PLUG-IN DEVELOPMENT
PLUGIN STRUCTURE

DIGGING INTO PLUG-IN DEV

General Considerations before starting development:

- Please use the PluginTemplate as a template to derive your plugin.
- Please make sure to request a Git repository for your plugin before starting.
- Decide what version of the SDK you would like to target.
- Decide what flavors you would like to support (MIL, CIV, FVEY, INTL)
  - Branching is important maintenance-<flavor>-<version> where flavor can be mil, civ, fvey, into and version can be 3.10, 3.11, 3.12, etc
- Use git tag –a to meaningfully describe your branches since it is used in the versionName

SDK FILES

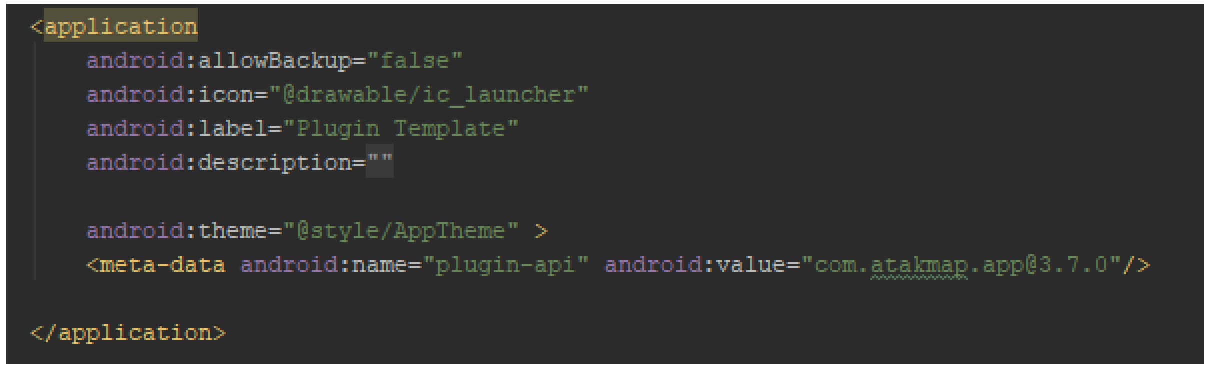
The SDK includes the following:

- ATAK.apk : development version of ATAK
- main.jar - ATAK library used to build plug-ins
- atak-javadoc.jar - javadoc for ATAK library
- doc/broadcast.txt - list of local broadcast intents in ATAK
- VERSION.txt - the version number associated with the current SDK
- Plugins - actively developed plug-in projects not in GiT
- Examples - example plug-ins

PLUG-IN DETECTION

ATAK finds plug-ins by checking installed apks for the following:

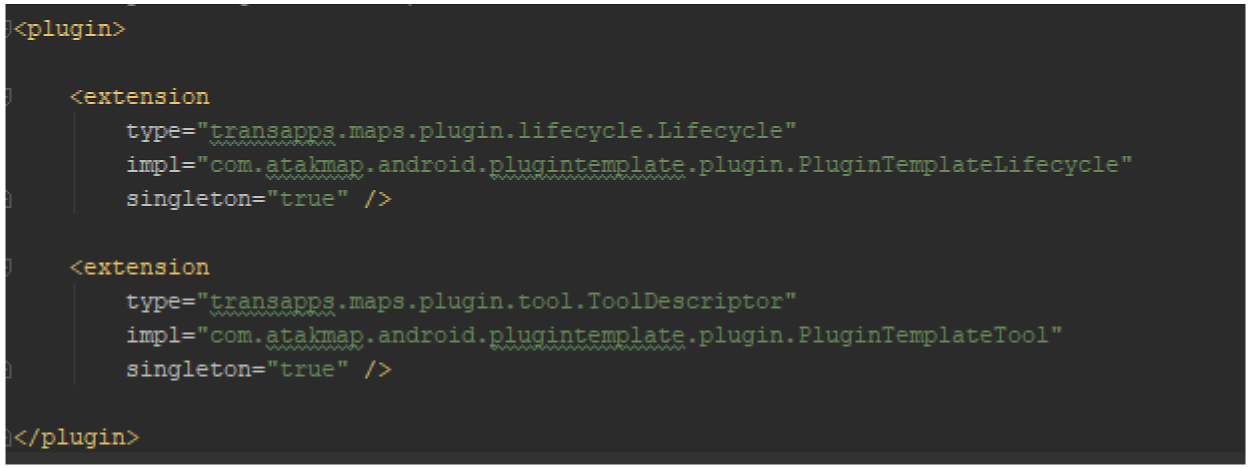
- The Manfiest must contain a `meta-data` tag with a matching plug-in api version
- There is an `assets/plugin.xml` file
  - This file defines the plug-in's extensions, or entry points. These are the classes that get loaded into ATAK by a custom class loader



```
<application
    android:allowBackup="false"
    android:icon="@drawable/ic_launcher"
    android:label="Plugin Template"
    android:description=""

    android:theme="@style/AppTheme" >
    <meta-data android:name="plugin-api" android:value="com.atakmap.app@3.7.0"/>

</application>
```



```
<plugin>

    <extension
        type="transapps.maps.plugin.lifecycle.Lifecycle"
        impl="com.atakmap.android.plugintemplate.plugin.PluginTemplateLifecycle"
        singleton="true" />

    <extension
        type="transapps.maps.plugin.tool.ToolDescriptor"
        impl="com.atakmap.android.plugintemplate.plugin.PluginTemplateTool"
        singleton="true" />

</plugin>
```

ATAK COMPONENTS

Tool (optional)

- A plug-in `Tool` is an entry in the ATAK toolbar. It lets the user to add an entry that can be used to perform an action on click (i.e., launch the plug-in's UI)

Lifecycle

- A plug-in `Lifecycle` is the main plugin entry point. It handles ATAK's lifecycle callbacks and is used to load and initialize the main plug-in components (`MapComponents`)
- Analogous to an Android `Application` class

ATAK COMPONENTS (CONT.)

(Abstract)MapComponent

- The main plug-in components are `MapComponent`:

  > A map component is the building block for all activities within the system. This defines a concrete thought or idea.

- Map components setup UI components (`DropDownReceivers`), preferences and other high-level components
- Analogous to an Android `Activity` class

ATAK COMPONENTS (CONT.)

DropDownReceiver

- A plug-in's UI is generally made of `DropDownReceivers`
  - These receivers are generally created in a MapComponent and registered to receive broadcasts (they subclass `BroadcastReceiver`).
  - This is how the Tool component can be used to launch a root `DropDownReceiver`
- A drop-down is a container (usually a side panel) that can contain any standard Android layout
- Analogous to an Android `Fragment` class

ATAK COMPONENTS (CONT.)

MapView

- The `MapView` object is passed to the `Lifecycle` component
- It contains the ATAK context
- It contains most of the moving map capability and APIs within ATAK
  - See example plug-ins on various uses of the `MapView`

PLUGIN CONTEXTS

A plug-in receives two contexts in it’s `Lifecycle` object, a *plug-in* context, and an *ATAK* context (attached to the `MapView` object)

- The *ATAK* context:
  - Has permissions from ATAK Manifest
  - Used to interact with the UI (toasts, dialogs, etc.)
  - Used to load assets and resources *from ATAK library*
- The *plug-in* context:
  - Has same permissions as ATAK context (not plugin’s Manfiest)
  - Cannot be used to interact with the UI (toasts, dialogs, etc.)
  - Used to load assets and resources *from plug-in* (layouts, images, etc.)

NDK

Plug-ins can utilize Native Libraries

- Only NDK version 12B is supported
- Only supported abis are `armeabi-v7a` and `x86`

DOWNLOAD NDK (12B)

Windows

- https://dl.google.com/android/repository/android-ndk-r12b-windows-x86.zip
- https://dl.google.com/android/repository/android-ndk-r12b-windows-x86_64.zip

Linux

- https://dl.google.com/android/repository/android-ndk-r12b-linux-x86_64.zip

Mac

- https://dl.google.com/android/repository/android-ndk-r12b-darwin-x86_64.zip

EXCLUDE NATIVE LIBRARIES

- If unsupported abis are included, ATAK may fail to load properly.
- Use `build.gradle` to include only supported abis:

```
defaultConfig {
    ndk {
        abiFilters "armeabi-v7a", "x86"
    }
    packagingOptions {
        exclude "lib/arm64-v8a/libsqliteX.so"
    }
}
```

SUPPORT LIBRARY V4

ATAK uses the support-v4 Android Support Library, revision 26.0.   Plugin developers are encouraged to make use of AndroidX support-v4 libraries.

ADDITIONAL NOTES

SETUP ANDROID DEBUGGING

Plug-ins do not have their own process, they are loaded into ATAK’s process space

1. Set break points in Android Studio
2. In Android Studio attach to an existing process
   - `Run -> Attach debugger to Android process`
   - Check `Show all processes`
   - Select `com.atakmap.app`

ANDROID VIRTUAL DEVICE (OPENGL ISSUES)

Note: For deploying in an emulated environment, ATAK might crash due to incompatibilities with the computer graphics card. Please either modify

1) construct a file in the /sdcard/atak directory called "opengl.broken"

   - this will set USE_GENERIC_EGL_CONFIG true

2) set the system property "USE_GENERIC_EGL_CONFIG" to "true"

If EGL is activated successfully, you should see a log message stating: "application has been informed that OPEN GL is a bit busted"

If using AVD, it may be necessary to switch the GPU mode. This can be done by accessing the Settings for your emulated device, then going to the Settings tab, then Advanced tab. From here you can select the OpenGL ES Renderer, which is the setting that may need to be changed. The emulated device will need to be cold booted for the change to be effective. If all else fails, AVD may be launchedfrom the command line and passed the -gpu flag with various options--"host" and "guest" are likely the most helpful.