com> [#181](#)

Hey google,

Most of us are eagerly looking at [this FusedLibraryPlugin](#)

Three questions:

1. You guys were able to create an entire IDE with Android flavour collaborating with JetBrains. How big is giving us a plugin collaborating with Gradle? If it is that difficult, it would help if you like it?
2. Date you can't provide, progress you won't reveal, task is not prioritised. You are forcing us not to update to latest AGP. Imagine the world stopped buying new Android phones and use/s you like it?
3. This feature is not requested by users to ignore, it is required by developers to make apps for you Google. Do you realize that?

I know asking these questions changes nothing, but asking feels right.

Anyone has any better ideas other than making your project into single module or publish all individually to maven?



ar...@gmail.com <ar...@gmail.com> [#182](#)

+1



pw...@gmail.com <pw...@gmail.com> [#183](#)

+1



su...@grabtaxi.com <su...@grabtaxi.com> [#184](#)

+1