

ATAK Plugin Development Introduction

Training

Agenda

- ◇ About Adeptus Cyber Solutions
- ◇ Pre-requisites
- ◇ Exploring the SDK from tak.gov
- ◇ Hello World Plugin
 - ◇ Keystore Setup
 - ◇ Building and Installing
 - ◇ Deploying to an Emulator
 - ◇ Debugging Hello World
- ◇ Plugin from Scratch
- ◇ Troubleshooting
- ◇ Third Party Pipeline (TPP) on tak.gov

About Adeptus Cyber Solutions, LLC

Adeptus Cyber Solutions (ACS), where real-world problems meet cutting-edge solutions. Our team is at the forefront of Android Team Awareness Kit (ATAK) Plugin development, dedicated to crafting custom solutions for the problems you face. With a rich history of expertise, our dedicated team at ACS is at the forefront of ATAK development, bringing innovation and practical solutions to meet the unique needs of our clients.

Our ATAK Expertise:

ACS has been a trailblazer in the ATAK development landscape, continuously evolving alongside the platform's advancements. With over 3 years heavily involved in the TAK ecosystem, we have consistently demonstrated our proficiency in developing tailored ATAK Plugins that cater to a diverse range of requirements. At ACS, we take pride in our extensive portfolio of successfully developed ATAK Plugins. These plugins span a spectrum of functionalities, from monitoring the health statistics of warfighters to creating seamless workflows for launching and recovering unmanned aircraft. Our commitment to delivering high-quality, robust solutions has garnered us a reputation as a trusted partner in the ATAK development ecosystem.

Elevate your ATAK experience with Adeptus Cyber Solutions. Whether you require specialized plugins, ATAK training, TAK Server support, assistance with CI/CD pipelines, or plugin quality testing we are here to assist.

Visit us at www.adeptuscybersolutions.com for more information

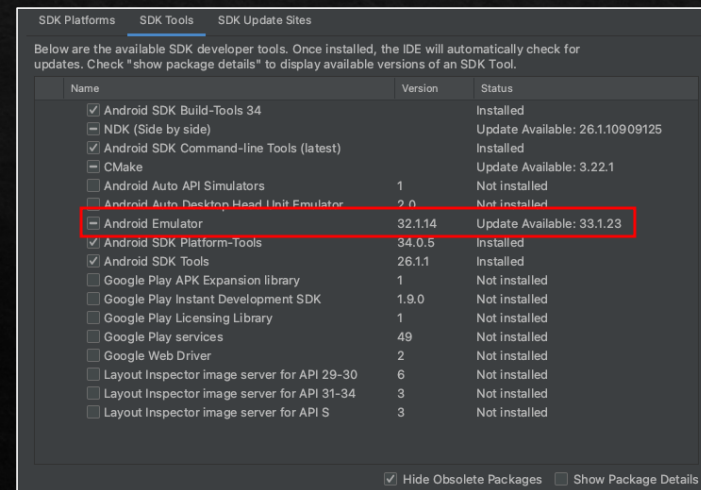
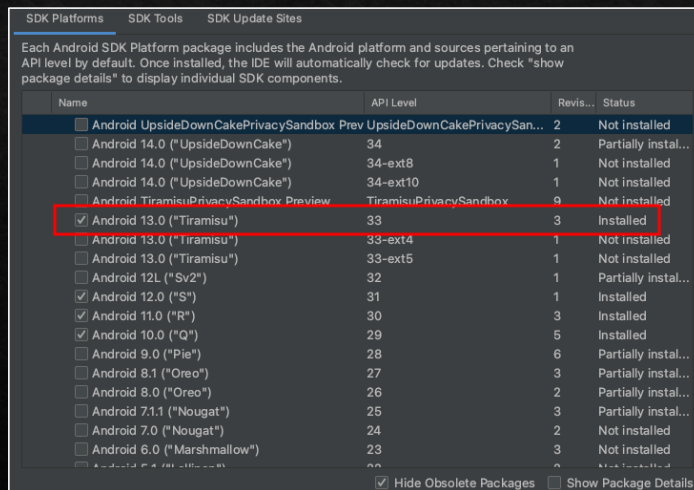
Pre-requisites

Pre-requisites

- ◇ A Mac, Windows or Linux system to install software on
- ◇ Knowledge of how to install software on your system
- ◇ An understanding of Android Studio, will not be covering how to use Android Studio
- ◇ Working knowledge of Android
- ◇ A tak.gov account so you can obtain the SDK and submit plugins to the Third Party Pipeline
- ◇ Android Studio – (<https://developer.android.com/studio>)
- ◇ Java 11 JDK
 - ◇ Make sure this is the default JDK being utilized
 - ◇ <https://docs.aws.amazon.com/corretto/latest/corretto-11-ug/downloads-list.html>
- ◇ Join the TAK Community on Discord
 - ◇ This is where you will find lots of good information and other devs willing to help

Pre-requisites: Android Studio

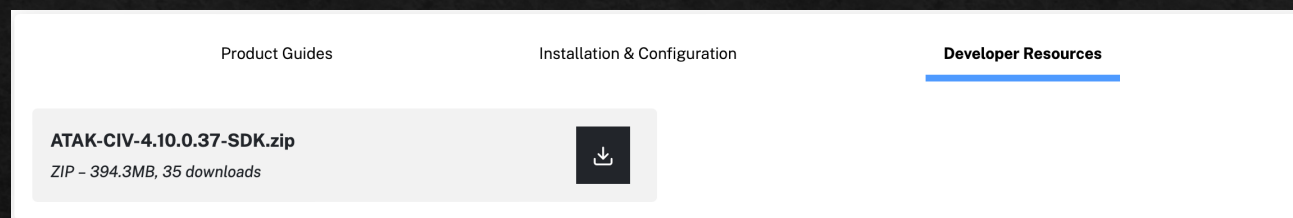
- ❖ SDK Manager (Tools -> SDK Manager)
 - ❖ SDK Platform: Android 13
 - ❖ SDK Tools: Android Emulator



Exploring the SDK from tak.gov

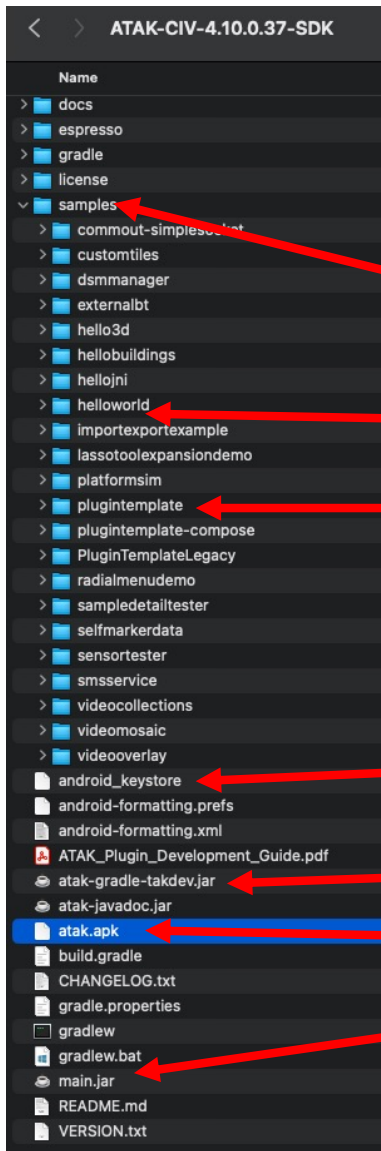
Exploring the SDK from tak.gov

- ◆ Download the latest ATAK 4.10 SDK



- ◆ Extract the ZIP
- ◆ Why not use GitHub?
 - ◆ Code base is older and not maintained
 - ◆ It's the actual code for ATAK and not the SDK, would require you to compile it to make the SDK
 - ◆ This training is on how to build plugins and not ATAK

Exploring the SDK from tak.gov



Sample Plugins Folder

Hello World Plugin

Plugin Template – Used to create your own templates

android_keystore – keystore used to sign plugins (note its location)

atak-gradle-takdev.jar – needed for compilation of plugins

Developer ATAK APK - install this on device or emulator

main.jar – needed for compilation of plugins

Hello World Plugin

Hello World Plugin - Setup

- ◇ Need to generate keys in a keystore to sign the APK
 - ◇ Debug Key:
 - ◇ *keytool -genkey -alias debug -keyalg RSA -keysize 2048 -keystore android_keystore*
 - ◇ Release Key:
 - ◇ *keytool -genkey -alias release -keyalg RSA -keysize 2048 -keystore android_keystore*
- ◇ Be sure to document the password for each key and for the keystore
- ◇ Keystore should be saved to the folder two levels up from the plugin source
 - ◇ See “Exploring the SDK from tak.gov” slide

Keystore Generation

```
>keytool -genkey -alias debug -keyalg RSA -keysize 2048 -keystore android_keystore
Command line args: [-genkey, -alias, debug, -keyalg, RSA, -keysize, 2048, -keystore, android_keystore]
Enter keystore password:
Re-enter new password:
What is your first and last name?
[Unknown]: Joe Smith
What is the name of your organizational unit?
[Unknown]:
What is the name of your organization?
[Unknown]: ACME Inc.
What is the name of your City or Locality?
[Unknown]: Anytown
What is the name of your State or Province?
[Unknown]: VA
What is the two-letter country code for this unit?
[Unknown]: US
Is CN=Joe Smith, OU=Unknown, O=ACME Inc., L=Anytown, ST=VA, C=US correct?
[no]: yes
```

Debug Key Generation

```
>keytool -genkey -alias release -keyalg RSA -keysize 2048 -keystore android_keystore
Enter keystore password:
What is your first and last name?
[Unknown]: Joe Smith
What is the name of your organizational unit?
[Unknown]:
What is the name of your organization?
[Unknown]: ACME Inc.
What is the name of your City or Locality?
[Unknown]: Anytown
What is the name of your State or Province?
[Unknown]: VA
What is the two-letter country code for this unit?
[Unknown]: US
Is CN=Joe Smith, OU=Unknown, O=ACME Inc., L=Anytown, ST=VA, C=US correct?
[no]: yes
```

Release Key Generation

```
>ls
ATAK_Plugin_Development_Guide.pdf  atak-gradle-takdev.jar  gradle.properties
CHANGELOG.txt                     atak-javadoc.jar      gradlew
README.md                          atak.apk              gradlew.bat
VERSION.txt                        build.gradle          license
android-formatting.prefs          doc                   main.jar
android-formatting.xml            espresso              samples
android_keystore                  gradle
```

New Keystore

Keystore Setup

- ◇ Edit the plugin's *app/build.gradle* file
- ◇ Locate the *signingConfigs* section
 - ◇ Updated the debug and release keys
 - ◇ Set the:
 - ◇ *storePassword*
 - ◇ *keyAlias*
 - ◇ *keyPassword*

```
signingConfigs {
    debug {
        storeFile file("${buildDir}/android_keystore")
        storePassword "password"
        keyAlias "debug"
        keyPassword "password"
    }
    release {
        storeFile file("${buildDir}/android_keystore")
        storePassword "password"
        keyAlias "release"
        keyPassword "password"
    }
}
```

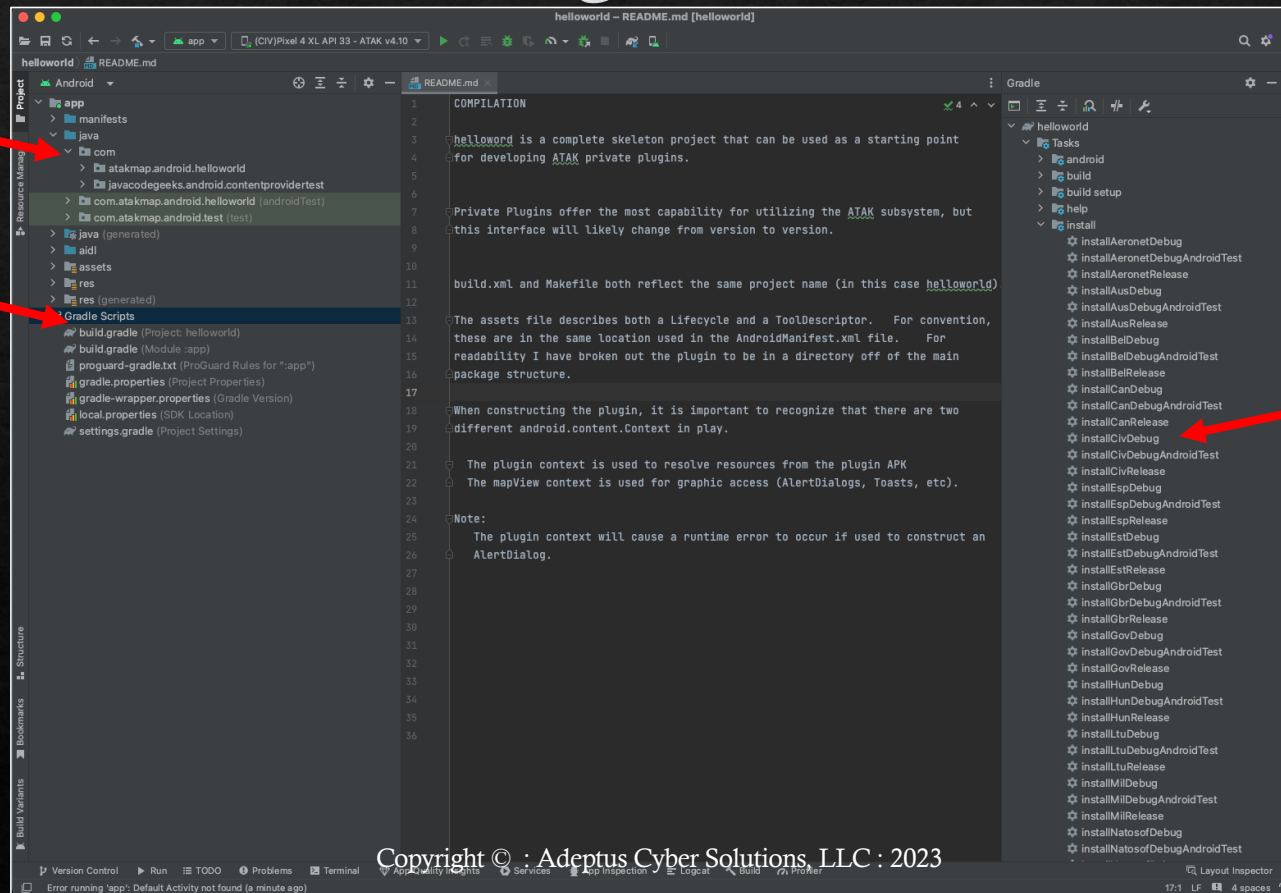
Hello World Plugin – Android Studio

Code

Gradle Build Scripts

Gradle Task Window

installCivDebug



The screenshot shows the Android Studio IDE with the following components:

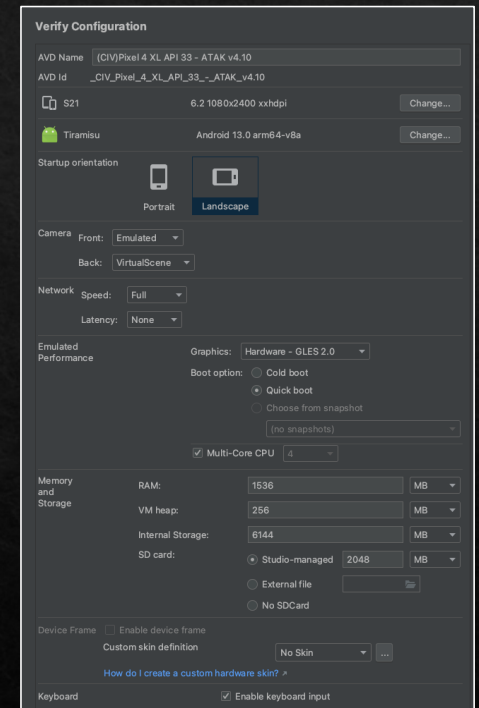
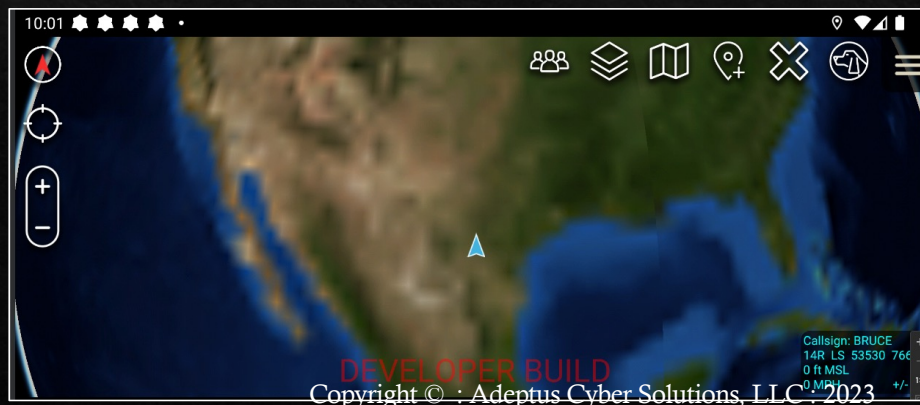
- Project View (Left):** Shows the project structure for 'helloworld'. The 'Gradle Scripts' folder is expanded, showing files like 'build.gradle (Project: helloworld)', 'proguard-gradle.txt', and 'settings.gradle'.
- Code Editor (Center):** Displays the 'README.md' file with the following content:


```

1  COMPILATION
2
3  helloworld is a complete skeleton project that can be used as a starting point
4  for developing ATAK private plugins.
5
6  Private Plugins offer the most capability for utilizing the ATAK subsystem, but
7  this interface will likely change from version to version.
8
9
10 build.xml and Makefile both reflect the same project name (in this case helloworld)
11
12 The assets file describes both a Lifecycle and a ToolDescriptor. For convention,
13 these are in the same location used in the AndroidManifest.xml file. For
14 readability I have broken out the plugin to be in a directory off of the main
15 package structure.
16
17 When constructing the plugin, it is important to recognize that there are two
18 different android.content.Context in play.
19
20 The plugin context is used to resolve resources from the plugin APK
21 The mapView context is used for graphic access (AlertDialogs, Toasts, etc).
22
23 Note:
24 The plugin context will cause a runtime error to occur if used to construct an
25 AlertDialog.
26
27
28
29
30
31
32
33
34
35
36
      
```
- Gradle Task Window (Right):** Shows a list of tasks for the 'helloworld' project. The 'installCivDebug' task is highlighted with a red arrow.

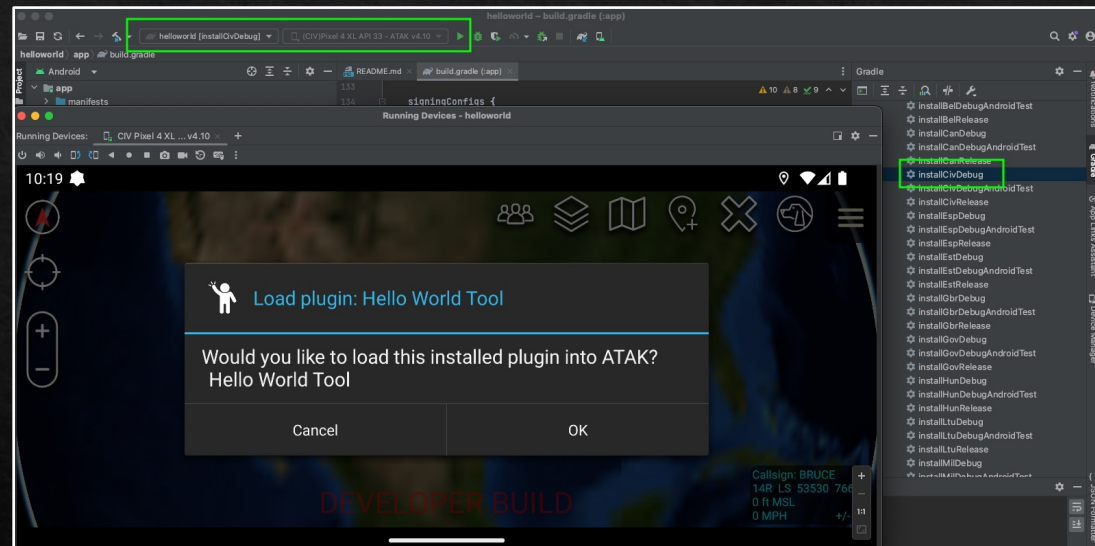
Hello World Plugin – Emulator

- ◆ Configure and start an emulator
 - ◆ Be sure to use Android 13 as the system image
- ◆ Transfer the *atak.apk* to the emulator and install it
- ◆ Follow the setup prompts for ATAK
- ◆ Similar steps for a physical device, copy ATAK APK and install it



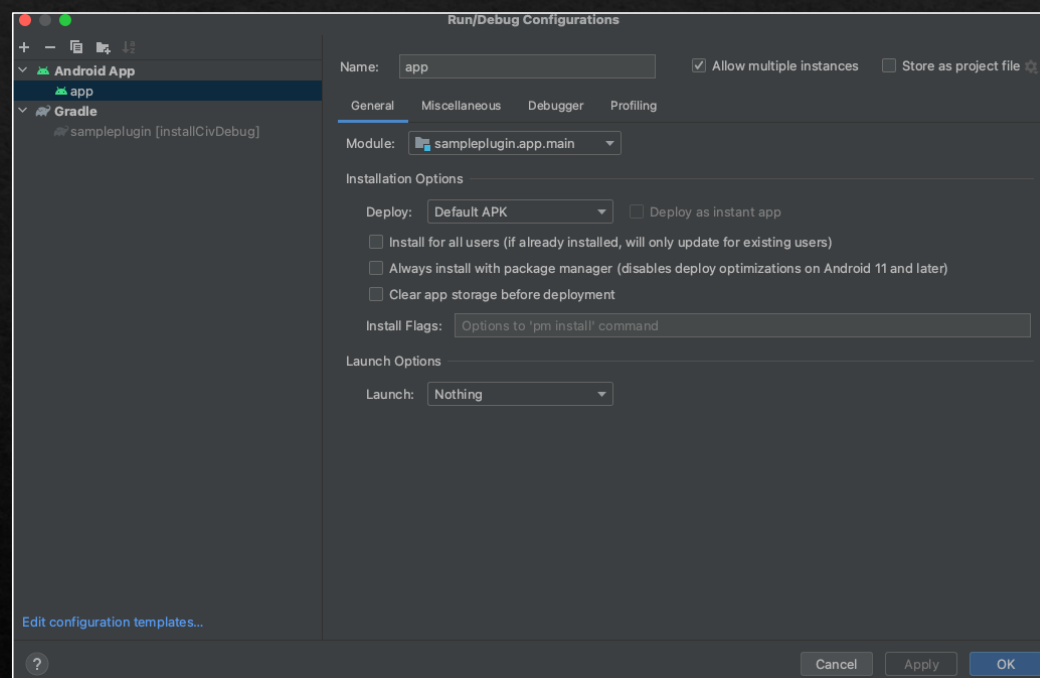
Hello World Plugin - Deployment

- ◆ Emulator/Device is running and ATAK is the app that is open
 - ◆ Map is displayed and it says “DEVELOPER BUILD” on the bottom of the screen
- ◆ From the Android Studio Gradle Task Window
 - ◆ Locate Tasks -> install -> installCivDebug
 - ◆ Double click *installCivDebug*
 - ◆ This launches the build and install of the plugin
- ◆ Hello World Tool now shows in the menu



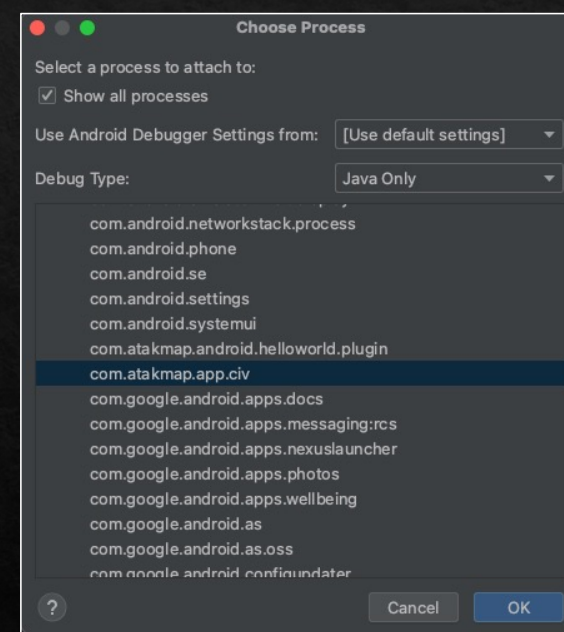
Hello World Plugin - Deployment

- ◆ To correct the Android App run configuration in Android Studio
- ◆ From the **Run** menu select **Edit Configurations**
 - ◆ Expand the **Android App** (if compressed)
 - ◆ Select app
 - ◆ In the **Launch Options** set the **Launch** to *Nothing*



Hello World Plugin - Debugging

- ◇ Since a plugin is not a true Android app you cannot click on the “Debug” button of Android Studio.
- ◇ To debug a plugin:
 - ◇ From the Android Studio Run menu select “Attach Debugger to Android Process”
 - ◇ In the popup window locate the device / emulator
 - ◇ Expand the list and select *com.atakmap.app.civ* to debug
 - ◇ You have to connect to the ATAK app in order to debug your plugin
- ◇ Once connected to ATAK, you can debug your plugin just as you would any other application



Plugin from Scratch

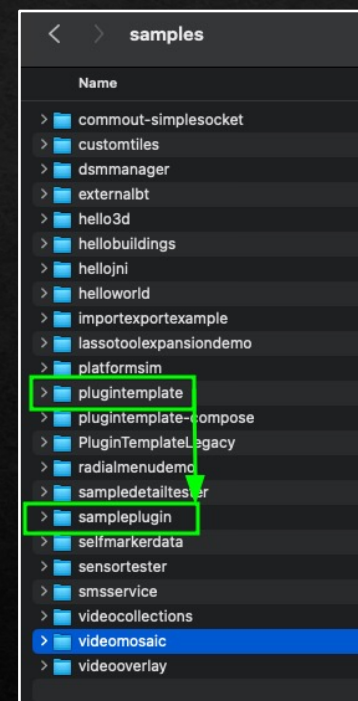
Plugin Development Cycle

- ◇ Start with the *plugintemplate* from the SDK
- ◇ Develop your plugin, testing with the SDK provided ATAK APK
- ◇ Once ready to deploy, send plugin code to Third Party Pipeline (TPP).
 - ◇ Version of ATAK to build against is defined in the *app/build.gradle*
 - ◇ Will compile a release build of your plugin
 - ◇ Will sign your plugin for release

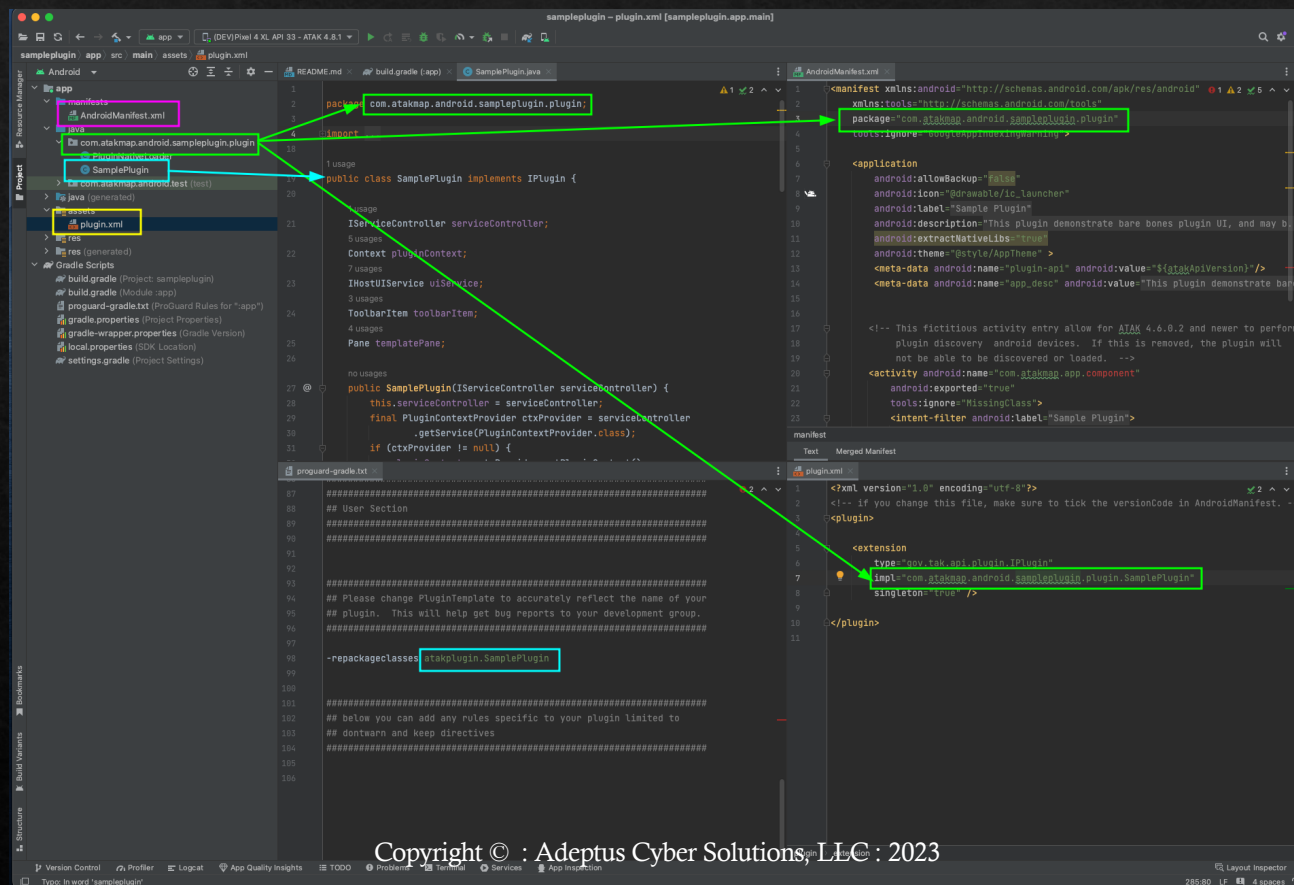
- ◇ Release version of plugins will only work in release versions of ATAK
- ◇ Debug versions of plugins will only work in debug versions of ATAK
- ◇ The version of ATAK the plugin was built for needs to match the version of ATAK.
 - ◇ For example:
 - ◇ A 4.9 plugin will NOT work on ATAK 4.8.1
 - ◇ A 4.8.1 plugin will NOT work on ATAK 4.9
 - ◇ This changes in ATAK 5.0, there is backwards compatibility with 4.10

Plugin from Scratch - Setup

- ◇ tak.gov provides in the SDK a *plugintemplate* that is used as the basis for any new plugin
- ◇ Copy the existing *plugintemplate* in the SDK folder to a new folder called *sampleplugin*
- ◇ Open the *sampleplugin* in Android Studio
- ◇ Open the *proguard-gradle.txt*
 - ◇ At the bottom, change the *PluginTemplate* in the *repackageclasses* setting to something meaningful. Otherwise the TPP build will fail.
- ◇ In the Java code
 - ◇ Rename the *plugintemplate* portion of the package to something meaningful
 - ◇ Rename the *PluginTemplate* class to something meaningful
 - ◇ Also adjust the package to the *R* import to match what was just set
- ◇ In the *AndroidManifest.xml*
 - ◇ Update the package to match the name of the package that was just renamed
- ◇ In the *assets* folder
 - ◇ Edit the *plugin.xml* and update the package name
- ◇ In the *res/values* folder
 - ◇ Edit the *strings.xml* file and change the name of the app



Plugin from Scratch - Setup



The screenshot displays the setup of an Android plugin from scratch. Key components and their relationships are highlighted:

- Project Explorer:** Shows the project structure with files like `AndroidManifest.xml`, `SamplePlugin.java`, and `plugin.xml` highlighted in red boxes.
- SamplePlugin.java:** The main plugin class, annotated with a red box around the package name `com.atakmap.android.sampleplugin.plugin` and the class name `SamplePlugin`.
- AndroidManifest.xml:** The main application manifest, annotated with a red box around the package name `com.atakmap.android.sampleplugin.plugin` and the `android:label` attribute.
- plugin.xml:** The plugin configuration file, annotated with a red box around the `impl` attribute pointing to the `SamplePlugin` class.
- Annotations:** Red arrows indicate the flow of information: from the package name in the Java code to its declaration in the AndroidManifest.xml, and from the class name in the Java code to its implementation in the plugin.xml file.

```
1 package com.atakmap.android.sampleplugin.plugin;
2
3 import
4
5
6
7 public class SamplePlugin implements IPlugin {
8
9     IServiceController serviceController;
10    Context pluginContext;
11    IHostUIService uiService;
12    ToolbarItem toolbarItem;
13    Pane templatePane;
14
15    no usages
16    public SamplePlugin(IServiceController serviceController) {
17        this.serviceController = serviceController;
18        final PluginContextProvider ctxProvider = serviceController
19            .getService(PluginContextProvider.class);
20        if (ctxProvider != null) {
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
```

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
1    xmlns:tools="http://schemas.android.com/tools"
2    package="com.atakmap.android.sampleplugin.plugin"
3    tools:ignore="GoogleAppIndexingWarning">
4
5
6
7    <application
8        android:allowBackup="false"
9        android:icon="@drawable/ic_launcher"
10       android:label="Sample Plugin"
11       android:description="This plugin demonstrate bare bones plugin UI, and may b
12       android:extractNativeLibs="true"
13       android:theme="@style/AppTheme">
14       <meta-data android:name="plugin-api" android:value="${gizakApiVersion}"/>
15       <meta-data android:name="app_desc" android:value="This plugin demonstrate ba
16
17
18       <!-- This fictitious activity entry allow for AT&A 4.6.0.2 and newer to perform
19       plugin discovery android devices. If this is removed, the plugin will
20       not be able to be discovered or loaded. -->
21       <activity android:name="com.atakmap.app.component"
22           android:exported="true"
23           tools:ignore="MissingClass">
24           <intent-filter android:label="Sample Plugin">
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
```

```
<?xml version="1.0" encoding="utf-8"?>
1 <!-- If you change this file, make sure to tick the versionCode in AndroidManifest. --
2
3 <plugin>
4
5
6 <extension
7     type="gov.tak.api.plugin.IPlugin"
8     impl="com.atakmap.android.sampleplugin.plugin.SamplePlugin"
9     singleton="true" />
10
11 </plugin>
```

```
#####
1  ## User Section
2
3 #####
4
5 #####
6
7 #####
8
9 ## Please change PluginTemplate to accurately reflect the name of your
10 ## plugin. This will help get bug reports to your development group.
11 #####
12
13 -repackageClasses atakPlugin.SamplePlugin
14
15 #####
16
17 ## Below you can add any rules specific to your plugin limited to
18 ## dontwarn and keep directives
19 #####
20
21 #####
```

Plugin from Scratch - Setup

- ◆ Edit the plugin's *app/build.gradle* file
- ◆ Locate the *signingConfigs* section
 - ◆ Updated the debug and release keys
 - ◆ Set the:
 - ◆ *storePassword*
 - ◆ *keyAlias*
 - ◆ *keyPassword*

```
signingConfigs {  
    debug {  
        storeFile file("${buildDir}/android_keystore")  
        storePassword "password"  
        keyAlias "debug"  
        keyPassword "password"  
    }  
    release {  
        storeFile file("${buildDir}/android_keystore")  
        storePassword "password"  
        keyAlias "release"  
        keyPassword "password"  
    }  
}
```

Plugin from Scratch – ATAK Version

- ◆ In order to support other versions of ATAK, the plugin has to be compiled against the correct *main.jar* and the version has to be set in the *app/build.gradle*
- ◆ When you submit your plugin to TPP, this variable in the *app/build.gradle* is used to determine the ATAK build environment
- ◆ If you need to test your plugin against ATAK 4.9, for example, you need to acquire the 4.9 SDK from tak.gov and change out the *main.jar* and change this variable to say “4.9.0”

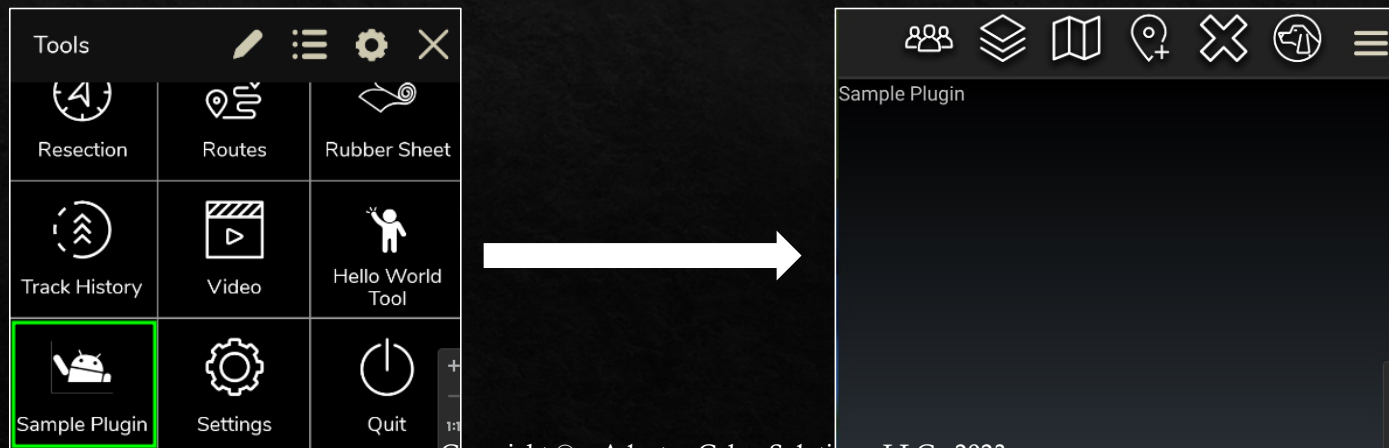
```
buildscript {  
  
    ext.PLUGIN_VERSION = "1.0"  
    ext.ATAK_VERSION = "4.10.0"  
  
    def takdevVersion : String = '2.+'
```


Plugin from Scratch - Building

- ◆ Now that everything is setup we can do a build
- ◆ Open the Gradle Task Window and locate the *assembleCivDebug* task
- ◆ Double click this task to start a build of the plugin
 - ◆ This will only compile your plugin and not install it
- ◆ Correct any build errors and try again
- ◆ Once you are satisfied with you build, run *installCivDebug* to install the plugin in ATAK

Plugin from Scratch - Running

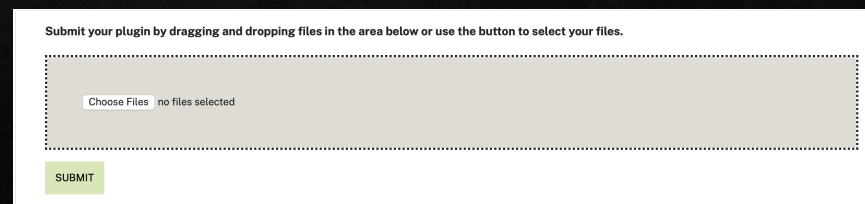
- ◆ Your plugin should now appear on the menu
- ◆ To change the icon, change the *res/drawable/ic_launcher.png*
 - ◆ This is defined in the *AndroidManifest.xml*
- ◆ Click on your icon to launch your plugin



Third Party Pipeline on tak.gov

Third Party Pipeline on tak.gov

- ◆ To have your plugin work on the full CIV version of ATAK it must be signed by the TAK Product Center (TPC) via the Third Party Pipeline (TPP)
- ◆ Simply zip the folder that contains your plugin and submit it to the TPP
 - ◆ In the previous example, we would zip the *SamplePlugin* folder
 - ◆ Recommend doing a *clean* before zipping to reduce the artifacts that are in the zip
- ◆ Login to tak.gov and goto Resources -> Third Party Pipeline
 - ◆ Upload the zip and submit
- ◆ You will receive an email notification:
 - ◆ When build begins
 - ◆ Complete of build (success or failure)
 - ◆ Failure will contain a build log that will need to be reviewed



Submit your plugin by dragging and dropping files in the area below or use the button to select your files.

Choose Files no files selected

SUBMIT

Troubleshooting

Troubleshooting

- ◆ If you are seeing old log statements that you have removed or weird behaviors in your plugin after several deployments during development, restart ATAK
 - ◆ This is due to the way that ATAK loads a plugin, it does not always unload the previous version of the Java class file. So, you could have old code still running.
 - ◆ This is only really an issue during development as we are constantly pushing new versions to the emulator/device.
 - ◆ A restart of ATAK clears out all the old class files.

Questions?

- ◆ Contact us: info@adeptus-cs.com
- ◆ Visit our website: www.adeptuscybersolutions.com