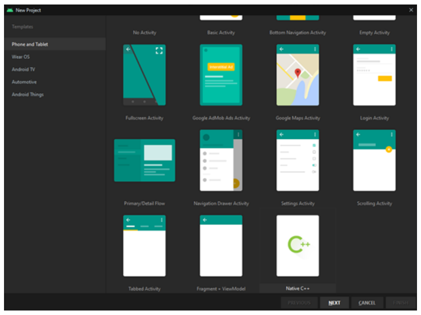
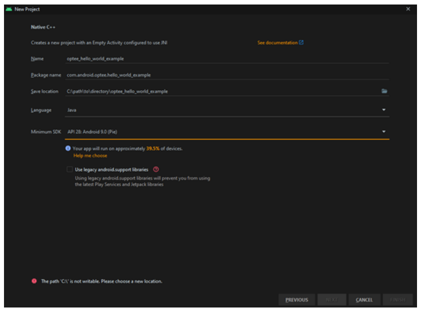
1 Procedure

1.1 App Creation

Start by creating a new project that follows the Native C++ template. After that, you must fill in with information regarding the project, and in the Minimum SDK field you must specify that you intend to use the API 28: Android 9.0 (Pie). Both steps are presented in figure 1.



(a)



Native C++

Creates a new project with an Empty Activity configured to use JNI

Name: optee_hello_world_example

Package name: com.android.optee.hello_world_example

Save location: C:\path\to\directory\optee_hello_world_example

Language: Java

Minimum SDK: API 28: Android 9.0 (Pie)

Your app will run on approximately 39.5% of devices.

[ ] Use legacy android.support libraries

Using legacy android.support libraries will prevent you from using the latest Play Services and Jetpack libraries

The path 'C:\' is not writable. Please choose a new location.

(b)

Figure 1: Project Creation Steps

The project will be generated and you will have a simple app that prints a "Hello World" in the main screen.

Our objective is now to move the OP-TEE Hello World [1] main file located in the host directory to the cpp directory. Since the original file will have to suffer some changes in order to be used in our app, I recommend you to use the one located at `app/app/src/main/cpp/hello_word.c`. Next, in the CMakeLists, you must replace your new added file with the pre-generated `native-lib.cpp` file, so that the project can use its functions. Additionally, you must call the function specified in the .c file, so that we can test if the app communicated with OP-TEE.

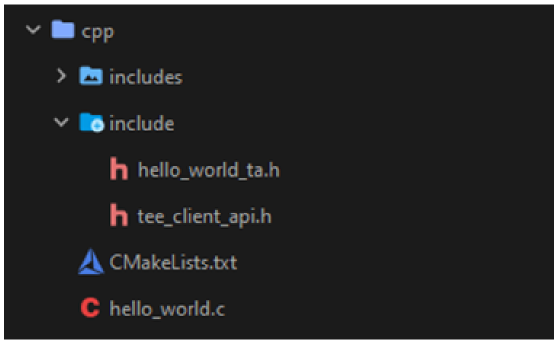
In the following step, you must create a directory and put inside the header files the `hello_world.c` needs. Your Project structure at the moment must resemble figure 2.

After synchronizing your project, you will notice that the `tee_api.h` functions are still invalid to use. This means you must include the library file that has the functions' implementation. So, we must create a new directory and add the `libteec.so` file generated during the AOSP+OPTEE build process. The file can be found at `out/target/product/hikey960/vendor/lib/libteec.so`.

Our cpp directory structure should now look like figure 4.

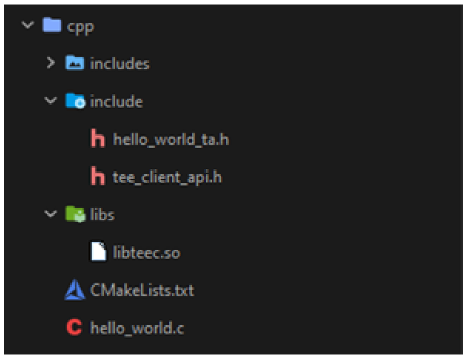
For the file to be recognized, you must add the `libteec.so` file as an external library to CMakeList file. I recommend you see how I did it (file location: `app/app/src/main/cpp/CMakeList`).

The process of building the app is almost done, but you must complete this last step. In the `build.gradle` file, you must specify the ndk abiFilters



```
cpp
  includes
  include
    hello_world_ta.h
    tee_client_api.h
  CMakeLists.txt
  hello_world.c
```

Figure 2: Project Organization (1)



```
cpp
  includes
  include
    hello_world_ta.h
    tee_client_api.h
  libs
    libteec.so
  CMakeLists.txt
  hello_world.c
```

Figure 3: Project Organization (2)

the android studio will use to build your application. The only version the `libteec.so` file currently supports is the "arm64-v8a", so you must write it in the ndk field, inside the android field, as illustrated in figure 4.

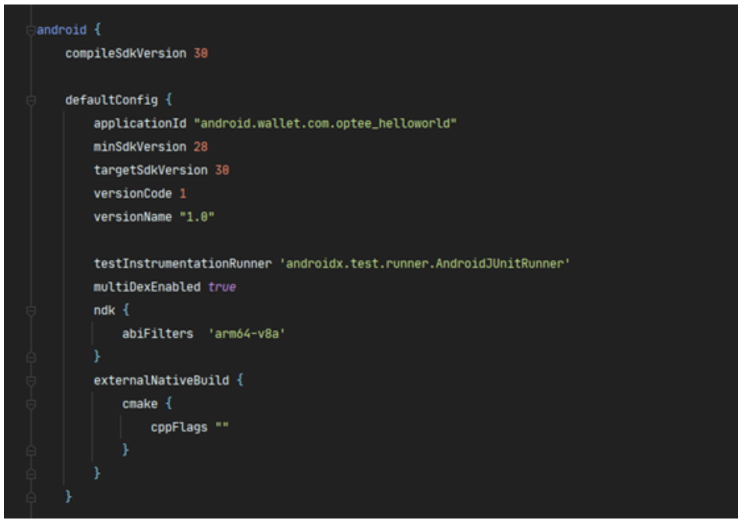
With this, you can build your app and generate an apk. The next step should be its installation in the Android system.

1.2 Install App

To run the App in the AOSP+OPTEE system successfully you need to install the app as a system app, otherwise the application will not be able to communicate with OP-TEE. To do this, you must type the following set of commands:

```
$ adb root
$ adb remount
$ adb push filename.apk  /system/app/
$ adb shell chmod 644 /system/app/ filename.apk
$ adb reboot
```

Finally, you can disconnect the board from your computer and run the app.



```
android {
    compileSdkVersion 30

    defaultConfig {
        applicationId "android.wallet.com.optee_helloworld"
        minSdkVersion 28
        targetSdkVersion 30
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner 'androidx.test.runner.AndroidJUnitRunner'
        multiDexEnabled true
        ndk {
            abiFilters  'arm64-v8a'
        }
        externalNativeBuild {
            cmake {
                cppFlags ""
            }
        }
    }
```

Figure 4: build.graddle file

If everything was correctly done, the app should run just fine, and the OP-TEE Hello World example must output something in your display.

References

[1] Linaro. Op-tee hello world example. 2021.