



ATID Co.,Ltd

RFID Programming Guide

Android Developer Guide

SW Team

2021-07-14

개정 이력

| 버전 | Date | Reason ¹ | Description ² | Writer |
|------|------------|---------------------|--------------------------|---------|
| V1.0 | 2021-07-14 | 초안 | 신규 생성 | SW Team |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

¹ 개정사유 : 제정 또는 개정 내용이 이전 문서에 대해 추가/수정/삭제인지 선택 기입

² 개정내역 : 개정이 발생하는 페이지 번호와 변경 내용을 기술

목차

| | |
|--|----|
| 목차 | 3 |
| 1. 개요 | 4 |
| 2. Development environment setup | 5 |
| 2.1. Create a project. | 5 |
| 3. Programing Guide..... | 10 |
| 3.1. 초기화 | 10 |
| 3.1.1 객체 생성 | 10 |
| 3.1.2 Event Listener 등록/해제 | 10 |
| 3.2. Module power 관리. | 11 |
| 3.3. Event Handler | 12 |
| 3.4. Start and Stop Inventory | 13 |
| 3.4.1 Start Inventory | 13 |
| 3.4.2 Stop Inventory..... | 14 |
| 3.5. 종료 | 14 |

1. 개요

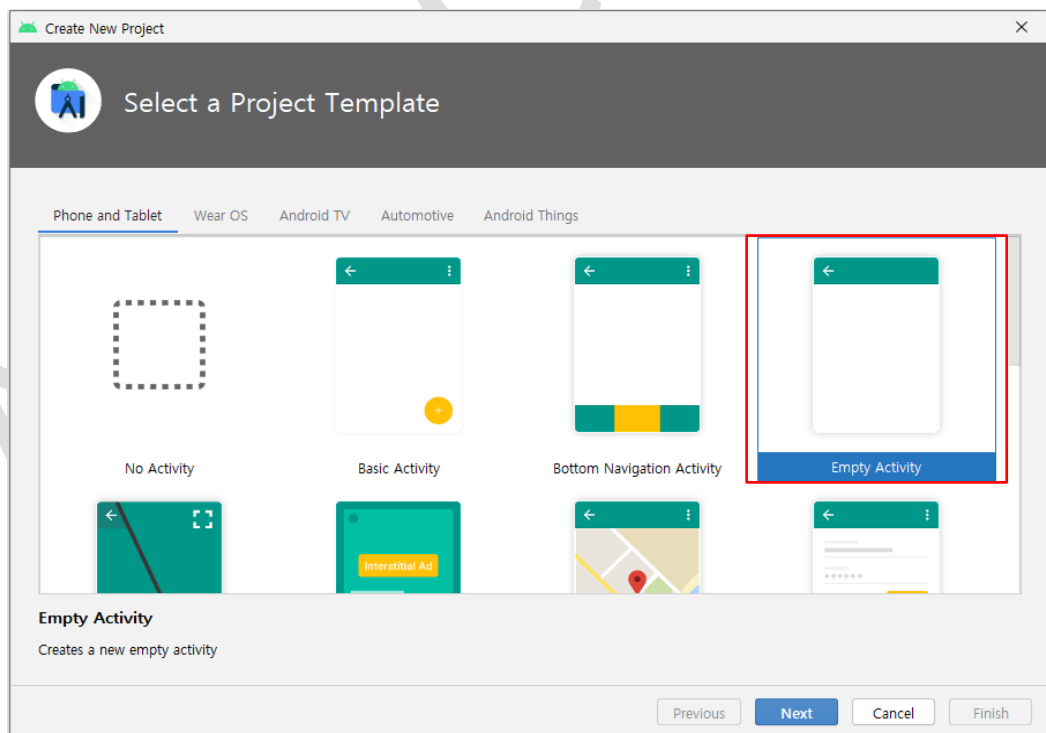
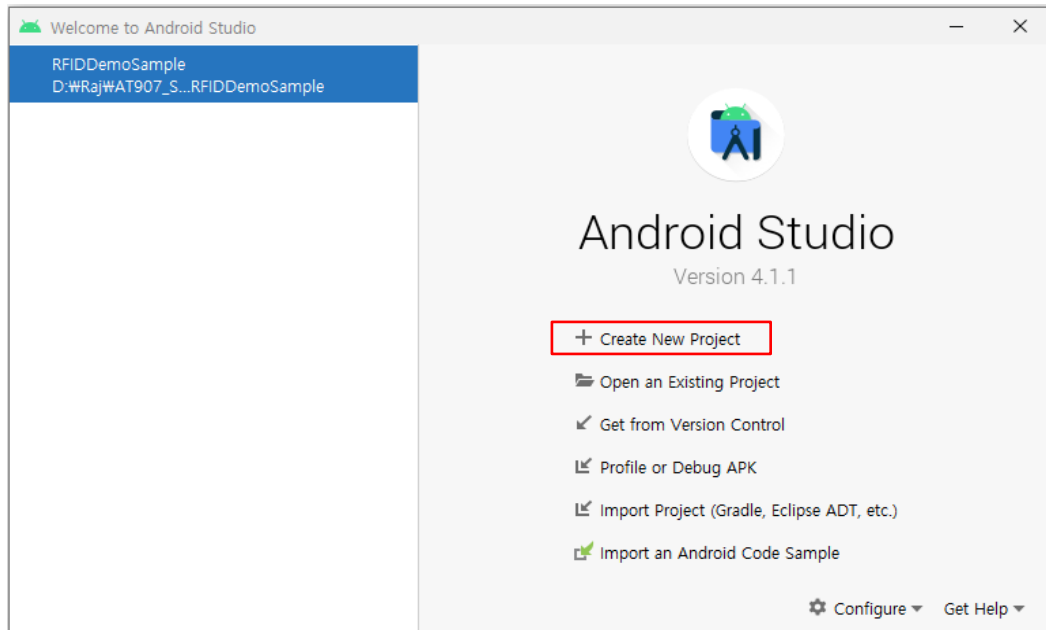
본 문서는 RFID SDK Library를 사용하여 응용 프로그램을 개발하고자 하는 Android개발자들을 위하여 SDK Library를 사용할 수 있는 개발환경 구축과 SDK Library 사용 방법을 기술 하는데 그 목적이 있다.

본 문서에서 사용되는 개발 도구는 Android Studio가 사용되었고 개발 대상 플랫폼은 Android 10을 지원한다.

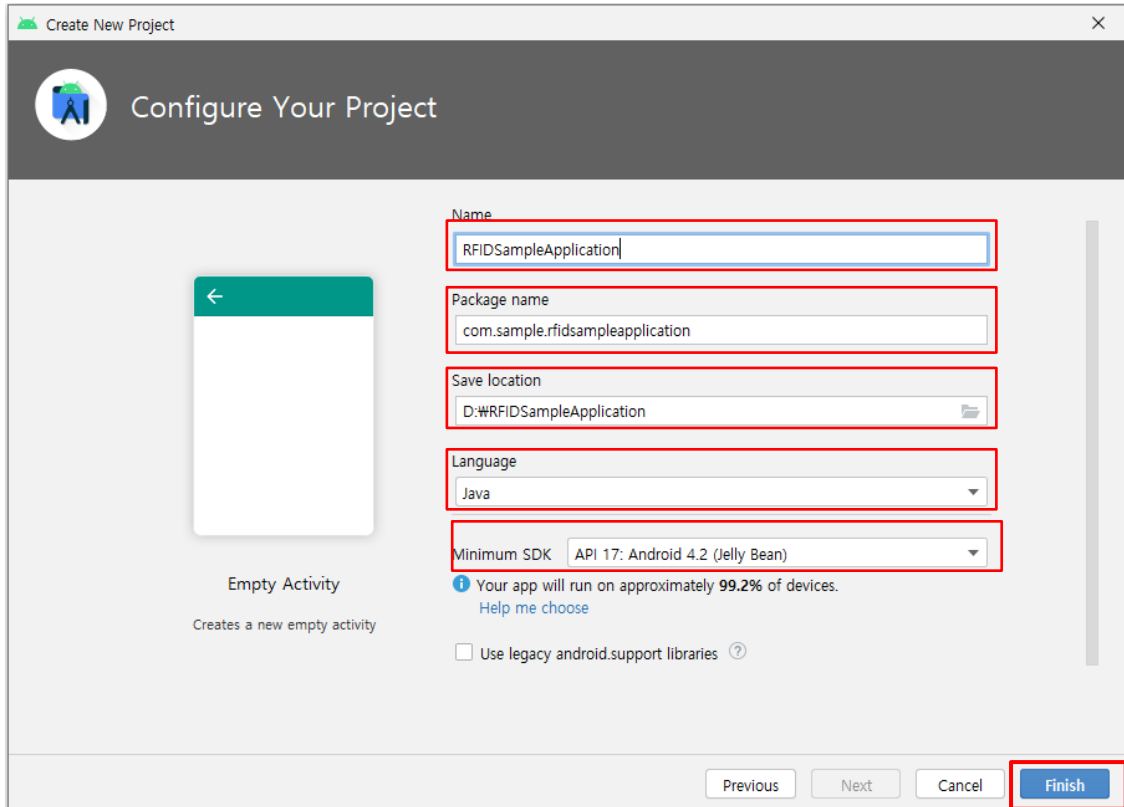
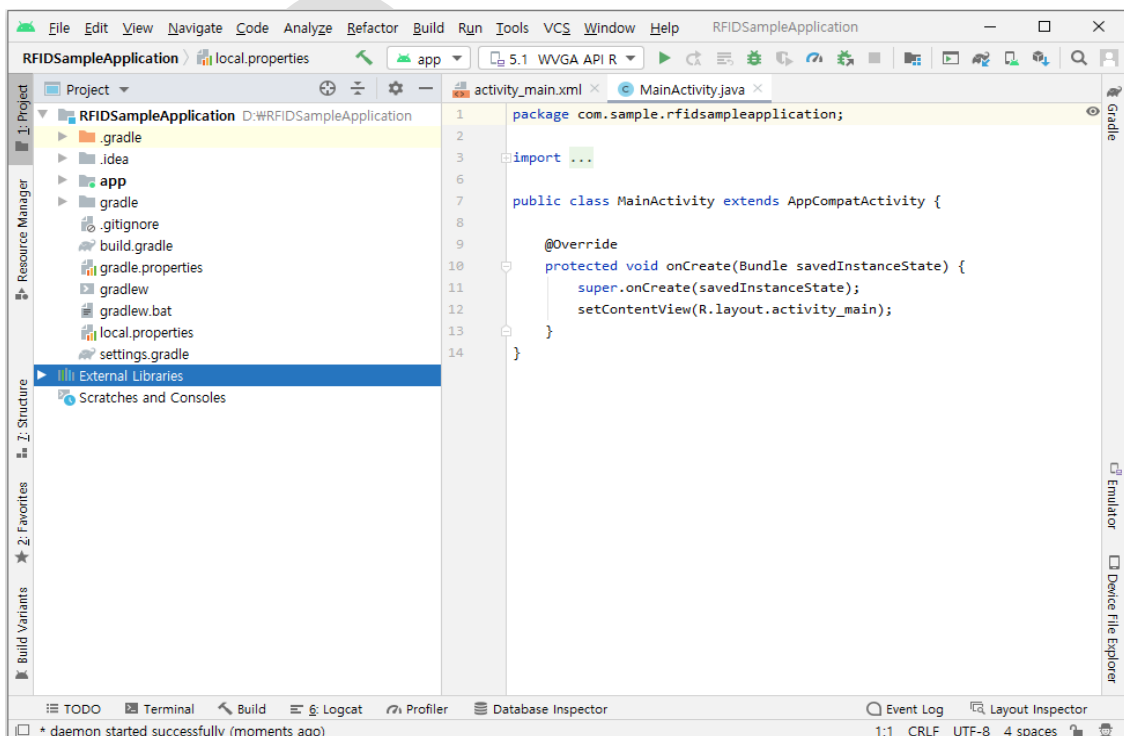
2. Development environment setup

2.1. Create a project.

Android용 응용프로그램을 개발 하기 위하여 Android Studio를 실행한다.



대화상자가 나타나면 Application Name과 Project Name을 입력하고 Package Name을 입력하고, Minimum Required SDK를 선택하고 “Finish” 버튼을 클릭하여 프로젝트 생성을 진행한다

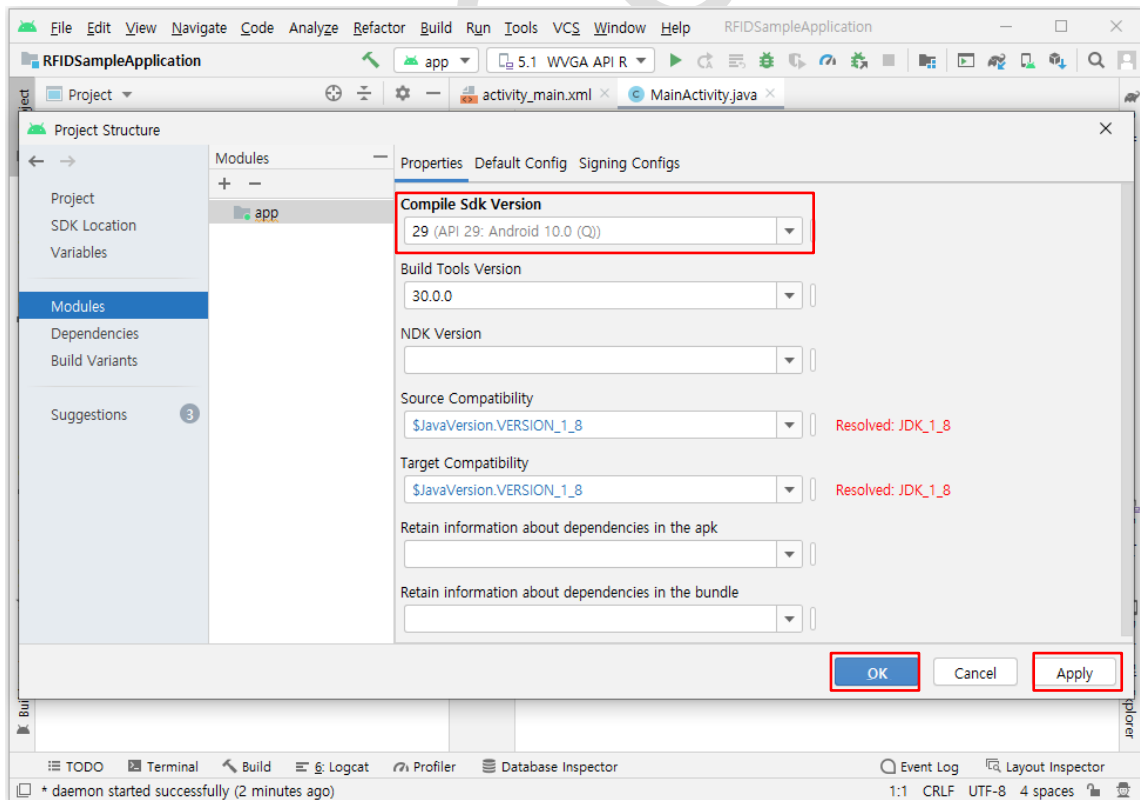
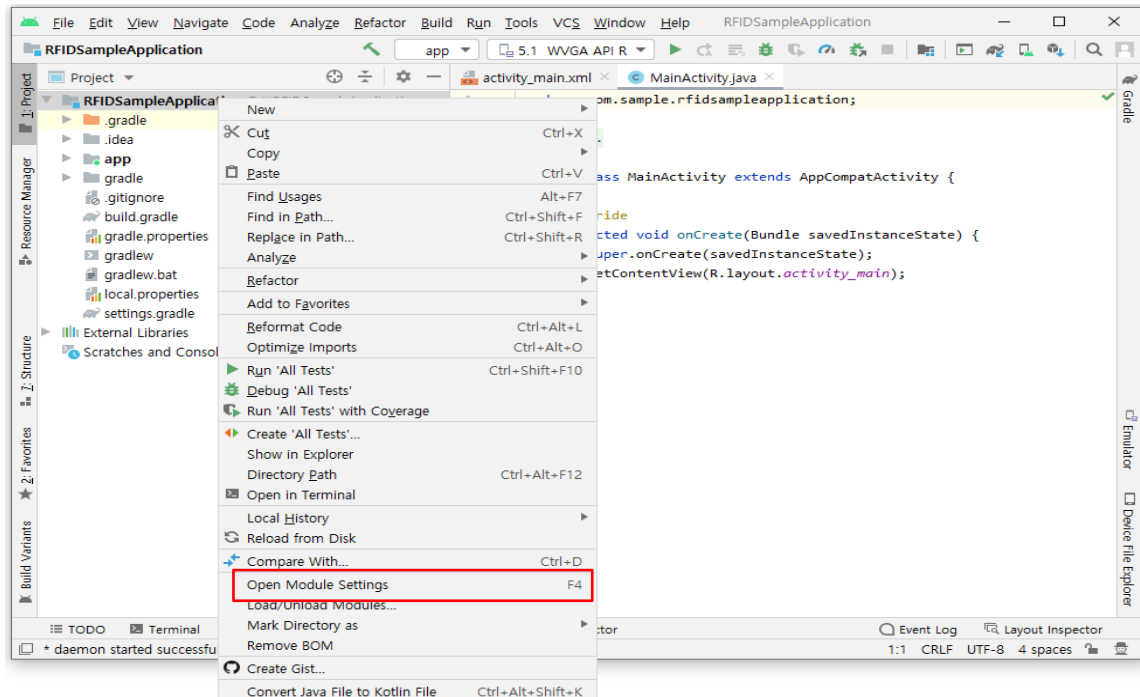



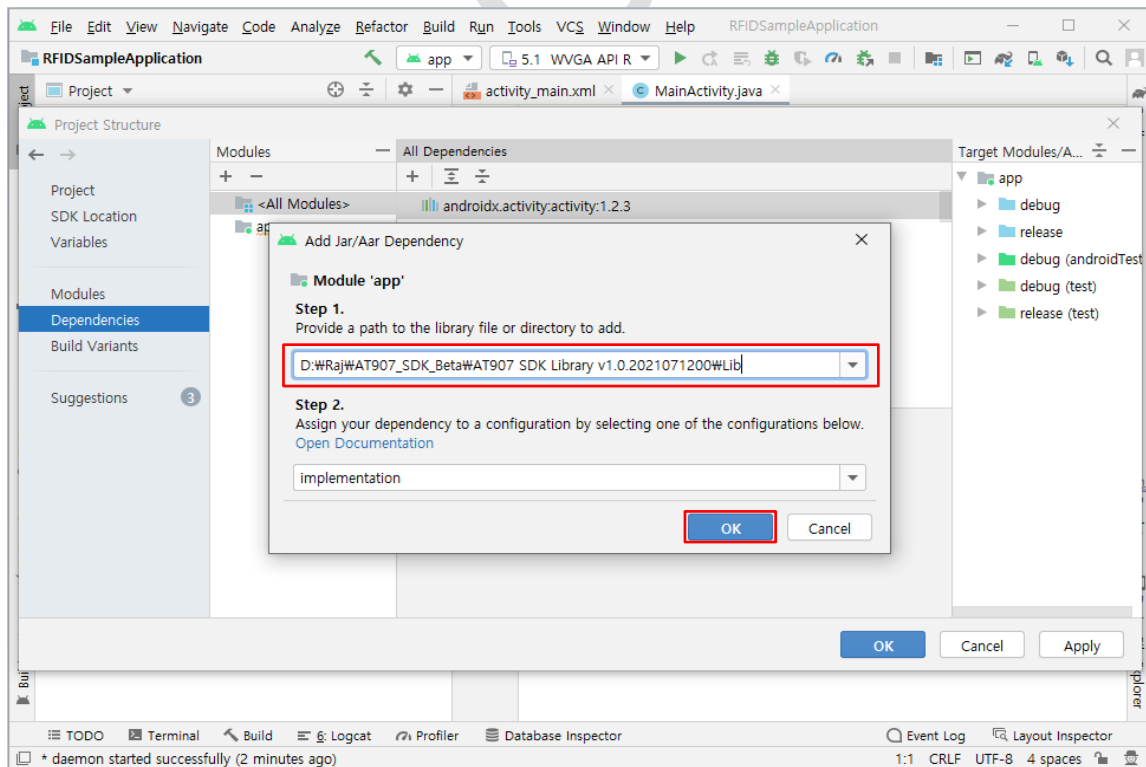
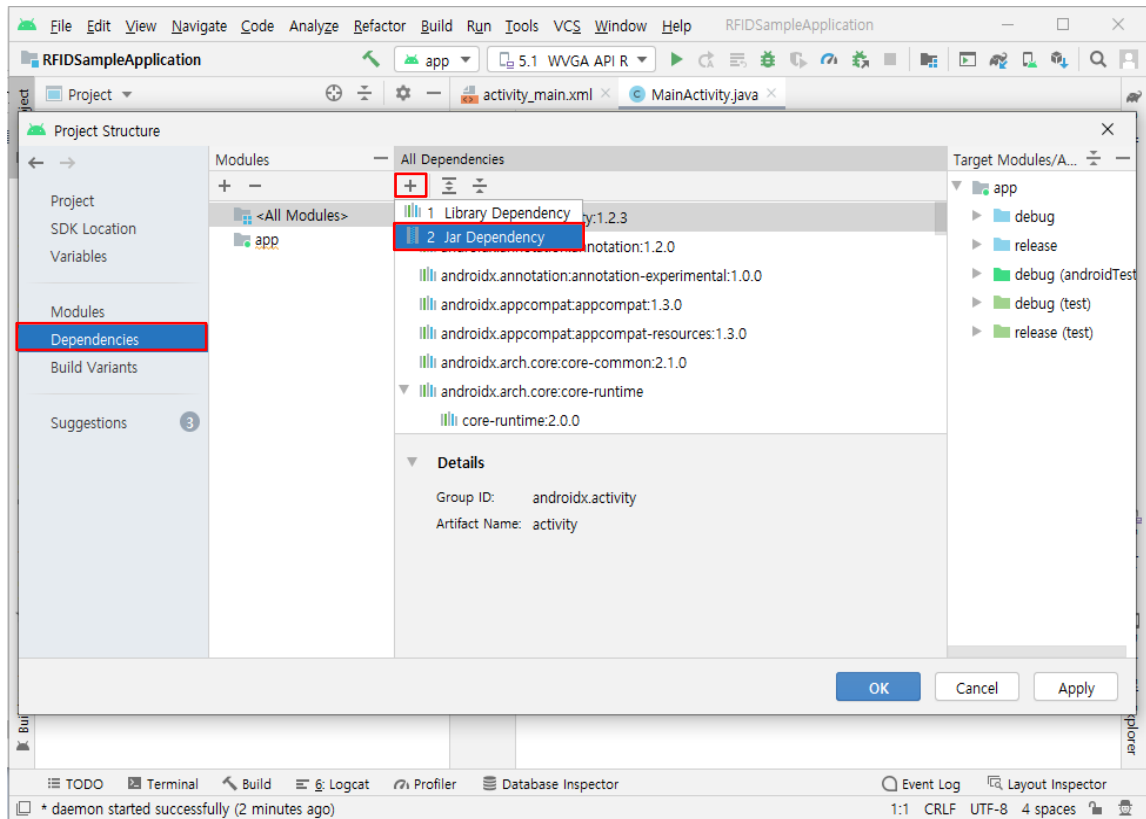
```

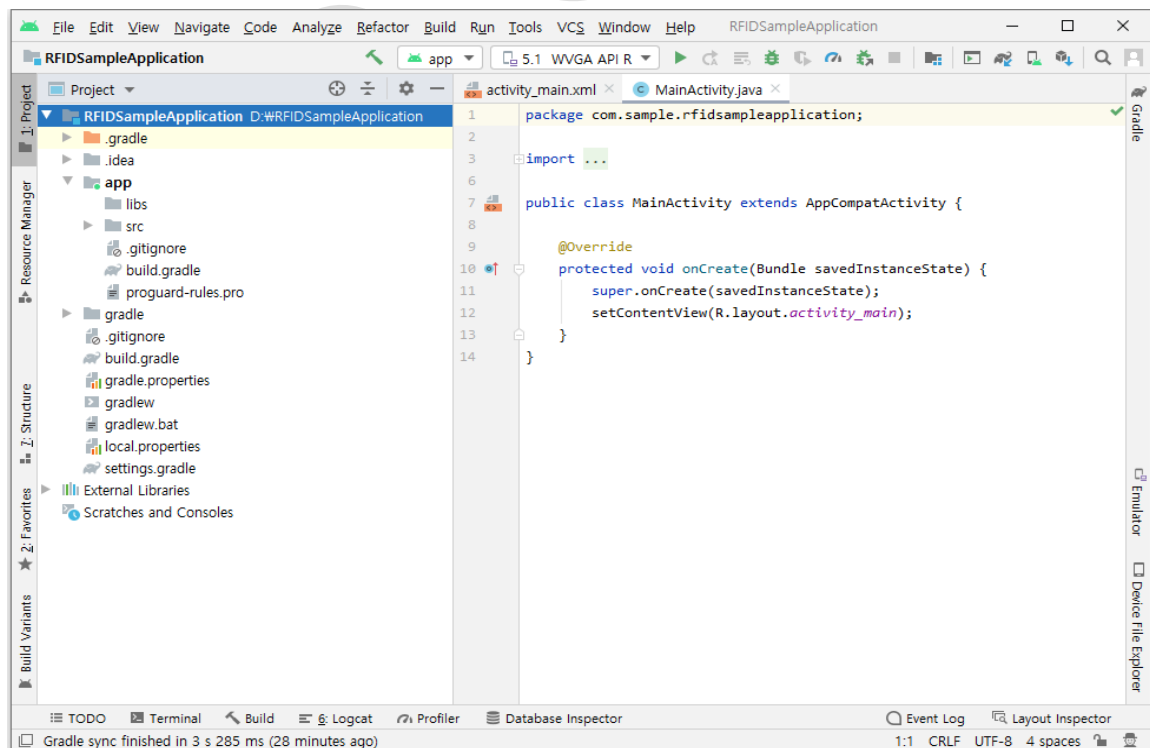
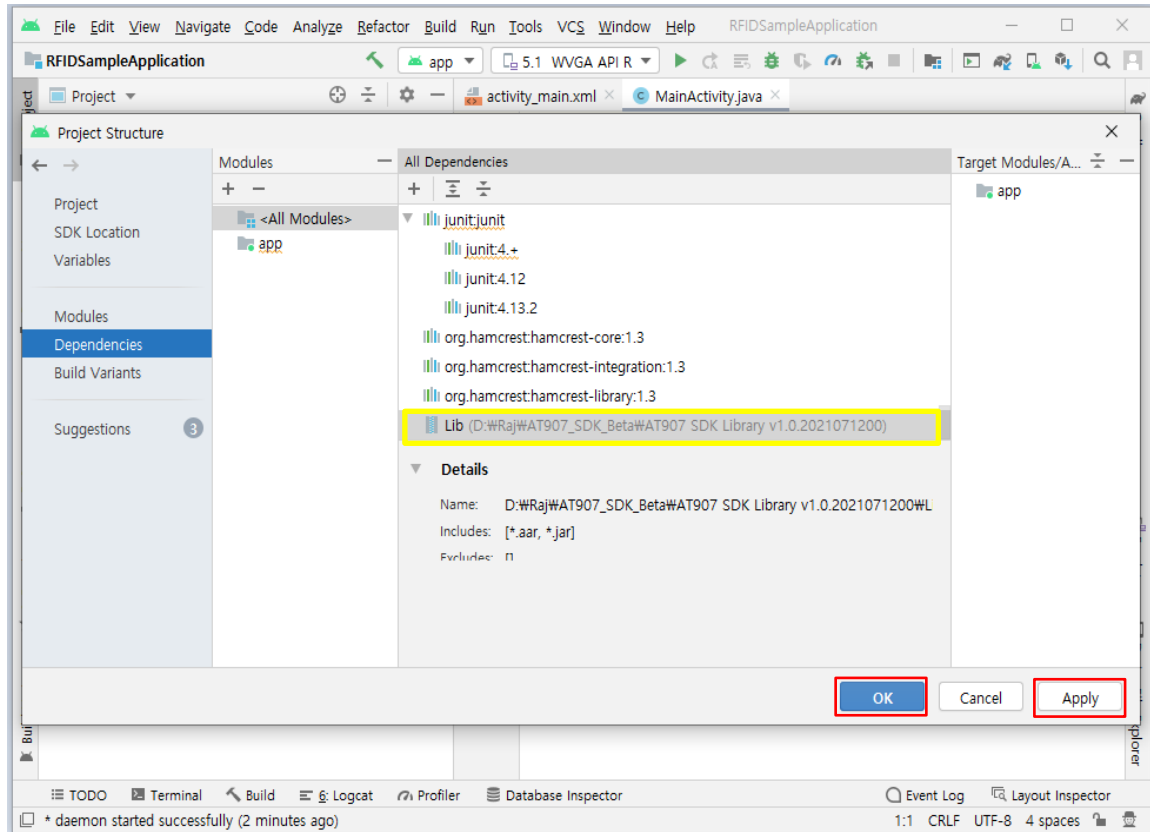
1 package com.sample.rfidsampleapplication;
2
3 import ...
4
5
6
7 public class MainActivity extends AppCompatActivity {
8
9     @Override
10    protected void onCreate(Bundle savedInstanceState) {
11        super.onCreate(savedInstanceState);
12        setContentView(R.layout.activity_main);
13    }
14 }

```

2.1.1 SDK setup and add RFID dependency libraries (ex: jar/aar files)







3. Programing Guide

3.1. 초기화

3.1.1 객체 생성

RFID SDK Library를 사용하여 RFID를 사용하기 위해서는 ATRfidReader 클래스의 인스턴스를 생성하여야 한다. ATRfidReader 클래스의 인스턴스를 생성하는 방법은 Sample 소스의 MainActivity.java파일에 onCreate메서드에서 확인할 수 있다

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    // TODO

    if ((mReader = ATRfidManager.getInstance()) == null) {
        // ERROR
    }
    // TODO
}
```

3.1.2 Event Listener 등록/해제.

ATRfidReader 인스턴스로부터 응답을 수신하기 위해서는, 응답을 수신할 사용자 Activity에 RfidReaderEventListener interface를 구현해야 하며, interface가 구현된 Activity를 setEventListener를 통해서 listener를 등록하고 removeEventListener를 통해서 등록을 해제 한다.

Activity가 여러 개인 App에서는 이 과정을 여러 번 거쳐야 하므로, Activity의 onResume, onPause 메서드에서 등록/해제를 하도록 한다.

```
@Override
protected void onResume() {
    super.onResume();

    if (mReader != null)
        mReader.setEventListener(this);

    ATLog.d(TAG, "INFO onResume()");
}

@Override
protected void onPause() {
    if (mReader != null)
        mReader.removeEventListener(this);

    ATLog.i(TAG, "INFO. onPause()");
    super.onPause();
}
```

3.2. Module power 관리.

Activity가 사용 중일 때와, 그렇지 않을 때를 구분하여 RFID Module의 전원 및 리소스를 관리 한다. Activity의 onStart에서 ATRfidManager.wakeUp()을 Activity의 onStop에서 ATRfidManager.sleep()을 호출한다. wakeUp과 sleep은 반드시 pair로 호출이 되어야 하며, sleep이 호출된 뒤에 wakeUp이 호출되지 않는다면 Module은 정상 동작 하지 않을 수 있다.

```
@Override
protected void onStart() {
    super.onStart();

    if (mReader != null) {
        ATRfidManager.wakeUp();
    }

    ATLog.i(TAG, "INFO. onStart()");
}
@Override
protected void onStop() {
    ATRfidManager.sleep();

    ATLog.i(TAG, "INFO. onStop()");

    super.onStop();
}
```

3.3. Event Handler

EventListener를 등록했다면, event가 발생할 때 마다 사용자가 구현한 interface 함수가 실행된다. 이벤트의 범주는

Module Operation (onReaderStateChanged), Inventory Operation (onReaderReadTag), Access Operation (onReaderResult) 으로 구분된다.

onReaderStateChange 메서드는 Module의 connection 상태에 대하여 이벤트를 수신하기 위한 메서드로 아래와 같이 구현되어 있으며 자세한 내역은 샘플 프로그램을 참조한다.

```
@Override
public void onReaderStateChanged(ATRfidReader reader, ConnectionState
state) {

    switch (state) {
        case Connected:
            // to do something
            break;
        case Disconnected:
            // to do something
            break;
        case Connecting:
            // to do something
            break;
        default:
            break;
    }

    ATLog.i(TAG, "EVENT. onReaderStateChanged(%s)", state);
}
```

onReaderReadTag 메서드는 Inventory operation으로 Tag를 인식 하였을 때 이벤트가 동작하며 아래와 같이 구현되어 있으며, 태그 데이터, RSSI, Phase 정보들을 전달 한다.

```
@Override
public void onReaderReadTag(ATRfidReader reader, String tag, float rssi,
float phase) {
    ATLog.i(TAG, "EVENT. onReaderReadTag([%s], %.2f, %.2f)", tag, rssi,
phase);
}
```

onReaderResult 메서드는 Read/Write/Lock/Kill등의 Access operation에 대한 결과를 확인 하기 위한 Event입니다.

```
@Override
public void onReaderResult(ATRfidReader reader, ResultCode code,
ActionState action, String epc, String data, float rssi, float phase) {
    ATLog.i(TAG, "EVENT. onReaderResult(%s, %s, [%s],
[%s], %.2f, %.2f", code, action, epc, data, rssi, phase);
}
```

3.4. Start and Stop Inventory

3.4.1 Start Inventory

Tag를 Inventory하기 위해서는 inventory6cTag, inventory6bTag, readEpc6cTag, readEpc6bTag 메서드를 사용한다. Inventory메서드를 사용하는 방법은 InventoryActivity.java파일에 구현되어 있다.

```
// Start Action
protected void startAction() {
    // 생략
    if(mReader.getModuleType() == RfidModuleType.I900MA) {
        mMAREader = (ATRfid900MAREader)mReader;
        if (chkContinuousMode.isChecked()) {

            // Multiple Reading
            if(tagType == TagType.Tag6B) {
                mMAREader.inventory6bTag()
            } else if(tagType == TagType.Tag6C) {
                mMAREader.inventory6cTag()
            } else if(tagType == TagType.TagRail) {
                mMAREader.inventoryRailTag()
            } else if(tagType == TagType.TagAny) {
                mMAREader.inventoryAnyTag()
            }
        } else {
            // Single Reading
            if(tagType == TagType.Tag6B) {
                mMAREader.readEpc6bTag()
            } else if(tagType == TagType.Tag6C) {
                mMAREader.readEpc6cTag()
            } else if(tagType == TagType.TagRail) {
                mMAREader.readEpcRailTag()
            } else if(tagType == TagType.TagAny) {
                mMAREader.readEpcAnyTag()
            }
        }
    } else {
        // Multiple Reading
        if (chkContinuousMode.isChecked()) {
            mReader.inventory6cTag()
        } else {
            // Single Reading
            mReader.readEpc6cTag()
        }
    }

    ATLog.i(TAG, "INFO. startAction()");
}
```

3.4.2 Stop Inventory

진행중인 Inventory 또는 Access 명령을 중단 시키기 위해서는 stop을 호출한다.
ActionActivity.java 파일을 참조한다.

```
protected void stopAction() {

    if(mReader.getAction() == ActionState.Stop) {
        ATLog.e(TAG, "ActionState is not busy.");
        return;
    }
    ResultCode res;
    enableWidgets(false);

    if ((res = mReader.stop()) != ResultCode.NoError) {
        ATLog.e(TAG, "ERROR. stopAction() - Failed to stop operation [%s]", res);
        enableWidgets(true);
        return;
    }

    ATLog.i(TAG, "INFO. stopAction()");
}
```

3.5. 종료

RFID의 사용이 끝나서 완전히 종료를 해야 한다면, ATRfidManager.onDestroy()를 호출하여 RFID와 관련된 자원을 해제 한다.

다음은 MainActivity.java의 onDestroy() 함수 구현 부 이다.

```
@Override
protected void onDestroy() {

    // Deinitialize RFID reader Instance
    ATRfidManager.onDestroy();

    // Wake Unlock
    SysUtil.wakeUnlock();

    saveConfig();

    ATLog.d(TAG, "INFO. onDestroy");
    ATLog.shutdown();

    super.onDestroy();
}
```