



ATID Co.,Ltd

AT288N Program Guide for Android Developers

AT288Reader API Reference Guide

SDK Team

2023-06-12

Revision History

Version	Revision Date	Reason ¹	Contents of Revision ²	Author
v0.1	2016-05-25	Draft		KJ, Min
v0.2	2019-01-21	Modify	Screen shot and comments are changed to Android Studio version.	YJ, Cho
v0.3	2023-06-12	Revision	Update tool version, BT permissions	SW Team

¹ Revision Reason : Compared to the previous file, add, modify or delete the contents of enactment or revision

² Revision Content : States the page number and changed contents that where to be revised.



	AT288N Program Guide for Android Developers						
AT288Reader API Reference Guide					Company	ATID Co.,Ltd	
DOC		Author	SDK Team	Date	2023-06-12	Version	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.....	26


		AT288N Program Guide for Android Developers					
AT288Reader API Reference Guide					Company	ATID Co.,Ltd	
DOC		Author	SDK Team	Date	2023-06-12	Version	v0.3

1. Intro

This document explains AT288N SDK library building & development environment and provides overall guide for Android developers who want to develop an application using AT288N SDK Library.

Android Studio Chipmunk is used in this document and minSdkVersion is 18.

- ※ From AT288N SDK Library version v1.10.2019012200, Android Studio is used instead of Eclipse.

		AT288N Program Guide for Android Developers					
AT288Reader API Reference Guide					Company		ATID Co.,Ltd
DOC		Author	SDK Team	Date	2023-06-12	Version	v0.3

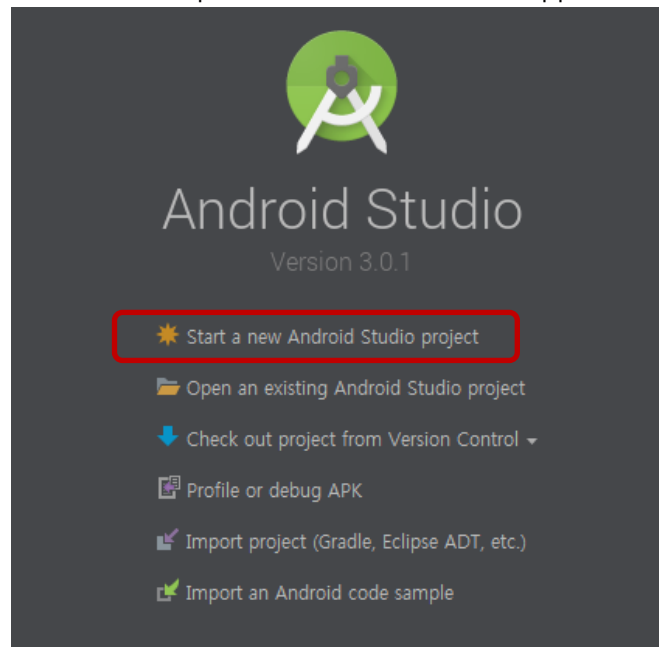
2. Getting Started

Getting started explains how to make simple Android sample and develop the application using the AT288N SDK Library. Furthermore, it explains how to prepare the development environment for AT288N SDK Library.

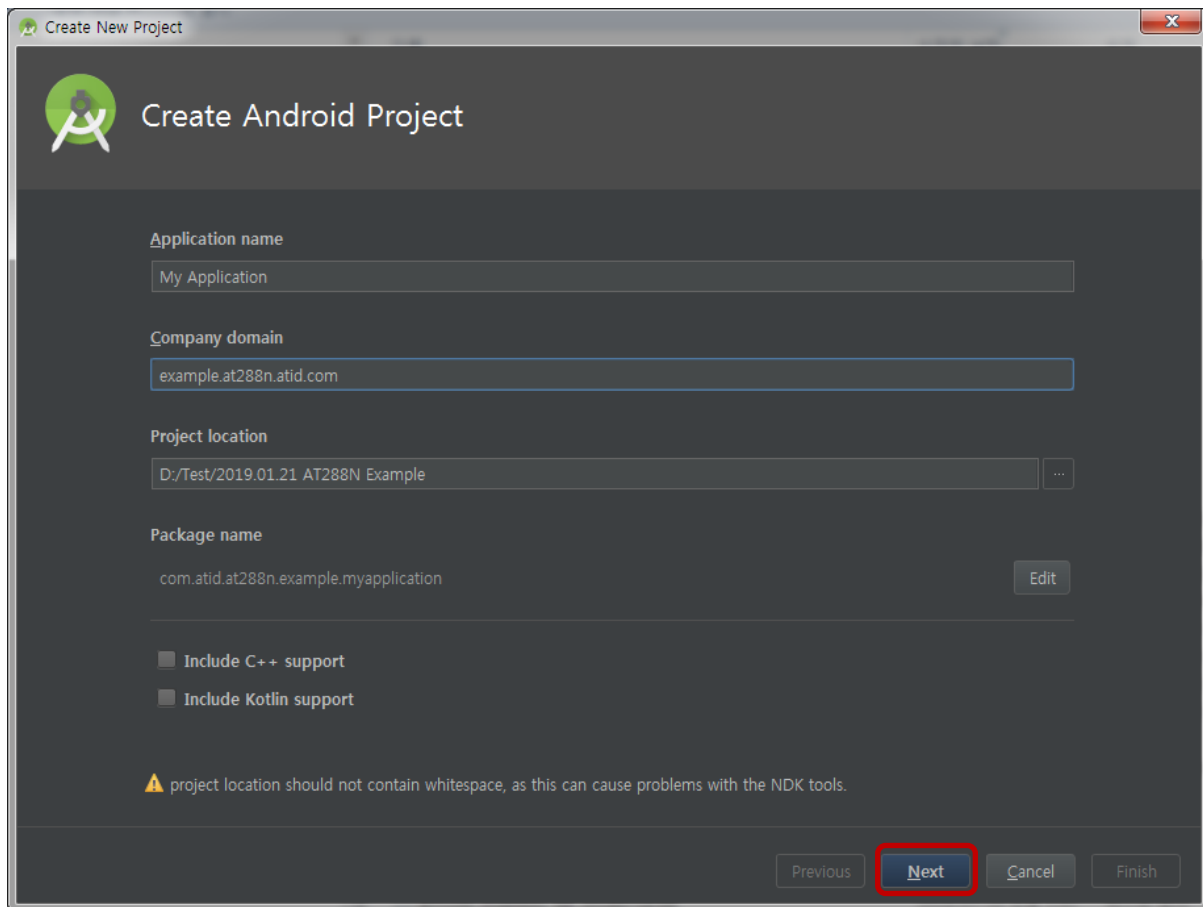
Development tool used here is Android Studio Chipmunk.

2.1. Create Project for Android

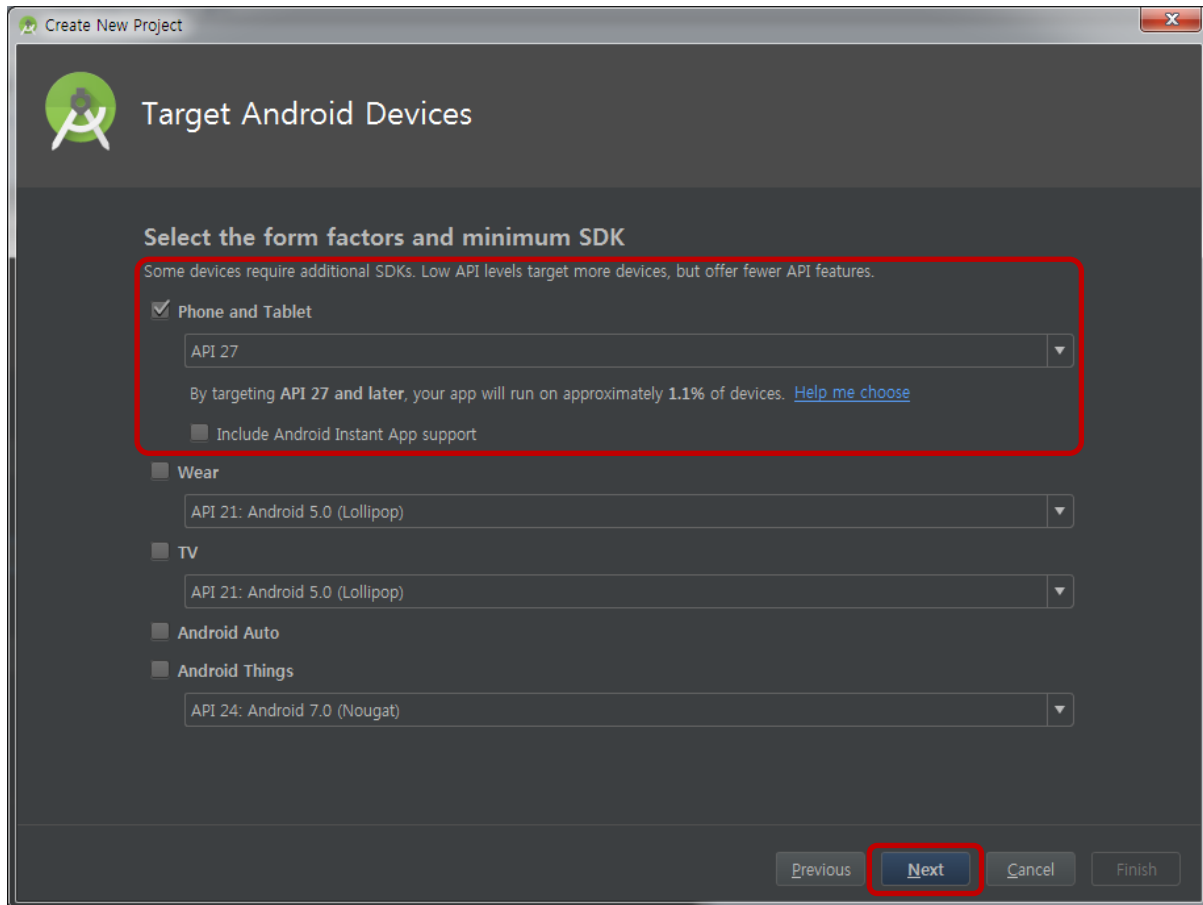
Launch the Android Studio for development of AT288N Android application.



Select "Start a new Android Studio project" to make new project.



When "Create New Project" dialog appears, select your project location and click "Next" button.



Create New Project

Target Android Devices

Select the form factors and minimum SDK

Some devices require additional SDKs. Low API levels target more devices, but offer fewer API features.

☒ Phone and Tablet

API 27

By targeting API 27 and later, your app will run on approximately 1.1% of devices. [Help me choose](#)

☐ Include Android Instant App support

☐ Wear

API 21: Android 5.0 (Lollipop)

☐ TV

API 21: Android 5.0 (Lollipop)

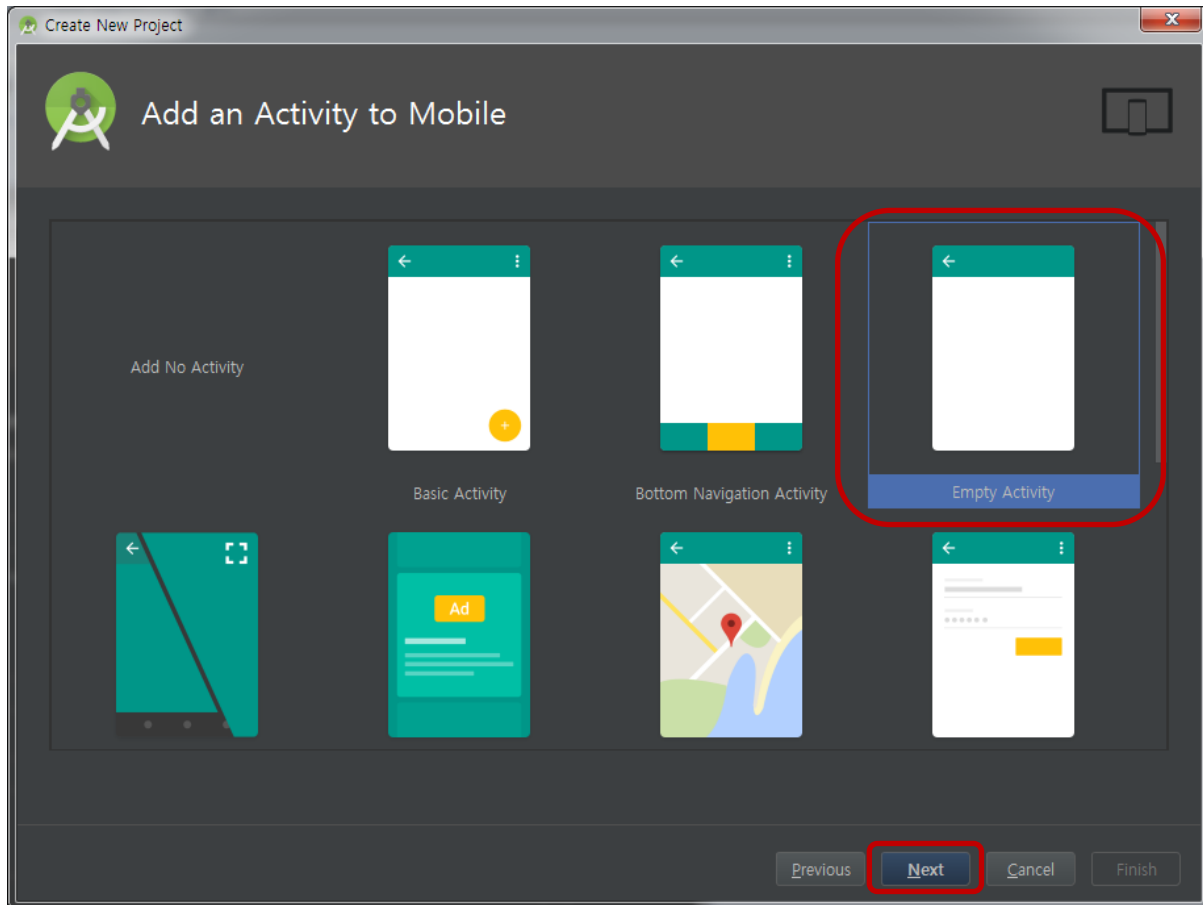
☐ Android Auto

☐ Android Things

API 24: Android 7.0 (Nougat)

Previous **Next** Cancel Finish

Select the form factors and minimum SDK. And click "Next" button.



Select "Empty Activity", Click "Next" button.


```

apply plugin: 'com.android.application'

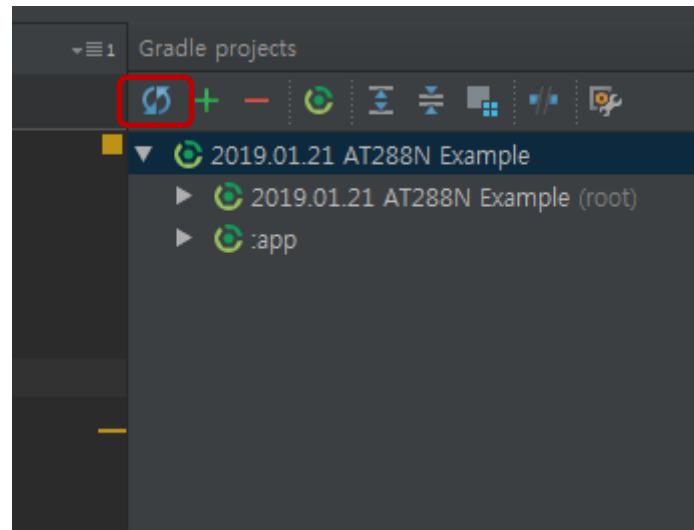
android {
    compileSdkVersion 27
    defaultConfig {
        applicationId "com.atid.at288n.example.myapplication"
        minSdkVersion 21
        targetSdkVersion 26
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }
    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')
}

```

Open the build.gradle file of your app, when the project is all made.

Set the SDK version and add at288lib.aar to dependencies. In above picture, at288lib.aar is already copied to libs folder.



After modifying build.gradle file, press "Sync" button to check if there is no error.

If you get an error, follow the instructions in Android Studio.

If you do not get an error when you build the project, you're ready.

3. Programing Guide

3.1. Add Permission

Please set up the Bluetooth permission for using the AT288N SDK Library at the Android.

Add User-Permissions are as below.

```
<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" />
```

It needs BLUETOOTH_ADMIN Permission which can control Bluetooth device using User-Permission. And also, the BLUETOOTH permission is needed for using Bluetooth Device.

In the case of Android 12 and above versions, additional permissions are required.

```
<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);
    }
}
```

And, add method call in onCreate(),

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

3.2. Create and Synchronization

3.2.1. Create Reader

To Use AT288N Device via AT288N SDK Library, the instance of the class Reader should be generated. The Generating the instance of the class Reader can be checked in the createReader method of sample source file

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

createReader method is invoked from onCreate method (which is overridden method of MainActivity). It is most suitable to create an instance of the class Reader when onCreate event is called as MainActivity is create. When you create an instance of the class Reader MainActivity and event handlers instance is passed as a parameter.

3.2.2. Synchronize Reader

To manage Life Cycle of Main Activity and Reader, redefine the overridden methods such as onCreate, onStart, onResume, onPause, onStop, onDestroy, onActivityResult etc.. If needed, to synchronize the reader, you can call same method from the each redefined method.

```
@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.");
    }
}

```

Note that the onActivityResult method should override the method to be called openDeviceListActivity the Reader class onActivityResult method can be normal behavior. openDeviceListActivity Method calls Activity through Intent and receives the results via onActivityResult of MainActivity. For this process to be executed in AT288N SDK Library, the call to onActivityResult of Reader is required.

3.3. Connect and Activate

There are two ways to connect to AT288N Device from AT288N SDK Library. First is to connect by scanning surrounding AT288N Device. Second is to connect to the recently connected AT288NDevice.

It is shown how to connect to AT288N Device by using the Reader's object in the MainActivity's onOptionsItemSelected method.

```

@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);
}

```

However, in case of connecting to the recently connected AT288N Device, there might be no AT288N Device recently connected or there might be no AT288N Device near the host. In these cases, connectReader method is implemented for calling openDeviceListActivity.

```

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

```

After connecting to AT289N device, the MESSAGE_STATE_CHANGE event is coming out as STATE_CONNECTED value, this time call the Activity method for AT288 device Reader object synchronization.

```

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

```



AT288N Program Guide for Android Developers

AT288Reader API Reference Guide

Company

ATID Co.,Ltd

DOC

Author

SDK Team

Date

2023-06-12

Version

v0.3

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

3.4. Event Handler

To receive the event from AT288N Device, Handler is entered to Reader as parameter. Reader generates events that are received from AT288N Device via Handler.

Interface members will be invoked when it is ready to invoke.

Thus, you have to handle it correctly in interface member.

onReaderStateChange was implemented in MainActivity.java.

It will be invoked when connection state has been changed.

```
@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);
}

```

onReaderTimeout was implemented in MainActivity.java.

It will be invoked when timeout has been occurred.

```

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

```

onReaderReadTag was implemented in MainActivity.java.

It will be invoked when device reads the tags by inventory command.

```

@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);
}

```

onReaderResponse was implemented in MainActivity.java.

It will be invoked when device finished Access(read/write/lock/kill) operation.

```

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

```

onReaderActionChange was implemented in MainActivity.java.

It will be invoked when Action has been changed.

```
@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);
}
```

onReaderProperty was implemented in MainActivity.java.

It will be invoked when properties have been read by user.

```
@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);
}
```

onReaderExtendedProperty was implemented in MainActivity.java.

It will be invoked when extended properties have been read by user.

```
@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.onReaderExtendedProperty(code, value);
}
}

```

If the developer wants to see more examples about properties and extended properties, refer to onReaderProperty method of OptionView.java and onReaderExtendedProperty method.

```

@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.java's onReaderExtendedProperty method has example of Power Gain, Tag Type, Connect Type and Read Type etc.

```

@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.adapter InventoryFormat);
    }

    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

Use the Inventory method for Tag inventory by AT288N device. Please check InventoryView.java file's onClick method for knowing that How to use the Inventory method.

```
@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;
    }
}
```

In case of Inventory method, transfer the null value to the parameter when Selection Mask is not used. And when Selection Mask is used, methods such as setSelectionBank, setSelectionOffset, setSelectionAction must be used.

"How to set Selection Mask" is explained in maskListener of InventoryView.java file.

```
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());
    }
}
```

		AT288N Program Guide for Android Developers					
AT288Reader API Reference Guide					Company	ATID Co.,Ltd	
DOC		Author	SDK Team	Date	2023-06-12	Version	v0.3

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

To use Selection Mask, bring the current value of Selection Mask and set the value via methods like `getSelectionBank`, `getSelectionOffset`, `getSelectionAction` etc. Then the inventory method can be called by entering parameter Mask value. When the Inventory method is called, AT288N Device starts Inventory and the tag value read by AT288N Device in `onReaderReadTag` event is returned.

3.5.2. Read Memory and Write Memory

To read the Tag memory, use readMemory method. To write on the Tag memory, use writeMemory method. readMemory method and writeMemory method is placed in onClick method of MemoryView.java file.

```
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;
}
}

```

readMemory method is calls specificTag's memory bank, start reading address and Memory read length.

readMemory method uses selectionMask as Inventory method.

If AT288N device is reading Tag memory normally, the onReaderReadTag event is coming out and stopping the reading Tag memory. And if it is failed, onReaderResponse events is coming out and stopping reading Memory. If the AT288N device wants to stop reading Memory before reading Tag memory, call the stopOperation method.

wirteMermory method is calling specifying Tag Memory bank which for writing data, start writing address and tag Memory. writeMemory method can use selectionMask as readMemory.

AT288N Device finish to write Tag memory, Response to the event success and failure is returned and the behavior will stop.

If the AT288N device wants to stop writing Tag Memory before writing Memory, call the stopOperation method.

3.5.3. Lock

For locking Tag Memory, use Lock method which located in the AccessView.java file's onClick method.

```
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(getContext(), "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(getContext(), "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;
}
}

```

By using Lock method, it is possible to working the part of Memory lock, unlock, permanent lock etc. It is in the AccessView.java file's lock method.

```

@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());
    }
}

```

	AT288N Program Guide for Android Developers						
AT288Reader API Reference Guide					Company	ATID Co.,Ltd	
DOC		Author	SDK Team	Date	2023-06-12	Version	v0.3

```

    }
};

```

Lock method also can use Selection Mask as Inventory method.

When the Lock method is called, AT288N Device start working the lock Tag memory or unlocking. If AT288N device lock or unlock the Tag memory, the result is sent to Response event and stop working.

To stop the memory unlock before AT288N Device unlocks Tag Memory, call stopOperation method to stop the process.