



ATID Co.,Ltd

AT288N Program Guide for Android Developers

AT288Reader API Reference Guide

SW 팀

2023-06-12

개정 이력

버전	개정일자	개정사유 ¹	개정내역 ²	작성자
v0.1	2016-05-25	초안		민경진
v0.2	2019-01-21	수정	스크린샷 및 설명을 Android Studio로 변경.	조영진
v0.3	2023-06-12	수정	개발 툴 버전 및 BT 권한 추가	SW 팀

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

² 개정내역 : 개정이 발생하는 페이지 번호와 변경 내용을 기술



		AT288N Program Guide for Android Developers					
AT288Reader API Reference Guide					회사	ATID Co.,Ltd	
문서이름		작성자	SW 팀	날자	2023-06-12	버전	v0.3

Table of Contents

Table of Contents	3
1. Intro	4
2. Getting Started	5
2.1. Create Project for Android.....	5
3. Programing Guide.....	11
3.1. Add Permission	11
3.2. Create and Synchronization	12
3.2.1. Create Reader.....	12
3.2.2. Synchronize Reader.....	12
3.3. Connect and Activate.....	13
3.4. Event Handler.....	16
3.5. Action Operation.....	22
3.5.1. Inventory and StopOperation.....	22
3.5.2. Read Memory and Write Memory.....	24
3.5.3. Lock.....	27


	AT288N Program Guide for Android Developers						
AT288Reader API Reference Guide					회사	ATID Co.,Ltd	
문서이름		작성자	SW 팀	날자	2023-06-12	버전	v0.3

1. Intro

본 문서는 AT288N 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 Studio Chipmunk 이며, minSdkVersion은 18 이다.

※ 버전 v1.10.2019012200 부터는 Eclipse 대신 Android Studio를 사용한다.

		AT288N Program Guide for Android Developers					
AT288Reader API Reference Guide					회사	ATID Co.,Ltd	
문서이름		작성자	SW 팀	날자	2023-06-12	버전	v0.3

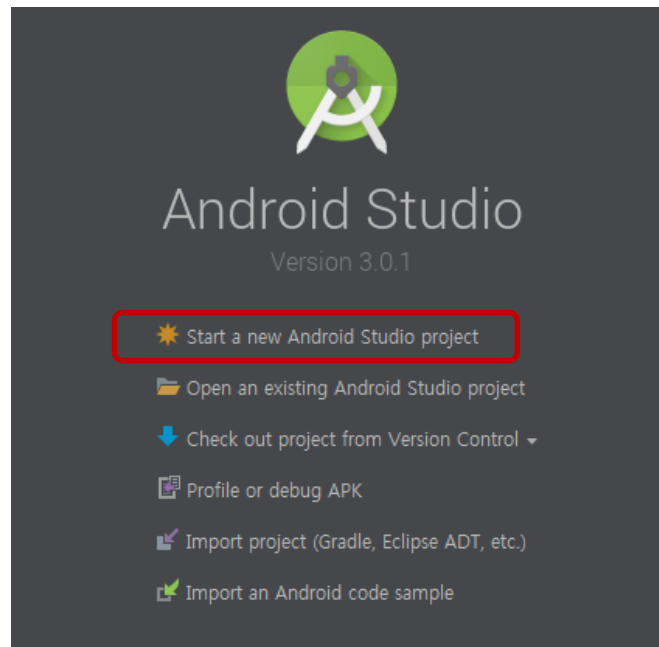
2. Getting Started

Getting Started에서는 AT288N SDK Library를 사용하여 간단하게 Android Sample을 작성하여 AT288N SDK Library 응용 프로그램을 개발하고, AT288N SDK Library를 사용하기 위하여 개발 환경을 구축하는 방법에 대하여 설명한다.

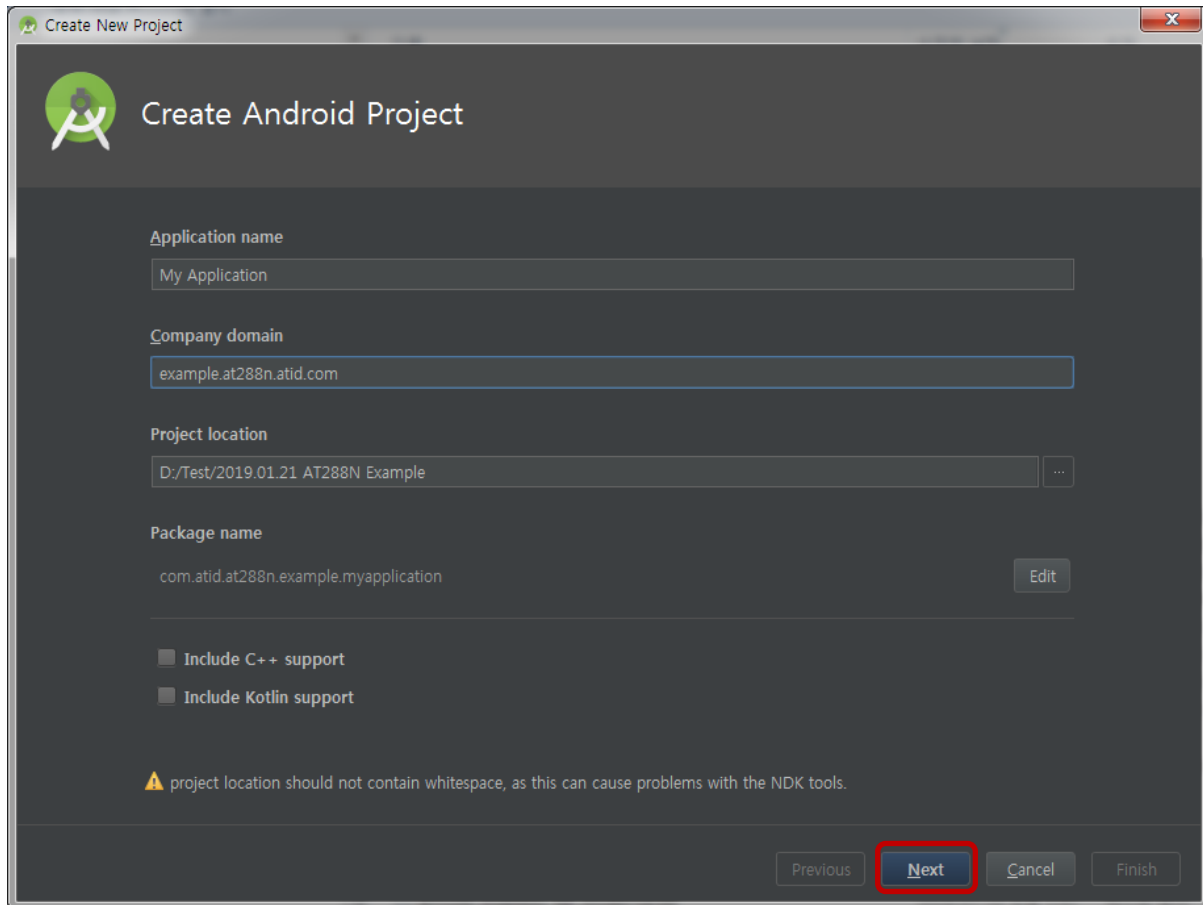
설명에 사용되는 개발 툴은 Android Studio 3.0.1이다.

2.1. Create Project for Android

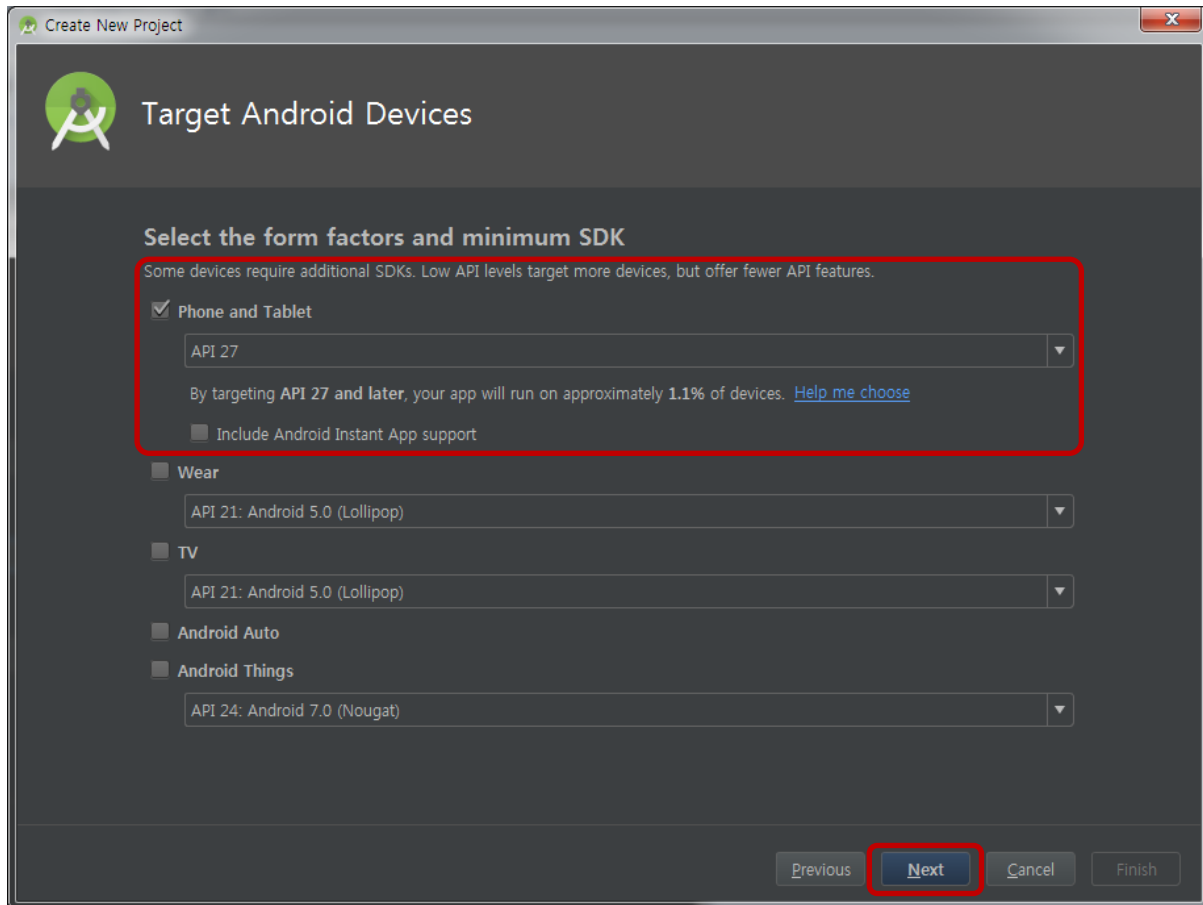
AT288N Android용 응용 프로그램을 개발 하기 위하여 Android Studio를 실행한다.



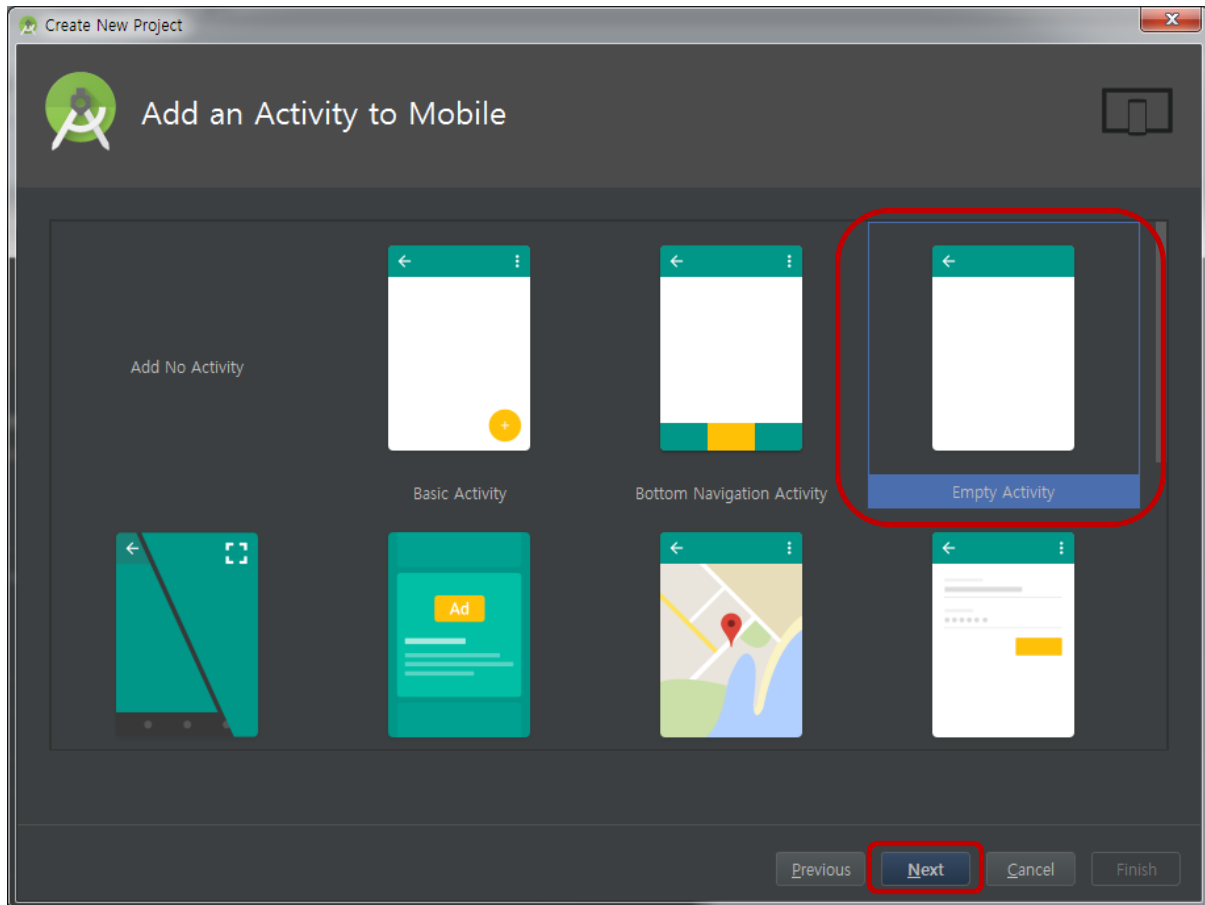
“Start a new Android Studio Project”를 선택하여 새로운 프로젝트를 생성한다.



"Create New Project" 창이 나타나면, Application 이름과 저장될 위치를 선택하고 "Next" 버튼을 누른다.



테스트 할 환경에 맞는 설정을 하고 "Next" 버튼을 누른다.



테스트를 위해 "Empty Activity"를 선택하고 "Next" 버튼을 누른다.


```

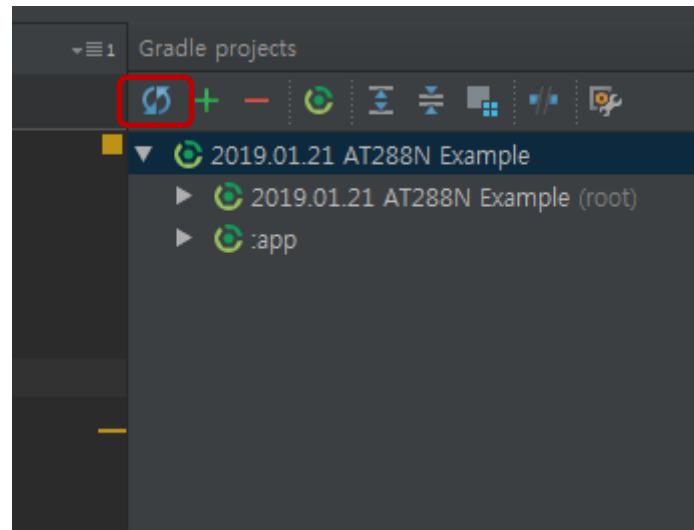
apply plugin: 'com.android.application'

android {
    compileSdkVersion 27
    defaultConfig {
        applicationId "com.atid.at288n.example.myapplication"
        minSdkVersion 21
        targetSdkVersion 26
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support:appcompat-v7:27.1.1'
    implementation 'com.android.support.constraint:constraint-layout:1.1.3'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'com.android.support.test:runner:1.0.2'
    androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'
    compile files('libs/at288lib.aar')
}

```

프로젝트가 만들어졌으면, app의 build.gradle 파일을 열어서 사용할 SDK 버전을 설정하고, AT288N Library를 dependencies에 추가한다. 위의 그림에서는 AT288N Library 파일을 app의 libs 폴더에 복사 해 놓은 상태이다.



build.gradle 파일을 수정한 후, "Sync" 버튼을 눌러서 에러가 없는지 확인한다.
에러가 발생했을 경우, Android Studio의 안내 멘트를 따르도록 한다.
프로젝트를 빌드했을 때 에러가 발생하지 않는다면, 준비가 끝난 것이다.

3. Programing Guide

3.1. Add Permission

AT288N SDK Library를 Android에서 사용하기 위해서는 Bluetooth관련 Permission을 설정하여야 한다. Manifest에 추가 하여야 하는 User-Permission은 다음과 같다.

```
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
```

User-Permission으로 Bluetooth Device을 제어할 수 있는 BLUETOOTH_ADMIN Permission이 필요하고 Bluetooth Device를 사용할 수 있는 BLUETOOTH Permission이 필요하다.

또한 안드로이드 OS 6.0이상의 장비에서 블루투스의 권한을 허용하기 위해서 ACCESS_FINE_LOCATION Permission과 ACCESS_COARSE_LOCATION이 필요하다.

안드로이드 OS 12 및 이상의 장비에서 추가 권한을 필요하다.

```
<uses-permission android:name="android.permission.BLUETOOTH_CONNECT" />
<uses-permission android:name="android.permission.BLUETOOTH_SCAN" />
```

```
void requestBluetoothPermissions(){
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.S) {
        requestPermissions(
            new String[]{
                Manifest.permission.BLUETOOTH,
                Manifest.permission.BLUETOOTH_SCAN,
                Manifest.permission.BLUETOOTH_ADVERTISE,
                Manifest.permission.BLUETOOTH_CONNECT
            },
            requestCode: 1);
    } else if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
        requestPermissions(
            new String[]{
                Manifest.permission.BLUETOOTH
            },
            requestCode: 1);
    }
}
```

onCreate() 메서드에 함수 추가,

```
//Request BT permissions
requestBluetoothPermissions();
```

3.2. Create and Synchronization

3.2.1. Create Reader

AT288N SDK Library를 사용하여 AT288N Device를 사용하기 위해서는 Reader 클래스의 인스턴스를 생성하여야 한다.

Reader클래스의 인스턴스를 생성하는 방법은 Sample 소스의 MainActivity.java파일에 createReader메서드에서 확인할 수 있다.

```
private void createReader() {
    reader = new Reader(this, this);
    reader.mDeviceAddress = deviceAddress;
}
```

createReader메서드는 Main Activity의 오버라이드된 메서드인 onCreate메서드에서 호출되는데 Main Activity가 생성되면서 onCreate이벤트가 호출되면 Reader클래스의 인스턴스를 생성하는 것이 가장 적당하다. Reader클래스의 인스턴스를 생성할 때는 Context와 이벤트를 전달받기 위해 IReaderCallbackReceiver를 구현한 인스턴스를 파라미터로 전달한다.

3.2.2. Synchronize Reader

Main Activity와 Reader개체의 Life Cycle를 관리하기 위해 onCreate, onStart, onResume, onPause, onStop, onDestroy, onActivityResult등의 오버라이드된 메서드를 재정의하는 경우가 발생하는데 필요에 따라서 Reader개체를 동기화 하기 위해서 각 재정의된 메서드에서 Reader개체의 동일한 메서드를 호출할 수 있다.

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

@Override
protected void onResume() {
    super.onResume();
    loadConfig();
    reader.onResume();
}

@Override
protected void onPause() {
    reader.onPause();
    super.onPause();
}

@Override
protected void onStop() {
    saveConfig();
    reader.onStop();
    super.onStop();
}
```

```
@Override
protected void onDestroy() {
    saveConfig();
    super.onDestroy();
    reader.onDestroy();
}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {

    if(data == null) {
        Log.d(TAG, "DEBUG. data is null.");
        return;
    }

    String address = data.getExtras().getString(DeviceListActivity.EXTRA_DEVICE_ADDRESS);
    if(address != null) {
        reader.onActivityResult(requestCode, resultCode, data);
        if (requestCode == Reader.REQUEST_CONNECT_DEVICE
            && resultCode == Activity.RESULT_OK) {
            showWaitDialog("", getResources()
                .getString(R.string.connect_reader));
        }
    } else {
        Log.e(TAG, "ERROR. Device address is null.");
    }
}
```

주의 할 것은 onActivityResult메서드를 반드시 재정의하여 Reader클래스의 onActivityResult메서드를 호출하여야 OpenDeviceListActivity메서드가 정상 동작한다는 것이다. OpenDeviceListActivity메서드는 내부에서 Intent로 Activity를 호출하며 그 결과값을 MainActivity의 onActivityResult를 통하여 받는데, AT288N SDK Library 내부에서 이를 처리하기 위해서 Reader 객체의 onActivityResult를 호출해 주어야 하는 것이다.

3.3. Connect and Activate

AT288N SDK Library에서 AT288N Device연결하는 방법을 크게 2가지로 제공한다. 주변의 AT288N Device를 Scan하여 연결하는 방법과 마지막 연결하였던 AT288N Device에 연결하는 방법이 있다.

MainActivity.java의 onOptionsItemSelected메서드에서 Reader 객체를 사용하여 AT288N Device에 접속하는 방법을 보여주고 있다.

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.device_connect:
            connectReader();
            break;
        case R.id.device_new_connect:
            reader.openDeviceListActivity();
    }
}
```

```

        break;
    case android.R.id.home:
        if (isShowSubView()) {
            closeSubView();
            return true;
        }
        break;
    case R.id.menu_disconnect:
        // Disconnect Device
        reader.stop();
        return true;
    }
    return super.onOptionsItemSelected(item);
}

```

그런데 최근에 접속하였던 AT288N Device에 접속하는 방법은 Host의 주변에 최근 접속했던 AT288N Device가 없는 경우가 존재하므로 Host의 주변에 AT288N Device가 존재하지 않거나 이전에 접속했던 AT288N Device가 없을 경우에는 OpenDeviceListActivity메서드를 호출할 수 있도록 connectReader메서드에서 구현되어있다.

```

private void connectReader() {
    reader.connectMostRecentDevice();
    if (reader.mDeviceAddress != null) {
        showWaitDialog("", getResources()
            .getString(R.string.connect_reader));
    }
}

```

AT288N Device와 연결이 완료되면 이벤트 핸들러로 MESSAGE_STATE_CHANGE이벤트가 STATE_CONNECTED상태값으로 발생하는데 이때, AT288N Device Reader개체를 동기화 하기 위해 activate 메서드를 호출해주어야 한다.

```

@Override
public void onReaderStateChange(int state) {
    switch (state) {
        ...
        case Reader.STATE_CONNECTED:
            // Case Connected Reader

            // Save Reader Bluetooth Mac Address
            txtBluetoothAddress.setText(reader.mDeviceAddress);
            deviceAddress = reader.mDeviceAddress;
            reader.setResponseTime(responseTimeout);
            saveConfig();

            // Reader Activate...
            reader.activate();

            //Reader Firmware Version
            reader.getFirmwareVersion();

            reader.getVersionEx();
            hideWaitDialog();
            visibleMenuItem(state);
    }
}

```



AT288N Program Guide for Android Developers

AT288Reader API Reference Guide

회사

ATID Co.,Ltd

문서이름

작성자

SW 팀

날자

2023-06-12

버전

v0.3

```
enableAllButtons(true);
txtLogo.setTextColor(getResources().getColor(
    R.color.connected_device));
break;
...
}

// Transfer Event to Subview...
if (isShowSubView())
    currentView.onReaderStateChange(state);
}
```

3.4. Event Handler

AT288N Device로부터 이벤트를 수신하기 위해서는, Reader 인스턴스 생성시에 IReaderCallbackReceiver를 구현한 개체를 파라미터로 입력해야 한다.

Reader에서는 상황에 맞는 IReaerCallbackReceiver 인터페이스 멤버를 실행 시킨다.

따라서, IReaderCallbackReceiver 인터페이스 멤버 구현부에서는 그에 해당하는 적당한 처리를 해주어야 한다.

MainActivity.java에 구현된 onReaderStateChange.

Device의 연결 상태가 변경되었을 때 실행된다.

```
@Override
public void onReaderStateChange(int state) {
    switch (state) {
        case Reader.STATE_CONNECTING:
            // Case Connecting Reader or Ready Connecting Reader...
            enableAllButtons(false);
            txtLogo.setTextColor(getResources().getColor(
                R.color.connecting_device));
            break;
        case Reader.STATE_CONNECTED:
            // Case Connected Reader

            // Save Reader Bluetooth Mac Address
            txtBluetoothAddress.setText(reader.mDeviceAddress);
            deviceAddress = reader.mDeviceAddress;
            reader.setResponseTime(responseTimeout);
            saveConfig();

            // Reader Activate...
            reader.activate();

            //Reader Firmware Version
            reader.getFirmwareVersion();

            reader.getVersionEx();
            hideWaitDialog();
            visibleMenuItem(state);
            enableAllButtons(true);
            txtLogo.setTextColor(getResources().getColor(
                R.color.connected_device));
            break;
        case Reader.STATE_LISTEN:
        default:
            // Case Disconnected Reader...
            hideWaitDialog();

            views[INVENTORY_VIEW].clearMaskParams();
            views[TID_VIEW].clearMaskParams();
            views[MEMORY_VIEW].clearMaskParams();
            views[ACCESS_VIEW].clearMaskParams();
    }
}
```



```

        isBusy = false;
        enableAllButtons(false);
        txtLogo.setTextColor(getResources().getColor(
            R.color.disconnected_device));
        txtReaderVersion.setText("");
        txtFirmwareVersion.setText("");
        txtBluetoothAddress.setText("");
        break;
    }

    // Transfer Event to Subview...
    if (isShowSubView())
        currentView.onReaderStateChange(state);
}

```

MainActivity.java에 구현된 onReaderTimeout.
Device에서 Timeout이 발생했을 때 실행된다.

```

@Override
public void onReaderTimeout() {
    Log.d(TAG, "### TIMEOUT");
    // Transfer Event to Subview...
    if (isShowSubView())
        currentView.onReaderTimeout();
}

```

MainActivity.java에 구현된 onReaderReadTag.
Inventory 명령으로 태그가 읽혔을 때 실행된다.

```

@Override
public void onReaderReadTag(int event, String tag) {
    Log.d(TAG, "### READ TAG [" + event + "] : " + tag);
    // Transfer Event to Subview...
    if (isShowSubView())
        currentView.onReaderReadTag(event, tag);
}

```

MainActivity.java에 구현된 onReaderResponse.
Access(Read/Write/Lock/Kill) 명령이 종료되었을 때 실행된다.

```

@Override
public void onReaderResponse(int event, String code) {
    Log.d(TAG, "### RESPONSE [" + event + "] : " + code);
    // Transfer Event to Subview...
    if (isShowSubView())
        currentView.onReaderResponse(event, code);
}

```

MainActivity.java에 구현된 onReaderActionChange

Device의 Action 상태가 변경되었을 때 실행된다.

```
@Override
public void onReaderActionChange(char action) {
    Log.d(TAG, "### ACTION : " + action);

    switch(action) {
        case Reader.ACTION_INVENTORY_6B_MULTIPLE:
        case Reader.ACTION_INVENTORY_6B_SINGLE:
        case Reader.ACTION_INVENTORY_6C_MULTIPLE:
        case Reader.ACTION_INVENTORY_6C_SELECTION:
        case Reader.ACTION_INVENTORY_6C_SINGLE:
        case Reader.ACTION_INVENTORY_6C_VLC:
            isBusy = true;
            enableAllButtons(false);
            break;
        case Reader.ACTION_STOP:
            isBusy = false;
            enableAllButtons(true);
            break;
    }

    // Transfer Event to Subview...
    if (isShowSubView())
        currentView.onReaderAction(action);
}
```

MainActivity.java에 구현된 onReaderProperty

Device의 속성을 읽었을 때 실행된다.

```
@Override
public void onReaderProperty(char code, String value) {
    //Log.d(TAG, "### PROPERTY [" + code + "] : " + value);

    if (code == Reader.PROPERTY_VERSION) {
        // Output Reader Firmware Version
        txtFirmwareVersion.setText(value);
        for (BaseView view : views)
            view.createView();
    }

    // Transfer Event to Subview...
    if (isShowSubView())
        currentView.onReaderProperty(code, value);
}
```

MainActivity.java에 구현된 onReaderExtendedProperty

Device의 확장 속성을 읽었을 때 실행된다.

```
@Override
public void onReaderExtendedProperty(char code, String value) {
    //Log.d(TAG, "### EXTENDED PROPERTY [" + code + "] : " + value);
}
```

```

if(code == Reader.PROPERTY_EX_VERSION)
    txtReaderVersion.setText(value);

if(code == Reader.PROPERTY_EX_VERSION || code == Reader.PROPERTY_EX_TAG_TYPE) {
    if(reader.IsAT288N_MA()) {
        btnMenus[5].setEnabled(false); // Inventory with Memory is not supported when module is
AT288N MA.
        if(reader.TagType == Reader.ISO18000_6B) {
            btnMenus[2].setEnabled(false); // Read/Write are not supported when module is AT288N MA
and 6B mode.
            btnMenus[3].setEnabled(false); // Lock/Kill are not supported when module is AT288N MA
and 6B mode.
        } else {
            btnMenus[2].setEnabled(true);
            btnMenus[3].setEnabled(true);
        }
    } else {
        btnMenus[2].setEnabled(true);
        btnMenus[3].setEnabled(true);
        btnMenus[5].setEnabled(true);
    }
}

// Transfer Event to Subview...
if (isShowSubView())
{
    currentView.onReaderExtentedProperty(code, value);
}
}

```

속성과 확장 속성에 대한 더 많은 예제를 보고 싶다면 OptionView.java의 onReaderProperty메서드와 onReaderExtentedProperty메서드를 참조하도록 한다.

```

@Override
public void onReaderProperty(char code, String value) {
    Log.d(TAG, String.format("onReaderProperty(%c, %s)", code, value));
    switch (code) {
        case Reader.PROPERTY_ANTENNA_SWITCHING_TIME:
            setInventoryTime(Integer.parseInt(value));
            checkInit(InitChecker.INVENTORY_TIME);
            break;
        case Reader.PROPERTY_POWER_IDLE_TIME:
            setIdleTime(Integer.parseInt(value));
            checkInit(InitChecker.IDLE_TIME);
            break;
        case Reader.PROPERTY_GLOBAL_BAND:
            int globalBand = Integer.parseInt(value);
            g_GlobalBand_Now = globalBand;

            if(reader.IsAT288N_MA()) {
                this.spinGlobalBand.setAdapter(null);
                this.adapterGlobalBand = ArrayAdapter.createFromResource(getContext(),
                    R.array.global_band_ma_array, android.R.layout.simple_spinner_dropdown_item);
            }
    }
}

```

```

        this.spinGlobalBand.setAdapter(this.adapterGlobalBand);
        txtGlobalBand.setText(getResources().getStringArray(
            R.array.global_band_ma_array)[globalBand]);
    } else {
        txtGlobalBand.setText(getResources().getStringArray(
            R.array.global_band_array)[globalBand]);
    }

    spinGlobalBand.setSelection(globalBand);
    checkInit(InitChecker.GLOBAL_BAND);

    createView(g_GlobalBand_Now);
    break;
case Reader.PROPERTY_LBT_CHANNEL:
    showLBTChannelDialog(Integer.parseInt(value));
    break;

case Reader.PROPERTY_AUTO_POWEROFF: //AT288N Only
    setAutoPowerOff(Integer.parseInt(value));
    break;
case Reader.PROPERTY_BUZZER:
    spinBeepMode.setSelection(Integer.parseInt(value));
    break;
}
}

```

OptionView.java의 onReaderExtendedProperty메서드에서는 Power Gain, Tag Type, Connect Type, Read Type등의 확장 속성에 대한 예제를 볼 수 있다.

```

@Override
public void onReaderExtendedProperty(char code, String value) {
    Log.d(TAG,
        String.format("onReaderExtendedProperty(%c, %s)", code, value));
    switch (code) {
        case Reader.PROPERTY_EX_TAG_TYPE:
            spinTagType.setSelection(Integer.parseInt(value));
            checkInit(InitChecker.TAG_TYPE);
            if(reader.isAT288N_MA())
                spinTagType.setEnabled(true); //AT288 Only 6C/6B Select , AT288N is Only 6C TagType
            else
                spinTagType.setEnabled(false);

            if(reader.TagType == Reader.ISO18000_6B && InventoryFormat.getCount() > 2) {
                this.InventoryFormat.setAdapter(null);
                this.adapterInventoryFormat = ArrayAdapter.createFromResource(getContext(),
                    R.array.inventory_format_array_ma,
                    android.R.layout.simple_spinner_dropdown_item);
                this.InventoryFormat.setAdapter(this.adapterInventoryFormat);
            } else if(reader.TagType == Reader.ISO18000_6C_GEN2 && InventoryFormat.getCount() < 4) {
                this.InventoryFormat.setAdapter(null);
                this.adapterInventoryFormat = ArrayAdapter.createFromResource(getContext(),
                    R.array.inventory_format_array,
                    android.R.layout.simple_spinner_dropdown_item);
            }
    }
}

```

```

        this.InventoryFormat.setAdapter(this.adapterInventoryFormat);
    }

    break;
case Reader.PROPERTY_EX_CONTINUE_MODE:
    spinReadType.setSelection(Integer.parseInt(value));
    checkInit(InitChecker.READ_TYPE);
    break;
case Reader.PROPERTY_EX_POWER_GAIN:
    spinPowerGain.setSelection(Integer.parseInt(value));
    checkInit(InitChecker.INVENTORY_TIME);
    checkInit(InitChecker.IDLE_TIME);
    checkInit(InitChecker.POWER);
    break;
case Reader.PROPERTY_EX_USE_SERIAL_NO:
    InventoryFormat.setSelection(Integer.parseInt(value));
    checkInit(InitChecker.USE_SERIAL_NO);
    break;
case Reader.PROPERTY_EX_BATTERY_STATE:
    String strValue;
    strValue = value.equals("0") ? "high" : "Low";
    strValue = "Battery State : " + strValue;
    Toast.makeText(getContext(), strValue, Toast.LENGTH_LONG).show();
    break;
case Reader.PROPERTY_EX_NATIONAL_CODE:
    txtNationalCode.setText(value);
    edtNationalCode.setText(value);
    checkInit(InitChecker.NATIONAL_CODE);
    break;
case Reader.PROPERTY_EX_SERIAL_NO:
    txtSerialNo.setText(value);
    edtSerialNo.setText(value);
    checkInit(InitChecker.SERIAL_NO);
    break;
}
}

```

3.5. Action Operation

3.5.1. Inventory and StopOperation

AT288N Device로 Tag를 Inventory하기 위해서는 Inventory메서드를 사용한다. Inventory메서드를 사용하는 방법은 InventoryView.java파일의 onClick메서드에 구현되어 있다.

```
@Override
public void onClick(View v) {
    Log.d(TAG, String.format("onClick(%d)", v.getId()));

    switch (v.getId()) {
        case R.id.stored_mode:
            reader.setStoredMode(this.chkStoredMode.isChecked() ? StoredModeType.Stored :
            StoredModeType.NotStored);
            break;
        case R.id.read_inventory:
            if (ActionType.Stop == this.action) {
                if (mask.getMask().equals("")) {
                    reader.inventory(null);
                    Log.e(TAG, "inventory without mask");
                } else {
                    reader.inventory(mask.getMask());
                    Log.e(TAG, "inventory with mask");
                }
            } else {
                reader.stopOperation();
            }
            enableWidget(false);
            break;
        case R.id.clear:
            clearTagList();
            break;
        case R.id.mask:
            mask.show();
            break;
    }
}
```

Inventory메서드의 경우 Selection Mask를 사용하지 않는 경우 null값을 파라미터로 전달하면 되고 Selection Mask를 사용하는 경우에는 SetSelectionBank, SetSelectionOffset, SetSelectionAction 등의 메서드를 사용하여야 한다.

Selection Mask를 설정하는 방법은 InventoryView.java파일의 maskListener를 할당하는 부분에 구현되어 있다.

```
DialogInterface.OnClickListener maskListener = new DialogInterface.OnClickListener() {

    @Override
    public void onClick(DialogInterface dialog, int which) {
        reader.setSelectionBank(mask.getBank());
        reader.setSelectionOffset(mask.getOffset());
        reader.setSelectionAction(mask.getAction());
    }
}
```

```
mask.setMask(mask.getMask());
}
};
```

Selection Mask를 사용하기 위해서는 GetSelectionBank, GetSelectionOffset, GetSelectionAction등의 메서드로 현재 Selection Mask값들을 가져온 후 SetSelectionBank, SetSelectionOffset, SetSelectionAction등의 메서드로 Selection Mask값을 설정한 후 Inventory 메서드를 호출할 때, 파라미터로 Mask값을 넣어서 호출하면된다. Inventory 메서드가 호출되면 AT288N Device는 Inventory를 시작하며, ReadTag이벤트로 AT288N Device가 읽은 Tag값이 반환된다. Inventory를 정지하고 싶다면 StopOperation메서드를 호출하면 된다.

3.5.2. Read Memory and Write Memory

Tag의 메모리를 읽어내기 위해서는 ReadMemory메서드를 사용하고, Tag 메모리에 쓰기 위해서는 WriteMemory메서드를 사용한다. ReadMemory메서드와 WriteMemory메서드는 MemoryView.java 파일의 onClick메서드에 구현되어 있다.

```
public void onClick(View v) {
    switch (v.getId()) {
        case R.id.read_memory:
            if (ActionType.Stop == this.action) {
                if (!adapterOption.getPassword().equals("")) {
                    if (adapterOption.getPassword().length() != 8) {
                        Toast.makeText(getContext(), "Password's length is must be 8",
                            Toast.LENGTH_SHORT).show();
                        return;
                    }

                    reader.setAccessPassword(adapterOption.getPassword());
                }
                if (mask.getMask().equals("")) {
                    reader.readMemory(adapterOption.getBank(),
                        adapterOption.getOffset(),
                        adapterOption.getLength());
                } else {
                    reader.readMemory(adapterOption.getBank(),
                        adapterOption.getOffset(),
                        adapterOption.getLength(), mask.getMask());
                }
                displayMessage(
                    getResources().getString(R.string.memory_read_label),
                    true);
            } else {
                reader.stopOperation();
                displayMessage(getResources().getString(R.string.stop_msg),
                    false);
            }
            enableWidget(false);
            break;
        case R.id.write_memory:
            if (ActionType.Stop == this.action) {

                if (edtWriteValue.getText().toString().isEmpty()) {
                    Toast.makeText(getContext(), "WriteValue is not to be null or empty",
                        Toast.LENGTH_SHORT).show();
                    return;
                }

                if (!adapterOption.getPassword().equals("")) {
                    if (adapterOption.getPassword().length() != 8) {
                        Toast.makeText(getContext(), "Password's length is must be 8",
                            Toast.LENGTH_SHORT).show();
                        return;
                    }
                }
            }
    }
}
```



```

        reader.setAccessPassword(adapterOption.getPassword());
    }
    if (mask.getMask().equals("")) {
        reader.writeMemory(adapterOption.getBank(), adapterOption
            .getOffset(), edtWriteValue.getText().toString());
    } else {
        reader.writeMemory(adapterOption.getBank(), adapterOption
            .getOffset(), edtWriteValue.getText().toString(),
            mask.getMask());
    }
    displayMessage(
        getResources().getString(R.string.memory_write_label),
        true);
} else {
    reader.stopOperation();
    displayMessage(getResources().getString(R.string.stop_msg),
        false);
}
enableWidget(false);
break;
case R.id.clear:
    clearTagList();
    break;
case R.id.mask:
    mask.show();
    break;
}
}

```

ReadMemory메서드는 읽을 Tag의 Memory Bank와 읽기 시작할 시작 주소, 메모리를 읽을 길이를 지정하여 호출한다.

ReadMemory메서드도 Inventory메서드와 같이 Selection Mask를 사용할 수 있다.

AT288N Device가 정상적으로 Tag Memory를 읽으면 ReadTag 이벤트가 발생하고, Tag Memory를 읽는 동작을 중지된다. 실패하였다면 Response 이벤트가 발생하고 Tag Memory를 읽는 동작이 중지된다.

AT288N Device가 Tag 메모리를 읽기 전에 Memory 읽기 동작을 취소하고 싶다면 StopOperation메서드를 호출하여 동작을 취소할 수 있다.

WriteMemory메서드는 데이터를 쓸 Tag의 Memory Bank와 쓰기 시작할 시작 주소, Tag Memory에 쓸 데이터를 지정하여 호출한다.

WriteMemory메서드도 ReadMemory와 같이 Selection Mask를 사용할 수 있다.

AT288N Device가 Tag Memory에 데이터를 쓰고 나면 성공과 실패여부가 Response이벤트로 반환되고 동작은 중지된다.

Write Memory역시 Tag 메모리에 쓰기 전에 Memory 쓰기 동작을 취소하고 싶다면 StopOperation메서드를 호출하여 동작을 취소할 수 있다.



AT288N Program Guide for Android Developers

AT288Reader API Reference Guide

회사

ATID Co.,Ltd

문서이름

작성자

SW 팀

날자

2023-06-12

버전

v0.3

3.5.3. Lock

Tag의 메모리를 잠그기 위해서는 Lock메서드를 사용하며, AccessView.java파일의 onClick메서드에 구현되어 있다.

```
public void onClick(View v) {
    switch (v.getId()) {
        case R.id.lock:
            if (ActionType.Stop == this.action) {

                if (edtAccessPassword.getText().toString().length() != 8) {
                    Toast.makeText(getApplicationContext(), "Access PWD's length is must be 8",
                        Toast.LENGTH_SHORT).show();
                    return;
                }

                reader.setAccessPassword(edtAccessPassword.getText().toString());
                if (mask.getMask().equals("")) {
                    reader.lock(adapterOption.getMask(),
                        adapterOption.getAction());
                } else {
                    reader.lock(adapterOption.getMask(),
                        adapterOption.getAction(), mask.getMask());
                }
                showMessage(
                    getResources().getString(R.string.access_lock_label),
                    true);
            } else {
                reader.stopOperation();
                showMessage(getResources().getString(R.string.stop_msg),
                    false);
            }
            enableWidget(false);
            break;
        case R.id.kill:
            if (ActionType.Stop == this.action) {

                if (edtKillPassword.getText().toString().length() != 8) {
                    Toast.makeText(getApplicationContext(), "Kill PWD's length is must be 8",
                        Toast.LENGTH_SHORT).show();
                    return;
                }

                if (mask.getMask().equals("")) {
                    reader.kill(edtKillPassword.getText().toString());
                } else {
                    reader.kill(edtKillPassword.getText().toString(), mask.getMask());
                }

                showMessage(
                    getResources().getString(R.string.access_kill_label),
                    true);
            } else {
                reader.stopOperation();
            }
    }
}
```

```

        displayMessage(getResources().getString(R.string.stop_msg),
            false);
    }
    enableWidget(false);
    break;
case R.id.clear:
    clearTagList();
    break;
case R.id.mask:
    mask.show();
    break;
}
}

```

Lock 메서드를 이용하여 각 메모리 부분별로 잠금, 잠금 해제, 영구 잠금, 등을 수행할 수 있는데, 잠금을 수행하는 부분은 AccessView.java파일의 onItemClick메서드에 구현되어 있다.

```

@Override
public void onItemClick(AdapterView<?> parent, View view, int position,
    long id) {
    Log.d(TAG, String.format("onItemClick(%d, %d, %d)", parent.getId(),
        position, id));

    final int pos = position;

    if (!this.enabledWidgets)
        return;

    AlertDialog.Builder builder = new AlertDialog.Builder(getContext());
    builder.setTitle(getResources().getString(
        R.string.bank_type_prompt_label));
    builder.setSingleChoiceItems(R.array.lock_type_array, adapterOption
        .getLock(position).value(),
        new DialogInterface.OnClickListener() {

            public void onClick(DialogInterface dialog, int which) {
                adapterOption.setLock(pos, LockType.valueOf(which));
                dialog.cancel();
            }
        });
    builder.setNegativeButton(
        getResources().getString(R.string.cancel_button_label), null);
    builder.show();
}

DialogInterface.OnClickListener maskListener = new DialogInterface.OnClickListener() {

    @Override
    public void onClick(DialogInterface dialog, int which) {
        reader.setSelectionBank(mask.getBank());
        reader.setSelectionOffset(mask.getOffset());
        reader.setSelectionAction(mask.getAction());
        mask.setMask(mask.getMask());
    }
}

```

```
}  
};
```

Lock 메서드도 Inventory메서드와 같이 Selection Mask를 사용할 수 있다.

Lock 메서드를 호출하면 AT288N Device가 Tag 메모리를 잠금이나 잠금 해제 동작에 들어간다.

AT288N Device가 Tag Memory를 잠그거나 해제하고 나면 결과는 Response이벤트로 반환되고 동작을 중지된다.

AT288N Device가 Tag Memory를 잠그기 전에 Memory 잠금 동작을 취소하고 싶다면 StopOperation메서드를 호출하여 동작을 취소할 수 있다.