

Near-term TODO

(Roughly in order)

Keystore management

DONE

- Change Keystore BUILD to take a config file instead of attributes.
 - allow sources (i.e. config file) outside of buildroot.
 - Best to leverage existing ConfigParser.
- Allow CLI plugging of credentials. See `credentials.py` and `netrc.py`.
- verify signature (`jarsigner -verify -verbose -certs my_application.apk`)
 - This would be a good fit in an integration test. (TODO)

release key notes:

- Storing in plaintext is not ideal. However git, Maven and Gradle also use cleartext config. Local access not in the threat model, as usual.
- Original issue: <https://github.com/pantsbuild/pants/issues/490>
- Design Doc for [Keystore handling here.](#)

ZipalignTask

DONE

- Should run only on release builds.
 - This means that I need to be careful when defining SignApk products.
- To do next after KeyResolver stuff.

Integrate Android SDK to CI

- Soon(!)
- Issue: <https://github.com/pantsbuild/pants/issues/937>

ManifestParser

DONE

- Create XmlParser
- robust to bad/missing info and malformed xml

Secure Config files in a non-volatile location

DONE

- So it won't get blown away with volatile state purges.

Android Library

DONE

- Create `android_library` target

- no target_sdk for libraries
- 3 types of library
 - .aar
 - ~~Will not have .aar support in this milestone. TODO~~ **DONE!!**
 - [See Issue #1390](#) for aar roadmap
 - .jar
 - source
 - To be done after jar support works.
 - This is a java_library that adds the android.jar
 - This shouldn't be too hard to add now (famous last words).

Interface with Android SDK

[DONE! \[0\] \[1\]](#)

- Google Play Services, support-lib, etc.
- With [new classpath rules](#), sdk libs will need to be in 3rdparty.
- Done, for jars and sources. Added the local SDK to ivysettings.
Going in with library support.
- [Here is a gist of the ivysettings.xml](#) with both SDK m2 repos included as resolvers.
- The above pulls aars by default.

Improve Testing :

[DONE! \[0\] \[1\] \[2\] \[3\] \[4\] \[5\]](#)

- The Android testing was added piecemeal, ad hoc as tasks were added. There is enough code to where I can start regularizing the tests and testing framework.
 - Some of the older classes have incomplete Task testing.
 - AaptGen DONE
 - dx_compile DONE
 - aapt_builder DONE
 - Targets
 - Most android targets have limited or no testing, since they were merged as skeleton targets. Needs to be fixed.
 - I pulled a bunch of the code out of android_target and into manifest_parser. So this is less of an issue (although it still exists)
 - Look at jar_target test classes.
- Could run some better tests on the products created by the integration tests.
- DONE

I am considering this closed. We could still use added testing but the existing testing is now in line with the rest of Pants.

Convert to new option system

DONE!

- Remove arbitrary config access.

Add a gradle-style Android project to examples

DONE!

- Current implementation makes porting easy. Alternative structures should plug right in.
- In progress

Map output streams (stdout, stderr) of binary use

DONE

- Jarsigner
- Aapt

Rename android_binary to android_app

Mid-Term TODO

Redesign Android Build Files

DONE

I would like to have this in Near-Term, but time runs short.

- Pull Keystore SigningConfig into the BUILD files
 - This just doesn't fit as an option, imo.
- Can we make android_resource targets implicit?
 - Just have the resource_dir param to android_targets and create the android_resource targets from there. Or is that mutating the target graph and not allowed?
- ~~• android_library is going in while in a transitional state. It is will be more complicated than necessary. Needs to be simplified.~~
- ~~• Consider unifying the now separate jar_library, unpacked_jar, and android_library. They are different pieces of the same android_library idea.~~
- ~~• As of now, android libraries with resources are AndroidLibrary targets and libraries w/o are just unpacked_jars. All of the android_library targets need to be unified.~~
- ~~• Should I move the include/excludes from AndroidLibrary to AndroidDependency?~~
- ~~◦ This would allow us to enforce the requirement for all Android targets to have a manifest in the BUILD file (except for android_dependency, I guess).~~
- ~~• Downsides is that it makes it hard to reuse things like the 3rdparty android definitions — it would have to be repeated for each includes pattern!~~

DONE!

Test and check invalidation

- I now believe the Android invalidation to be broken and in dire need of comprehensive tests.
- Close issue #468
 - Tests to check the invalidation pipeline for Android products.
 - See John's comment here: <https://rbcommons.com/s/twitter/r/1776/>
 - Re testing, its all there, the trick is to return the actual targets operated upon from execute and raise a TaskError subclass with the successfully processed and failing targets as members. See here - 97% test coverage for a task with semi-sophisticated invalidation:
<https://rbcommons.com/s/twitter/r/1772/>

Also note Benjy's comments, its slightly hacky to use Task.execute / TaskError like this, but I think its a net win.
 - hopefully some guidance can be found for this issue.
- Look at a fingerprint strategy for android artifacts. Android SDK should be considered with artifact caching.
 - JDK version is, [see bug here](#).
- Also make sure that resources are rebuilt if they change, currently that is not the behavior.

Finish the compilation differences between debug/release

- These are not handled differently yet, just signed differently.
 - See below in AaptTask improvement plans.

Rework Android options and consider subsystem

- How can I reduce the boilerplate in render_args for SDK tools?
- Use the JVM subsystem for jvm args and create a subsystem for tasks-wide target_sdk and build_tools, etc.
- Do something sensible with build_tools_version
 - Currently defined in the android_target init. Which, wat.

Pass Args to tools

- Make a passthrough option to allow shops to pass chosen flags to tooling.
- **Update:** Benjy just added this for Jvm. See the review for details:
 - <https://rbcommons.com/s/twitter/r/1696/>

Unittest and Android testing

- Android Testrunner task. This looks easy, which means it will be a real chore.

Research adding annotations.jar to global android classpath

- support for annotations processors
- Also, if minSdkVersion < 14 such that it adds an appcompat dependency

Manifest Merging.

- This will probably require the AndroidManifest become a first class target.
- <http://tools.android.com/tech-docs/new-build-system/user-guide/manifest-merger>
- When adding aars, sometimes things need to be added to the AndroidManifest (like google_play_services_version). This can probably be added after aar support with the caveat that it must be done by hand for now.
- This is a feature for library support and probably a must-have ASAP.

AaptTask Feature-adds

- A 3rd Aapt step?
 - Create a .ap_ that contains the merged resources, and then later add the dex file to that to create the unsigned.apk. Looks to have perf wins.
- Crunch PNGS
- Look for and integrate assets folder
 - could just be an optional field in android_target BUILD file.
- Look into the aapt flags
 - Our version creates fat files with all the debug code. Could be generously trimmed as seen elsewhere.
- The AaptGen refactor does not include invalidation framework.
 - Could roll its own which...not ideal. File presence checks work, but don't reflect changes.

Not sure when, but still priorities

- AIDL support
- minSdk
- Hook into pants bootstrap for the AndroidConfigUtil. I hate that the example config is only available if you have already built an Android target, that is so dumb.
- Get Android Aapt off of Codegen
 - 1-1 so not needed. Use round manager.
- Audit Release build support
 - Crunch .pngs with aapt
- Dex merging. This sounds like it has some worthwhile perf wins.

Long Term TODO

- run the binaries, ie ./pants run ...
- Proguard
- NDK
- Flavors
- Dex Merging
 - DxCompile is the slowest part of the build, the tools support merging dex files.
- Resource merging/shrinking
 - Match gradle priority order:
 - *BuildType -> Flavor -> main -> Dependencies.*
 - Custom gradle tasks, managing the xml I guess.
 - <https://android.googlesource.com/platform/tools/build/+/master/gradle/src/main/groovy/com/android/build/gradle/tasks/MergeResources.groovy>
 - Shrink/minify of resources
 - Probably custom task
 - Merge the resources manually and remove the use of the -S flag in Aapt.
 - <https://android-review.googlesource.com/#/c/109101>
 - Filter resources by supported locales
- **KeyToolTask:**
 - A ConsoleTask subclass. Seems low priority to me now that we fallback to SDK's debug.