Changes to the build environment for 5.4.0 from 5.3.0

This page will describe changes that will be made to the build environment for ATAK 5.4 as part of changes to the API/ABI that would affect plugin developers per the deprecation policy.

For previous changes see Changes to the build environment for 5.3.0 from 5.2.0

Build Chain

Minimum Version

ATAK Core is set up so the minimum Android version is Android 5.0 (21) but this does not restrict plugins from targeting a newer version of Android.

build.gradle

Before	After
compileSdkVersion 34 targetSdkVersion 34	compileSdkVersion 34 targetSdkVersion 34

VERY IMPORTANT

Please note the following changes enforced by this move from Android: https://developer.android.com/about/versions/14/behavior-changes-14

Specifically the change to Context::registerReceiver which now must take a 3rd parameter if used in the plugin (unless used in a service or activity within the plugin).   Please note that plugins should make use of AtakBroadcast instead of directly registering a system or application broadcast receiver.

If you must use Context::registerReceiver please make use of this pattern in your code

```
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.TIRAMISU) {
            context.registerReceiver(receiver, filter,
                    Context.RECEIVER_EXPORTED);
        } else {
            context.registerReceiver(receiver, filter);
        }
```

If you do not do this, you will receive crash reports that looks like this.   This was a google decision across all Android development.

```
Exception java.lang.SecurityException:
  at android.os.Parcel.createExceptionOrNull (Parcel.java:3069)
  at android.os.Parcel.createException (Parcel.java:3053)
  at android.os.Parcel.readException (Parcel.java:3036)
  at android.os.Parcel.readException (Parcel.java:2978)
  at android.app.IActivityManager$Stub$Proxy.registerReceiverWithFeature (IActivityManager.java:6137)
  at android.app.ContextImpl.registerReceiverInternal (ContextImpl.java:1913)
  at android.app.ContextImpl.registerReceiver (ContextImpl.java:1853)
  at android.app.ContextImpl.registerReceiver (ContextImpl.java:1841)
  at android.content.ContextWrapper.registerReceiver (ContextWrapper.java:772)
```

Gradle

The recommended build process is to use Gradle 8.13 and Android Gradle Plugin 8.9.0.   It is encouraged but not required as people have reported success with using newer versions of Gradle and the Android Gradle Plugin

Java

The source and target compatibility is still 1.8; however, the build environment must make use https://adoptium.net/ JDK 17.

NDK

The recommended NDK is 25.1.8937393 and the full version is shipped with an abiFilter of { "armeabi-v7a", "arm64-v8a", "x86", "x86_64" }. If your plugin makes use of native libraries we **REQUIRE** shipping at least "arm64-v8a".

Libraries

Upgrades to libraries have been deliberately slow in order to prevent this larger sprawl of dependencies. We continue to strive to keep the number of dependencies low to maximize plugin flexibility. Please note the following changes

Android X

ATAK Core will supply

- `implementation 'androidx.fragment:fragment:1.8.5'`
- `implementation 'androidx.exifinterface:exifinterface:1.3.7'`
- `implementation 'androidx.localbroadcastmanager:localbroadcastmanager:1.1.0'`
- `implementation 'androidx.lifecycle:lifecycle-process:2.8.7'`
- `implementation 'org.jetbrains.kotlin:kotlin-reflect:2.1.10'`

From TAK Kernel on which ATAK is built:

- implementation 'org.greenrobot:eventbus:3.2.0`
- **implementation 'com.squareup.okhttp3:okhttp::4.11.0`   **NEW****

As with previous versions, care must be taken not to duplicate the inclusion of these libraries in a plugin or utilize libraries with different versions. The attempt is still to keep core as lightweight as possible when utilizing any support library to allow for greater flexibility for plugin developers. This is the same paradigm that google has moved towards with breaking up the original android support libraries.

The above provided core have a transient dependencies which need to be considered and possibly omitted. These are listed here:

- `androidx.core:core:1.15.0`
- `androidx.lifecycle:lifecycle-runtime:2.8.7`
- `androidx.lifecycle:lifecycle-viewmodel:2.8.7`
- `androidx.lifecycle:lifecycle-viewmodel-savedstate:2.8.7`
- `androidx.viewpager:viewpager:1.0.0`

Kotlin **2.1.10** is brought in by the these dependencies. Please use ./gradlew app:dependencies to observe what libraries are being brought in by your 1st order dependencies.

At the bottom of the build.gradle file you can indicate which version to resolve.

```
configurations.all {
    resolutionStrategy.eachDependency { details ->
        if (details.requested.group == 'androidx.core') {
            details.useVersion "1.15.0"
        }
        if (details.requested.group == 'androidx.lifecycle') {
            details.useVersion "2.8.7"
        }
        if (details.requested.group == 'androidx.fragment') {
            details.useVersion "1.8.5"
        }
    }
}

configurations.implementation {
       exclude group: 'androidx.core', module: 'core-ktx'
       exclude group: 'androidx.core', module: 'core'
       exclude group: 'androidx.fragment', module: 'fragment'
       exclude group: 'androidx.lifecycle', module: 'lifecycle'
       exclude group: 'androidx.lifecycle', module: 'lifecycle-process'
}
```

For 1st order libraries, you may need to do additional work to exclude out any dependencies supplied by ATAK.   For example if you make use of RecyclerView you may need to do additional work to exclude items like this:

```
   // Recyclerview version depends on some androidx libraries which
   // are supplied by core, so they should be excluded.  Otherwise
   // bad things happen in the release builds after proguarding
   implementation ('androidx.recyclerview:recyclerview:1.1.0') {
       exclude module: 'collection'
       // this is an example of a local exclude since androidx.core is supplied by core atak
       exclude module: 'core'
       exclude module: 'lifecycle'
       exclude module: 'core-common'
       exclude module: 'collection'
       exclude module: 'customview'
   }
```

Please note that from time to time, there are libraries the compile but do not work at runtime.   Please check back for any links to Jira tickets that might provide benefit.

~~ATAK-17914~~ - NoSuchMethodError: No static method getAllViews CLOSED

Proguard Rules

Release building might require additional dontwarn directives at the bottom of the proguard file if you have correctly excluded things from the plugin that are supplied by core.

```
-dontwarn androidx.**
-dontwarn kotlin.**
-dontwarn kotlinx.**
```

Testing

For the test harness, migration was done to use androidTestImplementation 'org.mockito:mockito-core:5.15.2

We have fully removed PowerMockito from the core test harness since it is no longer in support and mockito has added all of the PowerMockito capability.

Additional Notes

Apache Libraries

Problems arise when attempting to use legacy apache http imports. Please add

```
android {
    compileSdkVersion XX
    buildToolsVersion "XX.X.X"

    useLibrary 'org.apache.http.legacy'
```

Package Discovery

Package discover is no longer allowed by normal applications unless there is a relationship between   Android 11 and forward - Google started working really hard to have applications remove `QUERY_ALL_PACKAGES` permission

https://developer.android.com/about/versions/11/privacy/package-visibility

This permission has now been completely removed from the core application and would require your associated apps to declare the faux query entry to allow for the ATAK application to find them.   This is required for plugins as well as any apps that the plugins need to be able to find.   For example if you have a service APK that is required by the plugin, the easiest way is to have your service APK include the same block in their Android Manifest.

see:

```
       <!-- allow for app discovery -->
      <activity android:name="com.atakmap.app.component"
          android:exported="true"
          tools:ignore="MissingClass">
          <intent-filter android:label="@string/app_name">
             <action android:name="com.atakmap.app.component" />
          </intent-filter>
      </activity>
```

ATAK Packages

- **If you have a legacy plugin and would like to bring it forward please review the this example plugin which highlights the changes required: https://git.tak.gov/samples/PluginTemplateLegacy specifically: https://git.tak.gov/samples/PluginTemplateLegacy/-/commit/6307d188e175b28a326a5e400824f67e9daba8e2**
- **If you have a new plugin, please use the PluginTemplate project https://git.tak.gov/samples/plugintemplate**

Native Libraries

If your plugin uses Native Libraries and set the Android SDK Target to 26 or newer (e.g. minSdkVersion 26)

Then you will need to make sure your plugin AndroidManifest.xml application block contains: android:extractNativeLibs="true"

gradle.properties

The gradle.properties files is required to have:

```
# required for app bundles that utilize native libraries
android.bundle.enableUncompressedNativeLibs=false
```

Transparent Signing

For developers who will need an AAB file capable of being turned into an APK that can load into ATAK, they will need to create the following file

app/src/main/res/xml/com_android_vending_archive_opt_out.xml

with the following contents:

```
<?xml version="1.0" encoding="utf-8"?>
<optOut />
```

UX/UI Style Guide

For plugin developers interested in utilizing the Arsenal Style guide - please see https://www.figma.com/file/7TEdV4um5gh8GPK5I7ts5a/ARSENAL---ATAK-Design-System?node-id=2%3A27

Plugin Linting

Linting is the **automated checking of your source code for programmatic and stylistic errors**. This is done by using a lint tool (otherwise known as linter). A lint tool is a basic static code analyzer.   As of atak-gradle-takdev version 2.5.1, linting is performed to make sure that your plugin will conform to certain static rules.   One such rule is the proguard obfuscation naming rule which requires that your plugin have an obfuscation name other than Plugin Template. The error message you will see is:

```
Proguard configured for plugintemplate, but this plugin is not the plugintemplate.
```

The fix for this would be to modify repackageclasses line in the proguard-gradle.txt file to correctly reflect an identifying name for your plugin.   For example

-repackageclasses atakplugin.PluginTemplate

would become

-repackageclasses atakplugin.MyCoolPlugin