

# **SMAATSDK**

**INSTALLATION MANUAL FOR THE  
SMAATSDK ON ANDROID  
RELEASE v1.0**

**Serimag**

---

## Table of contents

Scope.....	3
Purpose.....	3
Introduction.....	3
IntelligengScanner Module integration.....	3
Post-Processing Module Integration.....	9
NFC Module integration.....	15

## Scope

This document contains an introduction and detailed explanation of all the steps that must be taken to correctly integrate Release v1.0 in Android Studio.

## Purpose

The purpose of this document is to help the customer to properly integrate Release v1.0 in an Android Studio 2.3.1 or later project.

## Introduction

The integration of Release v1.0 is made of two parts:

1. Integration of the models: ES\_ID\_FRONT, ES\_ID\_BACK, ES\_ID\_23, ES\_RESIDENCE\_FRONT, ES\_RESIDENCE\_BACK and EU\_CHECK\_FRONT. Integration of the *“engine.cfg”* and *“post.cfg”* files is also required. This part is explained in detail in the document "IntelligentScanner Module Requirements and documentation.pdf".
2. Integration of AAR files in IntelligentScanner (intelligentscannerlibrary.aar), Post-Processing (postprocesslibrary.aar) and the NFC Module (nfclibrary.aar) cases.

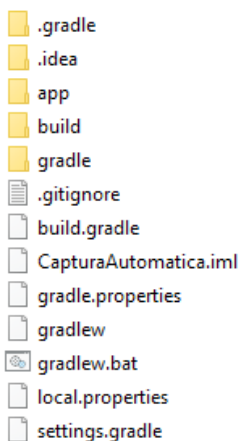
A series of steps are explained below for integrating the AARs in a Android Studio 2.3.1 or later project.

## IntelligentScanner Module integration

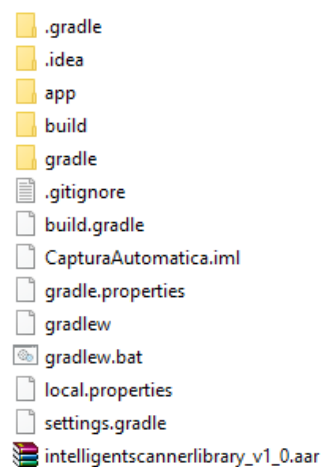
For IntelligentScanner Module integration, take the following steps:

1. Paste the intelligentscannerlibrary\_v1\_0.aar file into the project folder.

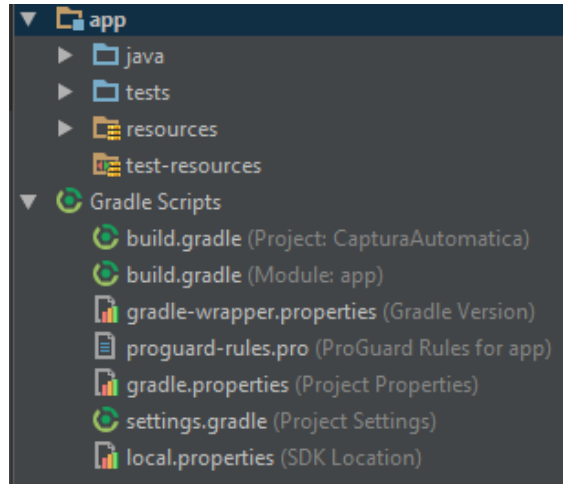
*Before copying the aar*



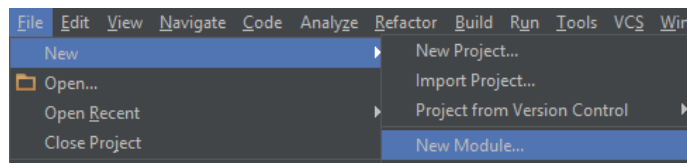
*After copying the aar*



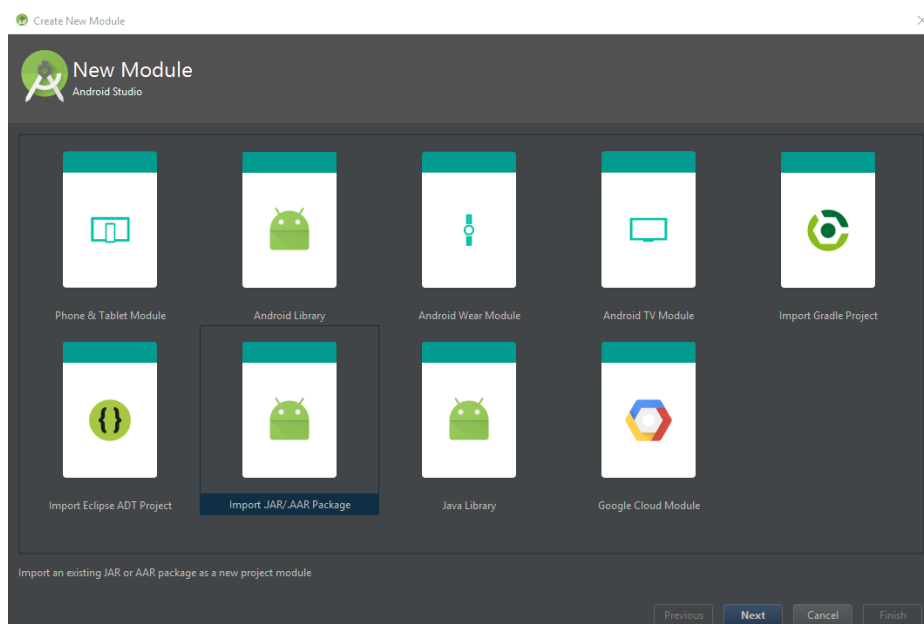
2. Then, open the project with Android Studio 2.3.1 or later and proceed to integrate the aar. In this case, we have a simple project that has the application folders and Gradle.



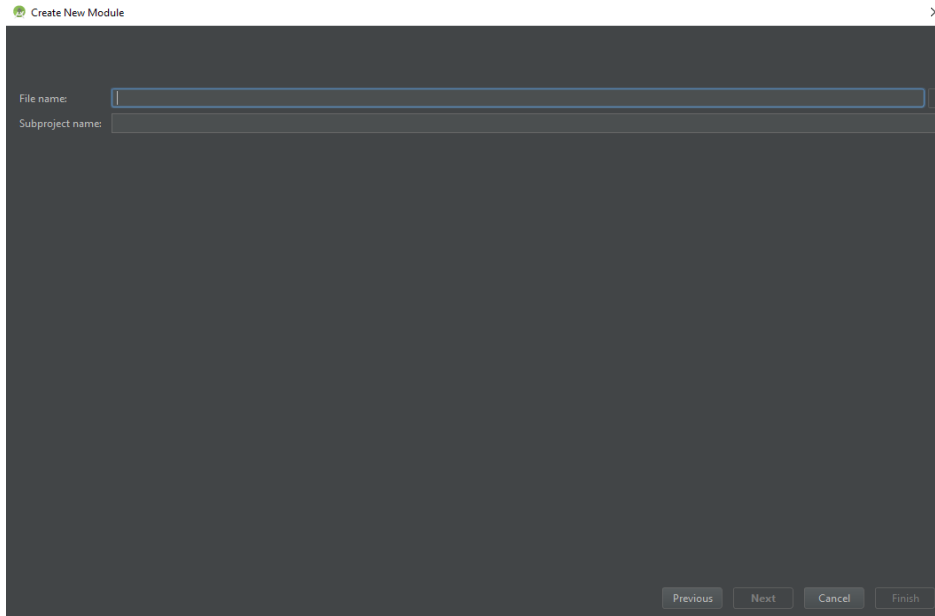
3. Click on **“File”** → **“New”** → **“New Module”**



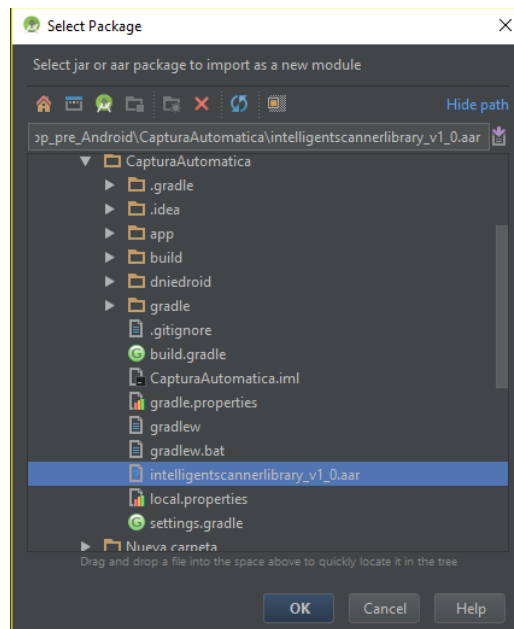
and the following window will appear:



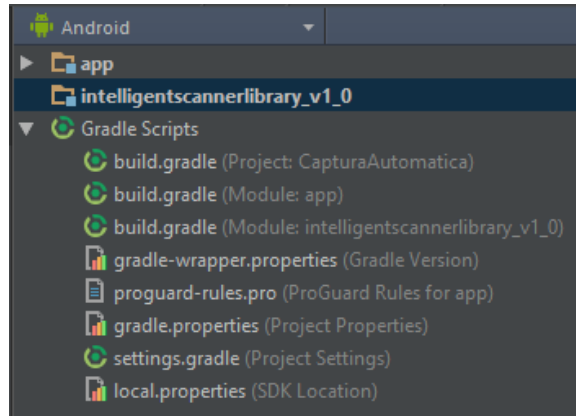
4. Select the **“Import JAR/AAR.Package”** → **“Next”** option, and the following window will appear:



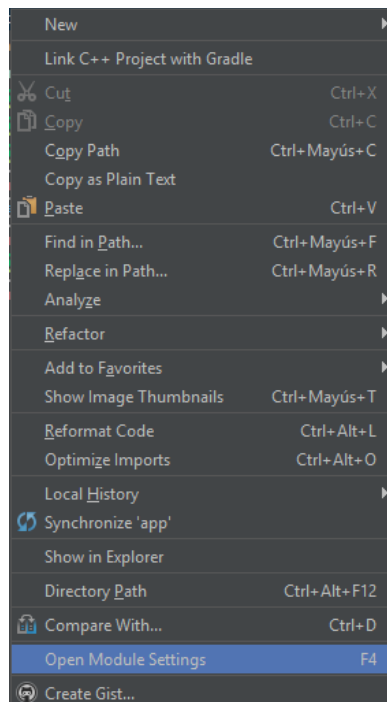
5. In the "File name" section, select the **“intelligentscannerlibrary\_v1\_0.aar”** file from the project folder:



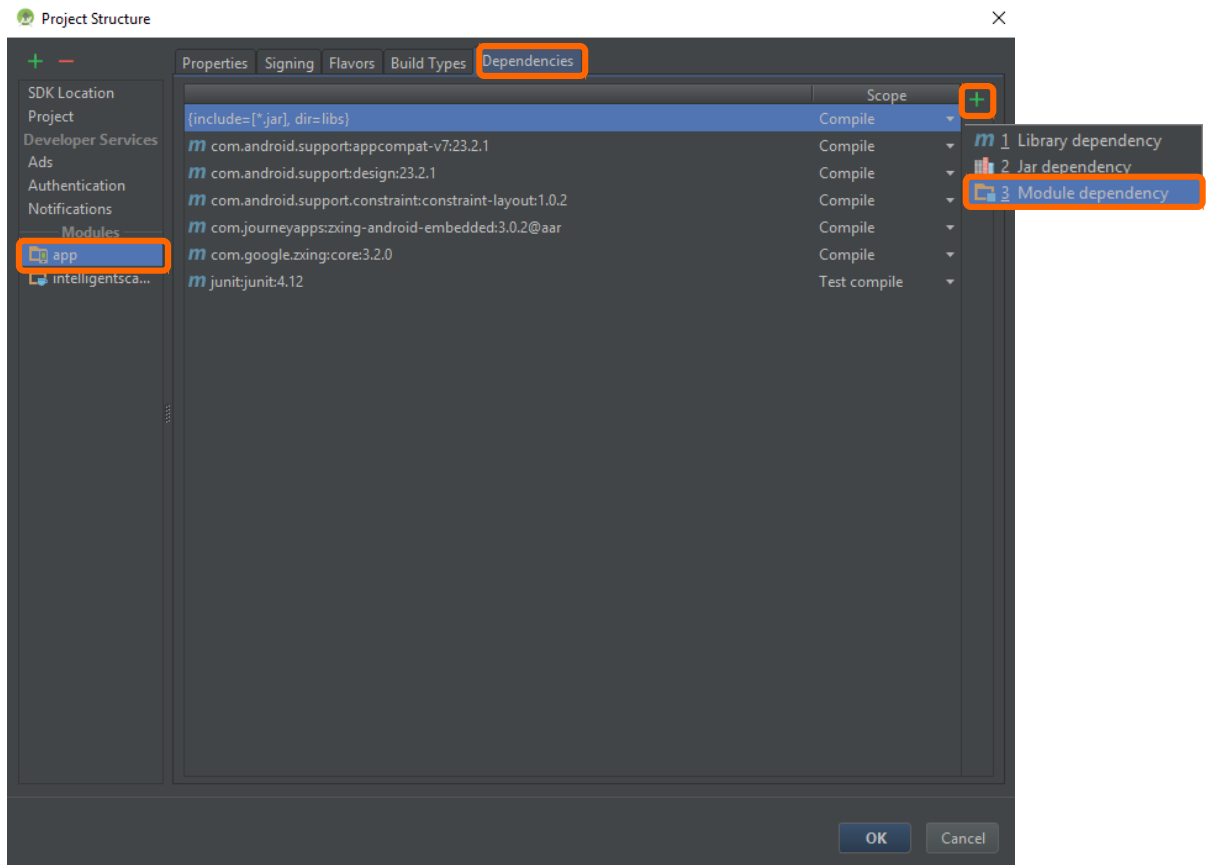
6. In the project folders, a new folder should be automatically added called `intelligentscannerlibrary_v1_0`, as you can see in the following capture:



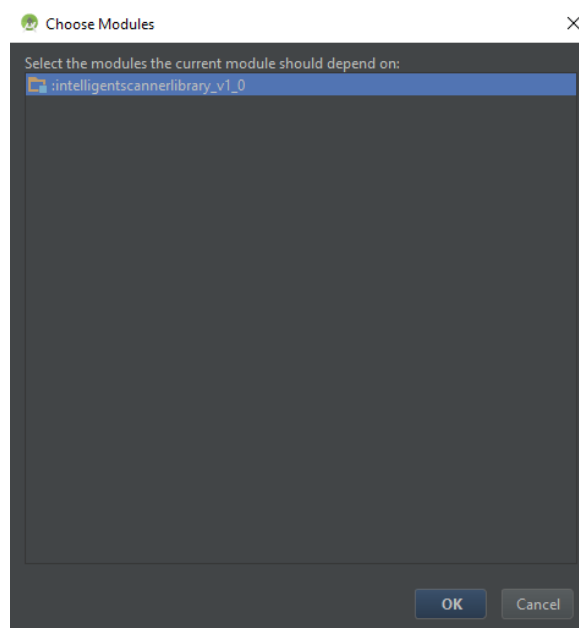
7. Once the aar file is integrated, its dependencies should be added. To do this, access them by **right clicking** on the "app" folder → "Open Module Settings".



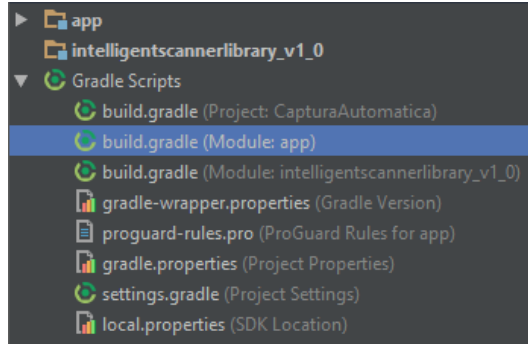
The following window will then be displayed:



Click on: **“app”** → **“Dependencies”** → **“+”** → **“Module dependency”**, select the **intelligentscannerlibrary\_v1\_0** module.



8. Finally, check that the dependencies have been added correctly. To do this, access the **“Gradle Scripts”** → **“build.gradle (Module:app)”** section:



In the dependencies section, check that the

```
compile project(':intelligentscannerlibrary_v1_0')
```

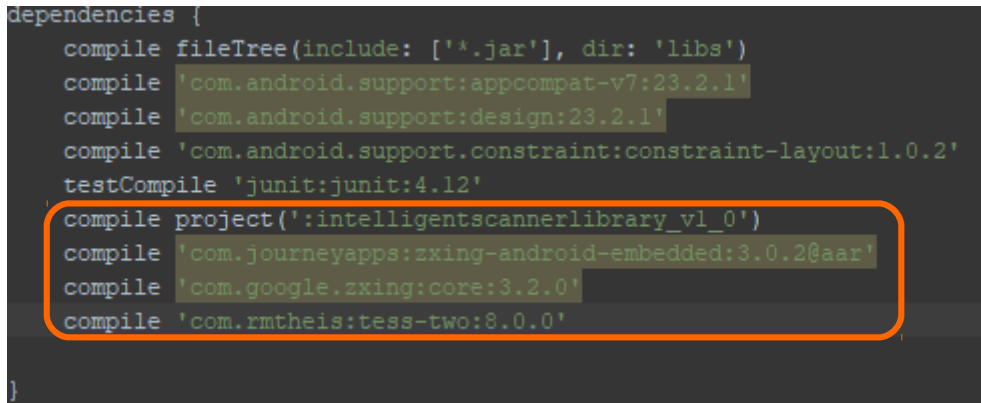
line has been added correctly. If not, add it manually.

```
dependencies {  
    compile fileTree(include: ['*.jar'], dir: 'libs')  
    compile 'com.android.support:appcompat-v7:23.2.1'  
    compile 'com.android.support:design:23.2.1'  
    compile 'com.android.support.constraint:constraint-layout:1.0.2'  
    testCompile 'junit:junit:4.12'  
    compile project(':intelligentscannerlibrary_v1_0')  
}
```



9. The IntelligentScanner module uses a series of external libraries. The following lines should be added manually:

```
compile 'com.journeyapps:zxing-android-embedded:3.0.2@aar'  
compile 'com.google.zxing:core:3.2.0'  
compile 'com.rmtheis:tess-two:8.0.0'
```



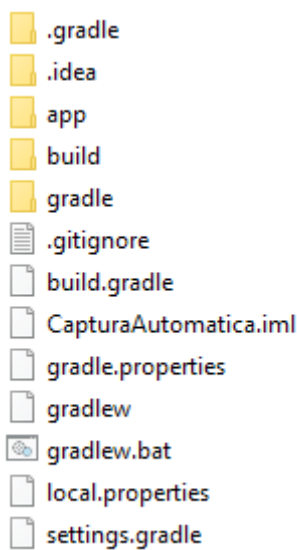
```
dependencies {  
    compile fileTree(include: ['*.jar'], dir: 'libs')  
    compile 'com.android.support:appcompat-v7:23.2.1'  
    compile 'com.android.support:design:23.2.1'  
    compile 'com.android.support.constraint:constraint-layout:1.0.2'  
    testCompile 'junit:junit:4.12'  
    compile project(':intelligentscannerlibrary_v1_0')  
    compile 'com.journeyapps:zxing-android-embedded:3.0.2@aar'  
    compile 'com.google.zxing:core:3.2.0'  
    compile 'com.rmtheis:tess-two:8.0.0'  
}
```

## Post-Processing Module Integration

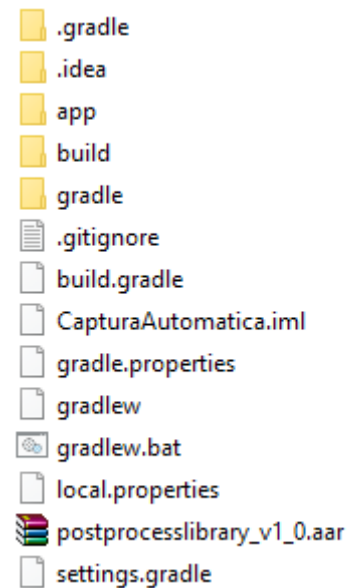
For Post-Processing Module integration, take the following steps:

1. Paste the postprocesslibrary\_v1\_0.aar file into the project folder.

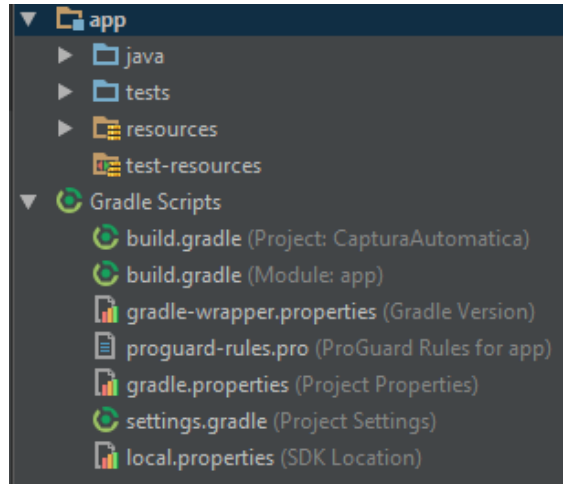
*Before copying the aar*



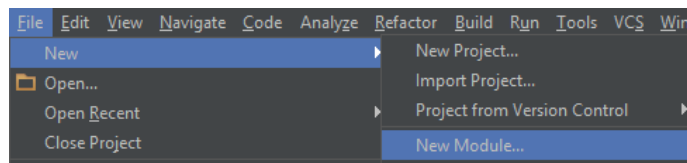
*After copying the aar*



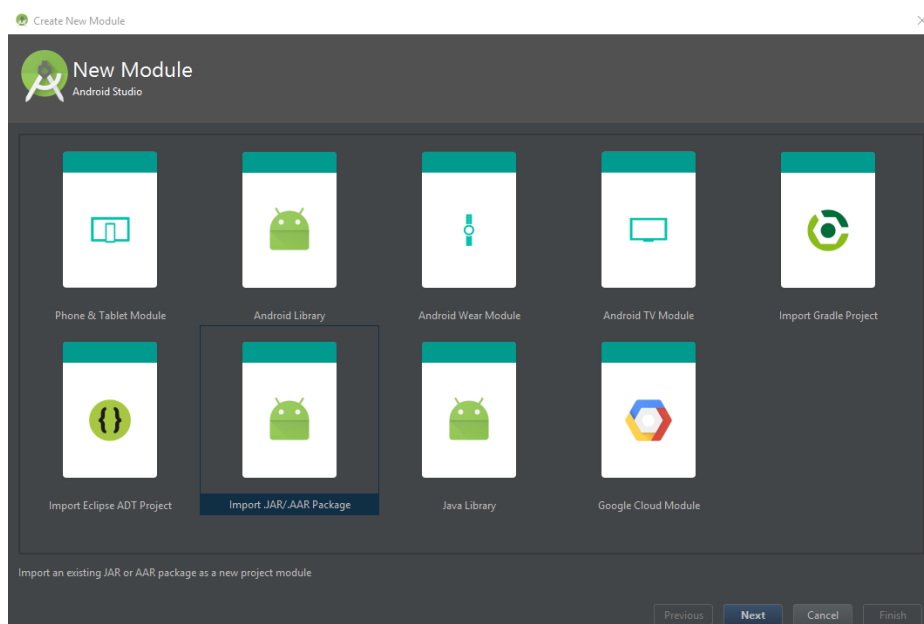
2. Then, open the project with Android Studio 2.3.1 or later and proceed to integrate the aar. In this case, we have a simple project that has the application folders and Gradle.



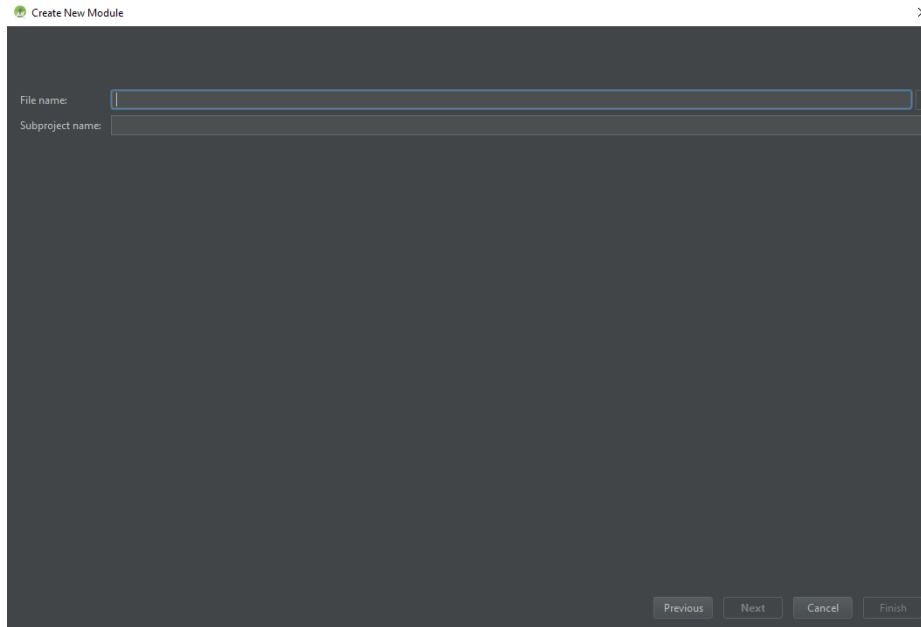
3. Click on **"File" → "New" → "New Module"**



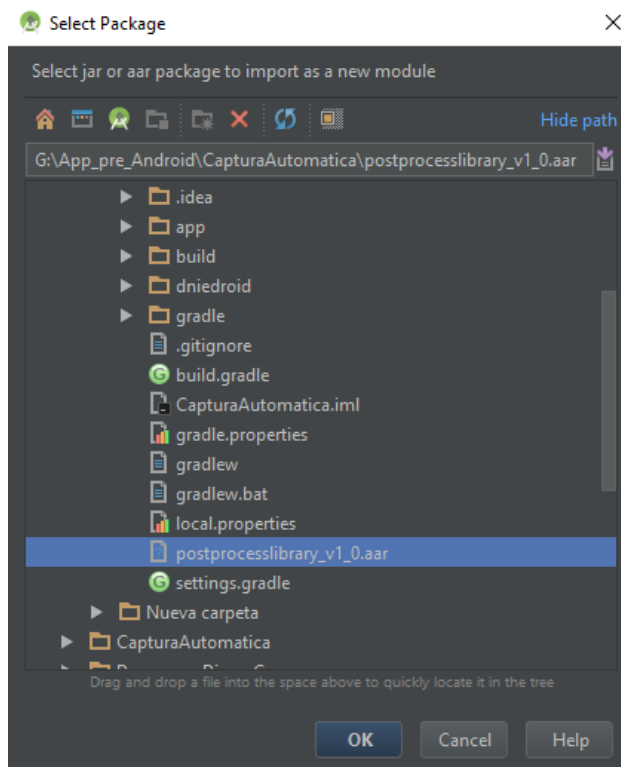
and the following window will appear:



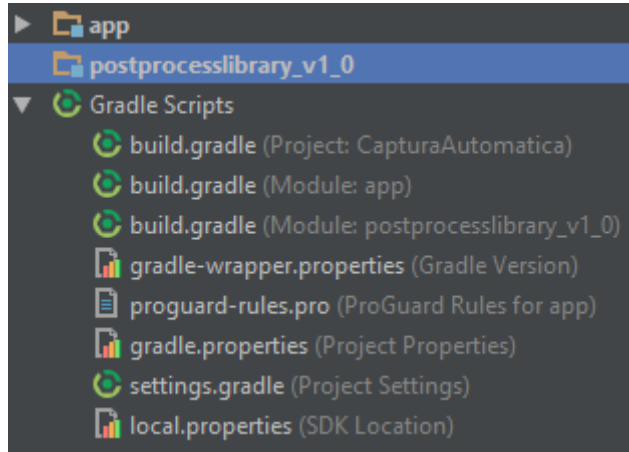
4. Select the **“Import JAR/AAR.Package”** → **“Next”** option, and the following window will appear:



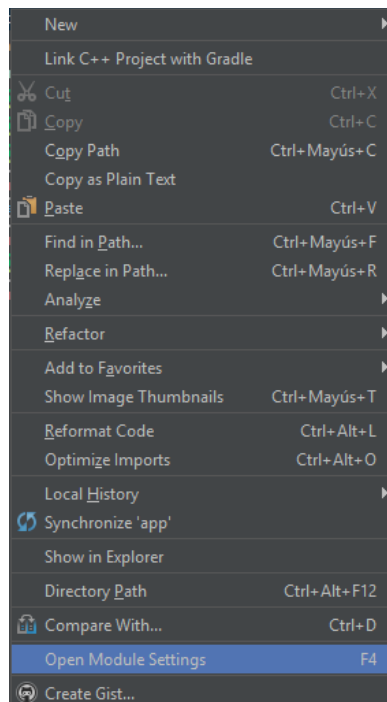
5. In the "File name" section, select the **“postprocesslibrary\_v1\_0.aar”** file from the project folder:



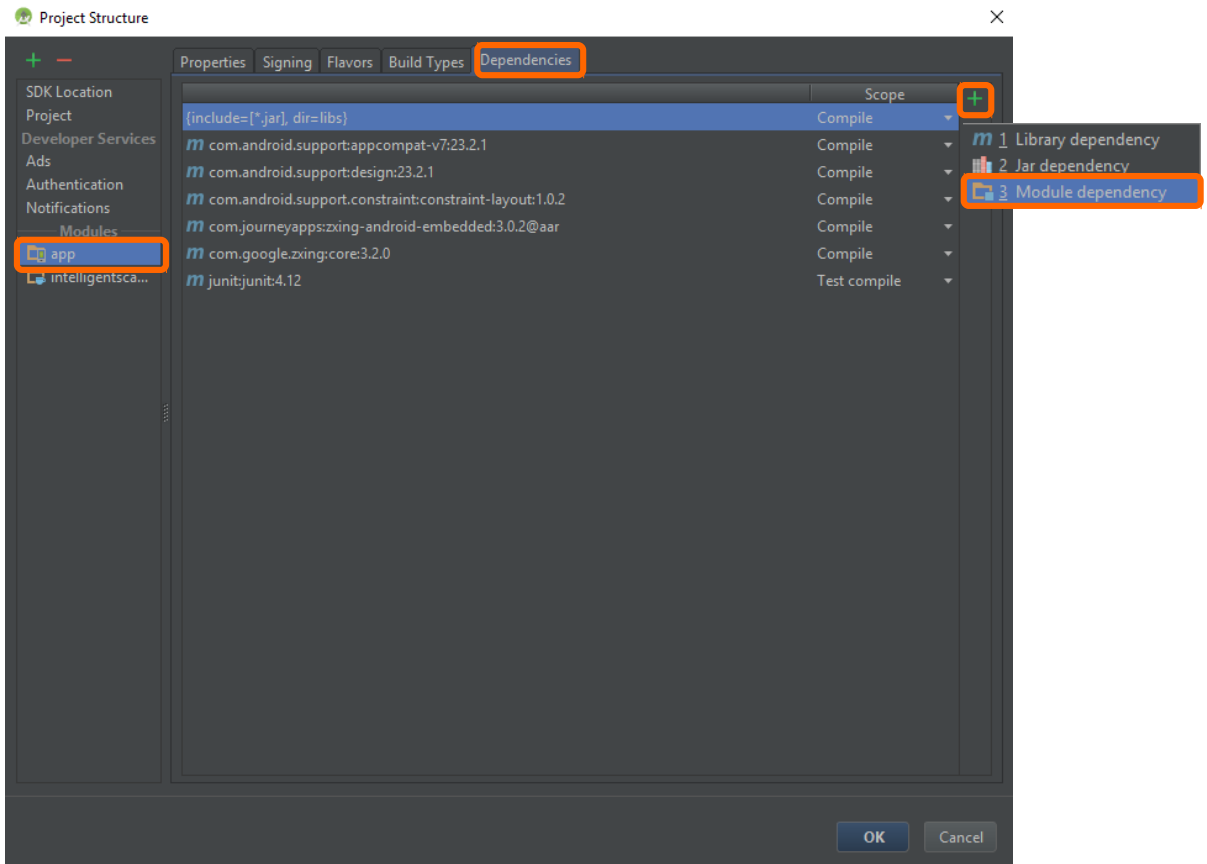
6. In the project folders, a new folder should be automatically added called `postprocesslibrary_v1_0`, as you can see in the following capture:



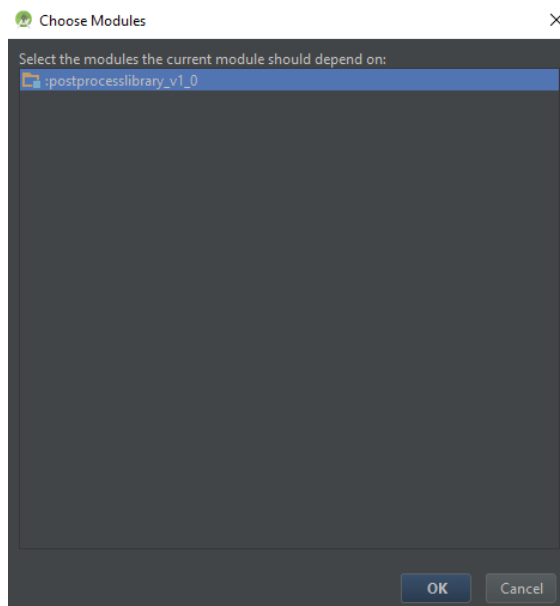
7. Once the aar file is integrated, its dependencies should be added. To do this, access them by **right clicking** on the "app" folder → "Open Module Settings"



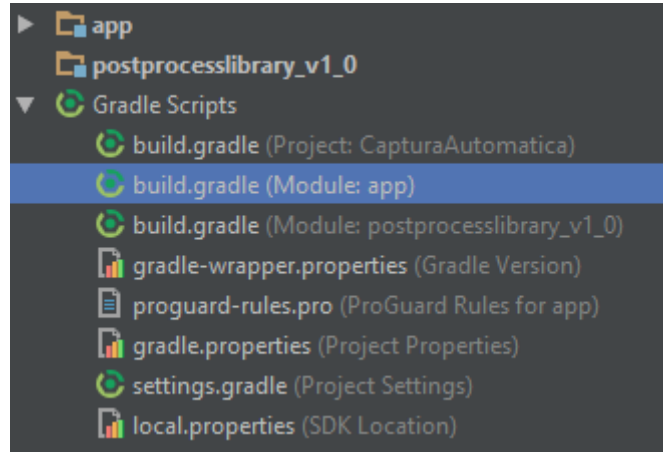
The following window will then be displayed:



Click on: "app" → "Dependencies" → "+" → "Module dependency", select the postprocesslibrary\_v1\_0 module.



8. Finally, check that the dependencies have been added correctly. To do this, access the “**Gradle Scripts**” → “**build.gradle (Module:app)**” section:



In the dependencies section, check that the

```
compile project(':postprocesslibrary_v.1.0')
```

line has been added correctly. If not, add it manually.

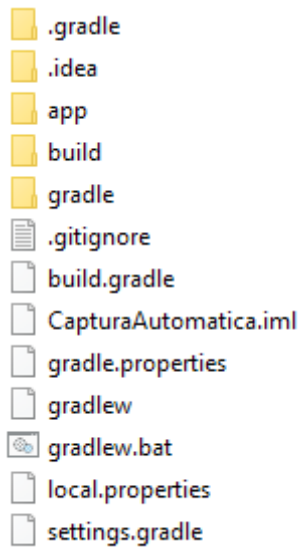
```
dependencies {  
    compile fileTree(include: ['*.jar'], dir: 'libs')  
    compile 'com.android.support:appcompat-v7:23.2.1'  
    compile 'com.android.support:design:23.2.1'  
    testCompile 'junit:junit:4.12'  
    compile 'com.android.support.constraint:constraint-layout:1.0.2'  
    compile project(':postprocesslibrary_v1_0')  
}
```

## NFC Module integration

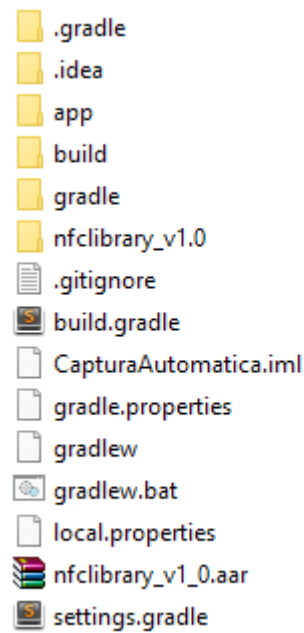
For Post-Processing module integration, take the following steps:

1. Paste the `nfclibrary_v1_0.aar` and `dniedroid.aar` files into the project folder.

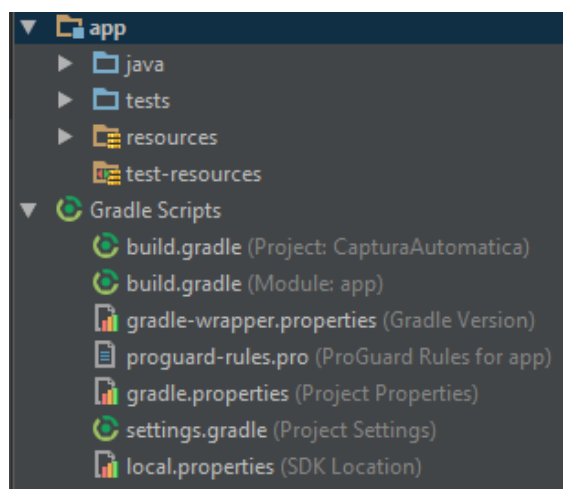
*Before copying the aar*



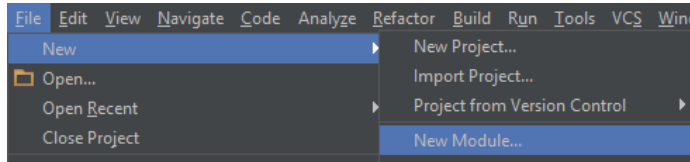
*After copying the aar*



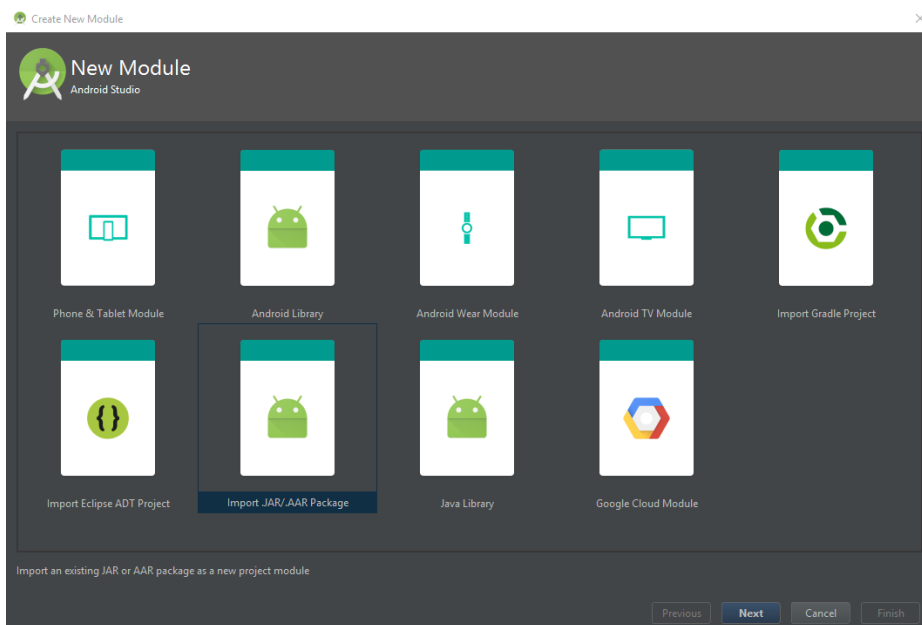
2. Then, open the project with Android Studio 2.3.1 or later and proceed to integrate the aar. In this case, we have a simple project that has the application folders and Gradle.



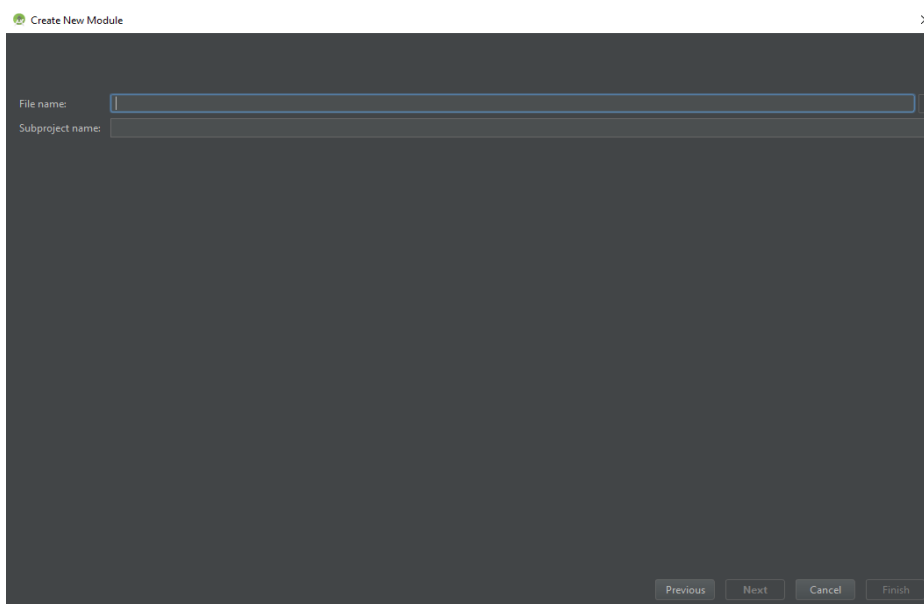
3. Click on **“File”** → **“New”** → **“New Module”**,



and the following window will appear

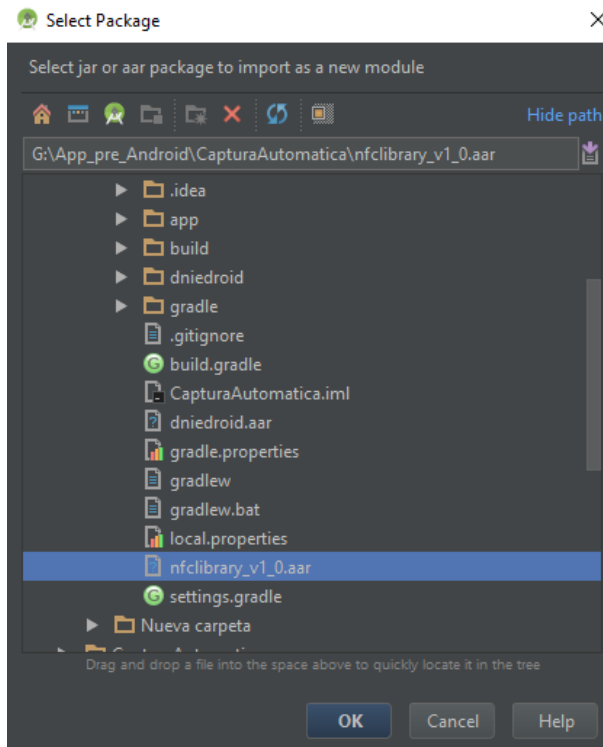


4. Select the **“Import JAR/AAR.Package”** → **“Next”** option, and the following window will appear:

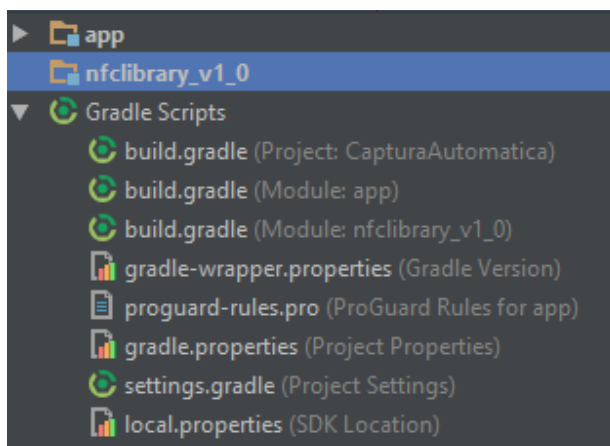




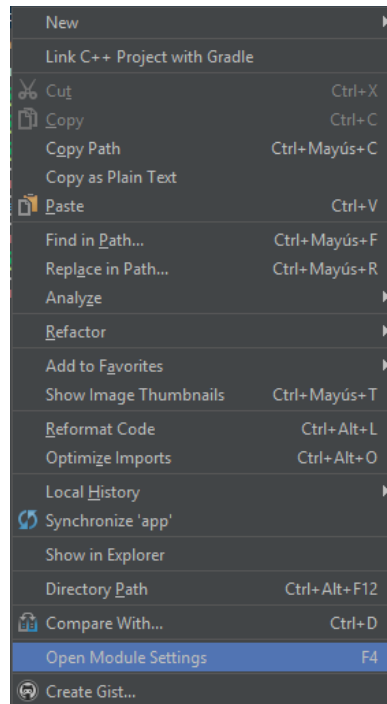
5. In the "File name" section, select the "nfclibrary\_v1\_0.aar" file from the project folder:



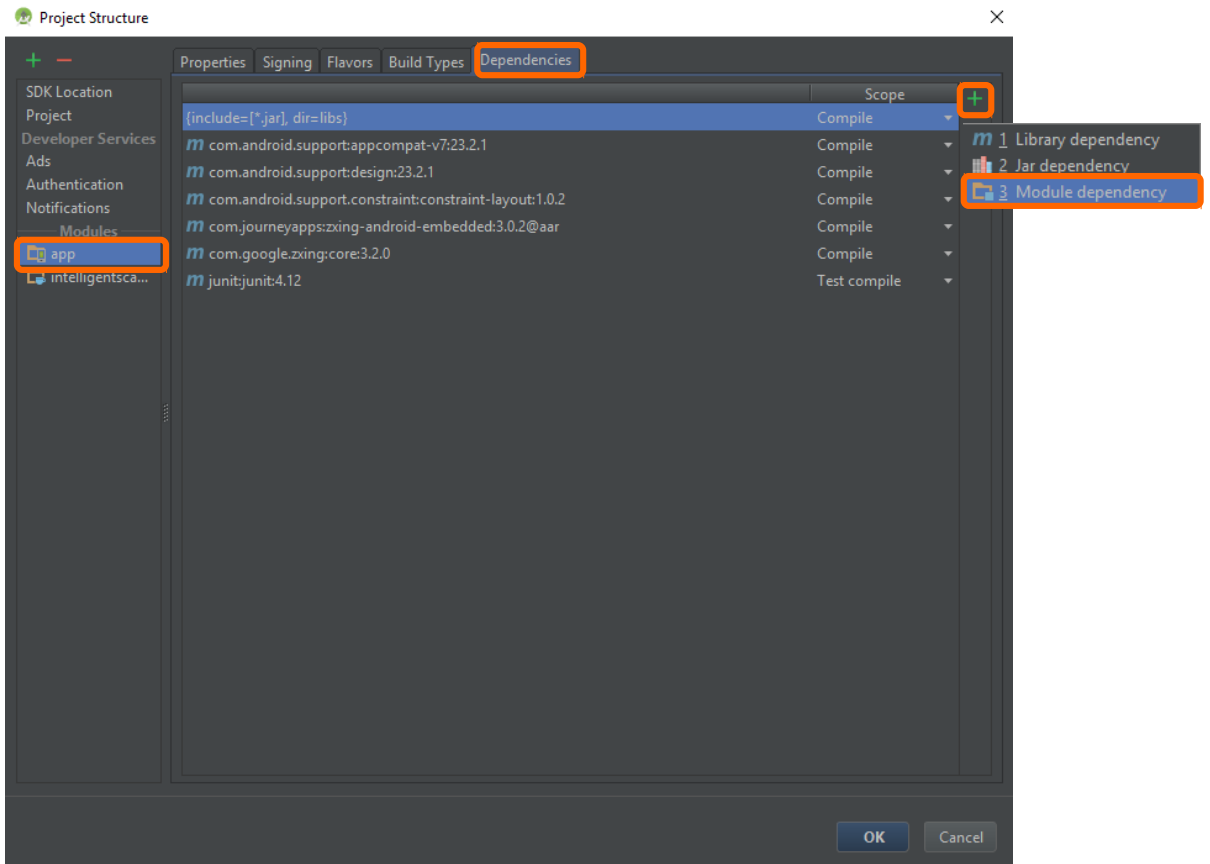
6. In the project folders, a new folder should be automatically added called nfclibrary\_v1\_0, as you can see in the following capture:



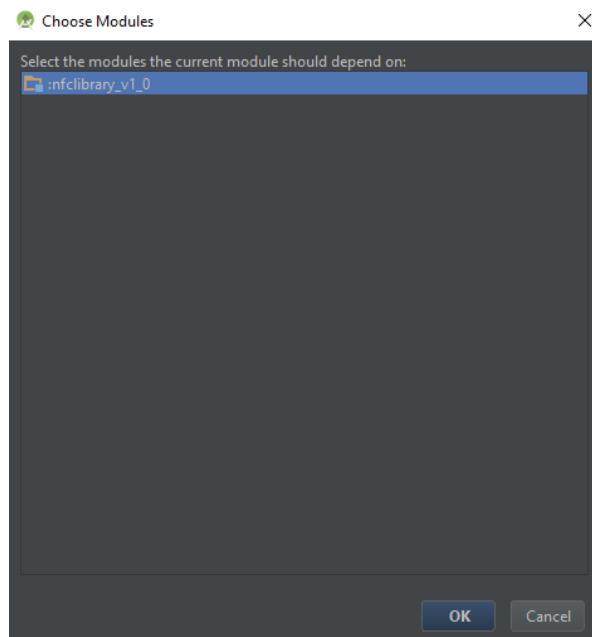
7. Once the aar file is integrated, its dependencies should be added. To do this, access them by **right clicking** on the "app" folder → "Open Module Settings".



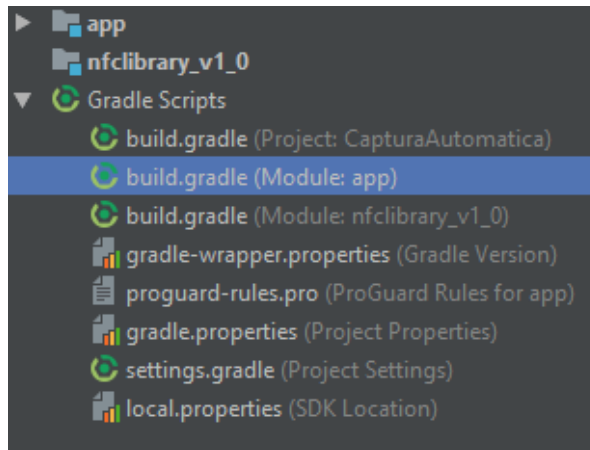
The following window will then be displayed:



By clicking on: “**app**” → “**Dependencies**” → “**+**” → “**Module dependency**”, select the `nfclibrary_v1_0` module.



8. Finally, check that the dependencies have been added correctly. To do this, access the “**Gradle Scripts**” → “**build.gradle (Module:app)**” section:



In the dependencies section, check that the

```
compile project(':nfclibrary_v1_0')
```

line has been added correctly. If not, add it manually.

```
dependencies {  
    compile fileTree(include: ['*.jar'], dir: 'libs')  
    compile 'com.android.support:appcompat-v7:23.2.1'  
    compile 'com.android.support:design:23.2.1'  
    compile 'com.android.support.constraint:constraint-layout:1.0.2'  
    testCompile 'junit:junit:4.12'  
    compile project(':nfclibrary_v1_0')  
}
```