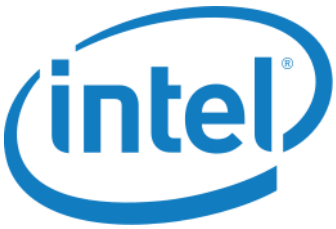
Intel Software Platform for Curie

Android Getting Started Guide

Version 1.0

January 2016

BETA DRAFT



Contents

List of Code Samples

Code Samples

CHAPTER

1 About This Guide

This guide provides information for setting up your environment to work with Intel Software Platform for Curie on Android.

This chapter contains the following sections:

1.1 Who Should Read This Book

This guide is designed for Android developers.

To use the Intel Curie Software Platform for Curie SDK, Android developers should have proficiency in Java and experience developing mobile applications for the Android mobile platform.

1.2 Terminology Used in This Guide

Table 1: Terms and Definitions

Term	Definition
Android	Android mobile operating system
Gradle	Open-source build automation system
Github	Web-based Git repository
JDK	Java Development Kit
Maven	Repository containing artifacts you may require to compile, test, package, perform integration testing, or deploy a mobile application
Wearable Platform	Intel Sofware Platform for Curie

1.3 Additional Resources

The following documents are included in the Wearable Platform SDK to help you get started:

- *Intel Curie Platform Software Android Developer's Guide*
- *Wearable Platform Javadoc API Reference for Android*
- *Cloud Services Portal Administrator Guide*
- *Intel® Curie™ Platform Customer Reference Board (CRB) Hardware User Guide*
- *Intel® Curie™ Platform Hardware User's Guide*
- *Intel® Curie™ Platform Software User's Guide*

CHAPTER

2 Setting Up Your Build Environment

Intel Software Platform for Curie provides a basic Hello World Android Studio project that is properly configured to use the Wearable Platform Core APIs. The Wearable Platform Core APIs implement services essential to managing and communicating with all Curie-based wearable devices.

Tip: In this guide, "Intel Software Platform for Curie" is sometimes called "Wearable Platform" for better readability.

You can use the Hello World project to start your own project in Android Studio.

This document contains the following sections:

- About the Hello World Reference Project
- Prerequisites
- Installing the hello_world Reference Application
- Building Your Own Application

For more information about the Wearable Platform Core APIs, see the *Intel Curie Platform Software Android Developer's Guide.*

2.1 About the Hello World Reference Project

The Hello World project references 2 pre-built `.aar` files in the local filesystem, and a number of third-party dependencies are automatically fetched by Gradle when you sync the project.

When running the Hello World project, it will invoke `Core.init()` and display the result of this call in the Main Activity window. After the `init()` call is successful, you're ready to start using the Wearable Platform Core APIs.

If you prefer to build an application yourself, you can refer to the sample application `hello_world` to copy information into a new project.

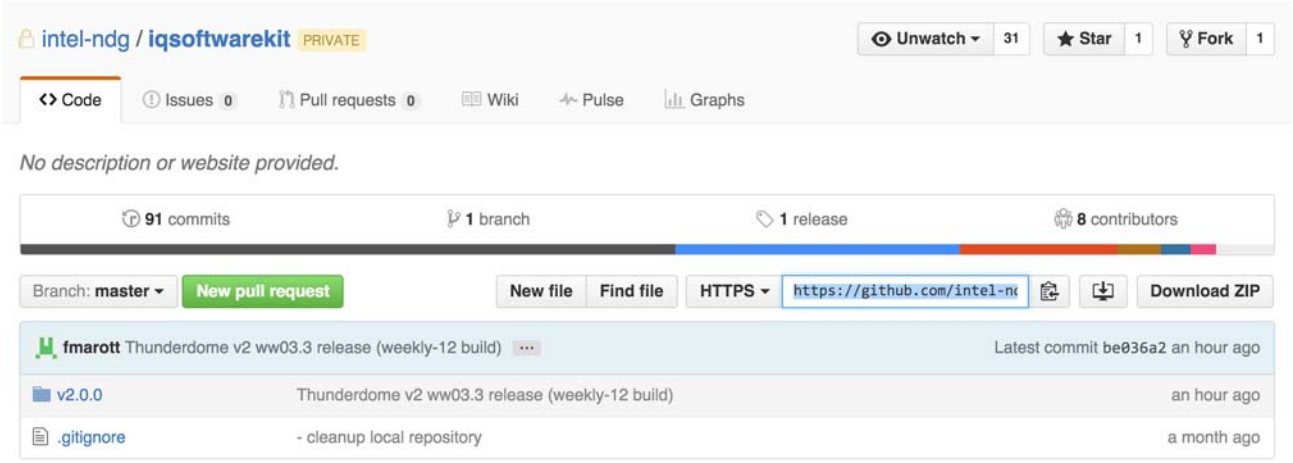
2.2 Prerequisites

- Gradle version 2.4 or greater
- JDK version 7 or greater
- Latest Android Studio

2.3 Installing the hello_world Reference Application

Step 1. Login to Github to access the Git source code repository.

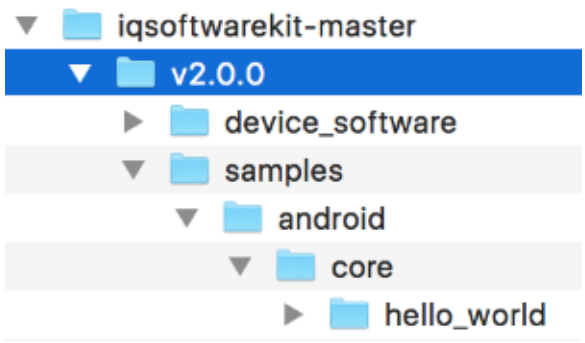
Step 2. Clone this Github repository: `https://github.com/intel-ndg/iqsoftwarekit.git`



Step 3. (Optional) Download the zip file to make a local copy.

Step 4. Open Android Studio, and import the `hello_world project`.

Step 5. Navigate to the `hello_world` project folder `v2.0.0/samples/android/core/hello_world.`



Step 6. To sync the project, click the Sync icon.

Android Studio should build the project and fetch dependencies automatically.

Step 7. Upon success, you can run the application in the Android Studio application emulator.

You will see text saying "Core initialized" if Core SDK has initialized properly.

You are ready to start using the Wearable Platform Core APIs.

2.4 Building Your Own Application

You can also create a new project in Android Studio, and specify the dependencies.

Tip: Open the `hello_world` *application project to view information that you need to copy to your new blank project.*

Step 1. Create a new project.

Step 2. Edit the top level `build.gradle` file to add a local Maven repository. Under all projects add the path to the Maven repository.

Example: **Adding Path to the Maven Repository**

```
allprojects {
    repositories {
        jcenter()
        maven { url "${rootDir}/../../../../sdk/android/maven-repo" }
    }
}
```

Step 3. In your main module `build.gradle` file, declare the dependencies as in the following example.

***Example:* Declaring Gradle Dependencies**

```
compile "com.android.support:appcompat-v7:$GOOGLE_SUPPORT_APPCOMPAT_V7_VERSION"
compile "com.android.support:design:23.1.0"
compile "com.android.support:support-annotations:$GOOGLE_SUPPORT_ANNOTATIONS_VERSION"
compile "org.apache.commons:commons-lang3:$APACHE_COMMONS_LANG3_VERSION"
compile "com.jakewharton.timber:timber:$JAKEWHARTON_TIMBER_VERSION"
compile "com.twofortyfouram:android-assertion:$TWOFORTYFOURAM_ANDROID_ASSERTION_VERSION"
compile "com.google.protobuf:protobuf-java:$GOOGLE_PROTOBUF_VERSION"
compile "io.realm:realm-android:$REALM_VERSION"
compile "com.intel.wearable.platform.icsf:icsflib:$ICSF_LIB_VERSION@aar"
compile "com.intel.wearable.platform:core:$WEARABLE_CORE_SDK_VERSION_NAME@aar"
```

Step 4. In your `gradle.properties` file, declare constant values as in the following example.

***Example:* Declaring Constant Values in gradle.properties**

```
GOOGLE_SUPPORT_ANNOTATIONS_VERSION=22.2+
ANDROID_TOOLS_GRADLE_VERSION=1.3.1
APACHE_COMMONS_LANG3_VERSION=3.4
JAKEWHARTON_TIMBER_VERSION=3.1.0
WEARABLE_CORE_SDK_VERSION_NAME=0.3.3
TWOFORTYFOURAM_ANDROID_ASSERTION_VERSION=[1.0.2,2.0[
GOOGLE_PROTOBUF_VERSION=3.0.0-alpha-2
ICSF_LIB_VERSION=0.1.7
GOOGLE_SUPPORT_APPCOMPAT_V7_VERSION=22.2+
REALM_VERSION=0.86.0
```

Step 5. Explore the `Application.init()`method to see how `Core.init()`is invoked.

***Example:* Invoking Core.init()**

```
public static final void init(ICoreInitListener initListener){
    if (!mIsInitialized) {
        mIsInitialized = true;
        byte[] key = new byte[64]; // using an empty key for simplicity
        Core.init(getContext(), initListener, key);
    }
}
```

Step 6. Invoke the `Application.init()` method.

Example: **Processing the ICoreInitListener.onInitialized() Callback**

```
Application.init(new ICoreInitListener() {
    @Override
  public void onInitialized() {
        Logger.d("Core.onInitialized() in Application");
        helloTextView.setText("Core initialized.");
    }
});
```