



ATID Co.,Ltd

XC1003 RFID


Program

Developers Guide

Android Developer Guide

Y.H. Park

2015-04-23

		XC1003 RFID Program Developers Guide					
Android Developer Guide					Company	ATID Co.,Ltd	
Doc.		Writer	Y.H.Park	Date	2015-04-23	Version	v1.0

Document Change Record

Version	Date	Revision ¹	Description of Change ²	Writer
V1.0	2015-04-23	Draft	1st Draft	Y.H.Park

¹ Revision: Define the contents are addition/ modification/ deletion.

² Description: Describe revised page number and contents.

		XC1003 RFID Program Developers Guide					
Android Developer Guide					Company	ATID Co.,Ltd	
Doc.		Writer	Y.H.Park	Date	2015-04-23	Version	v1.0

Table of Contents

Table of Contents	3
1. Intro	4
2. Getting Started	5
2.1. Create Project for Android.....	5
2.2. Development Environment	13
3. Programing Guide.....	17
3.1. Create and Synchronization	17
3.1.1. Create Reader.....	17
3.1.2. Destroy Reader	20
3.2. Event Handler.....	21
3.3. Action Operation.....	23
3.3.1. Inventory and StopOperation.....	23

	XC1003 RFID Program Developers Guide						
Android Developer Guide					Company	ATID Co.,Ltd	
Doc.		Writer	Y.H.Park	Date	2015-04-23	Version	v1.0

1. Intro

This document explains about the XC1003 RFID SDK Library to Android developers who wants to build development environment and use SDK Library in developing application programs..

Eclipse IDE for Java Developers is used and the development platform Android 4.4.2 is supported.

		XC1003 RFID Program Developers Guide					
Android Developer Guide					Company	ATID Co.,Ltd	
Doc.		Writer	Y.H.Park	Date	2015-04-23	Version	v1.0

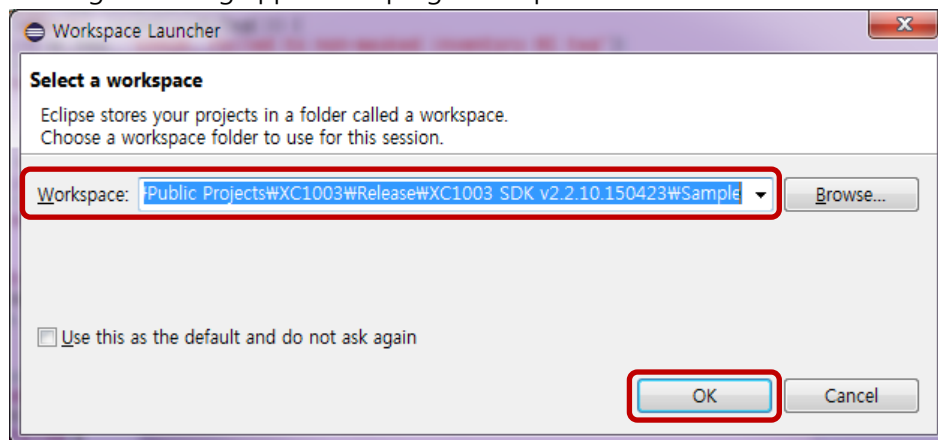
2. Getting Started

In Getting Started, the steps in building development environment for using XC1003 RFID SDK Library and building application program are explained.

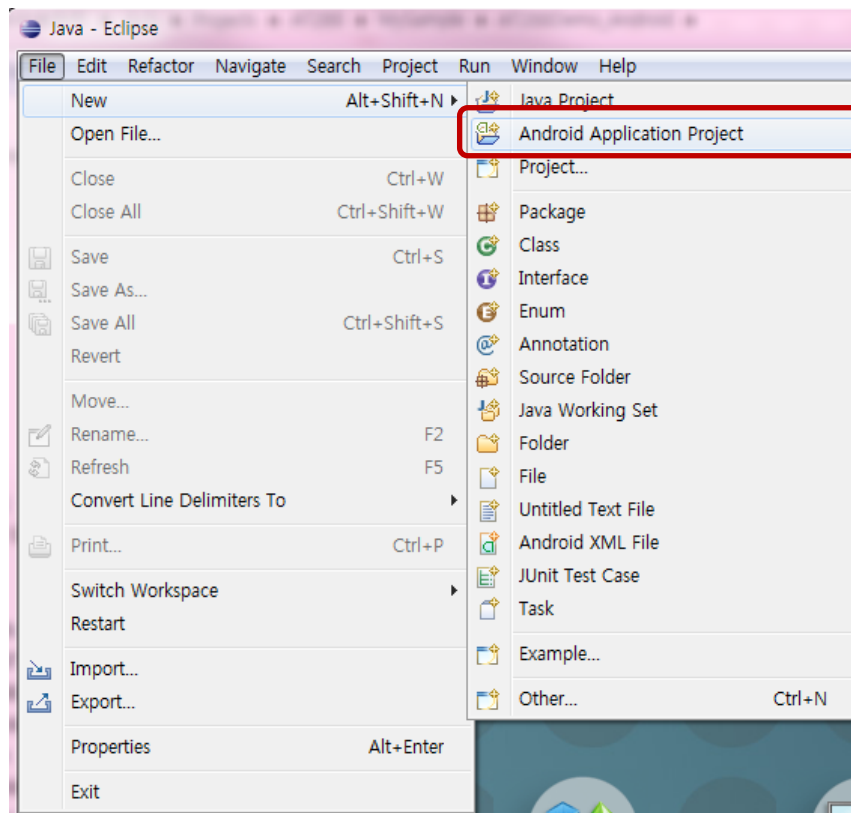
Eclipse Juno is used for the demonstration.

2.1. Create Project for Android

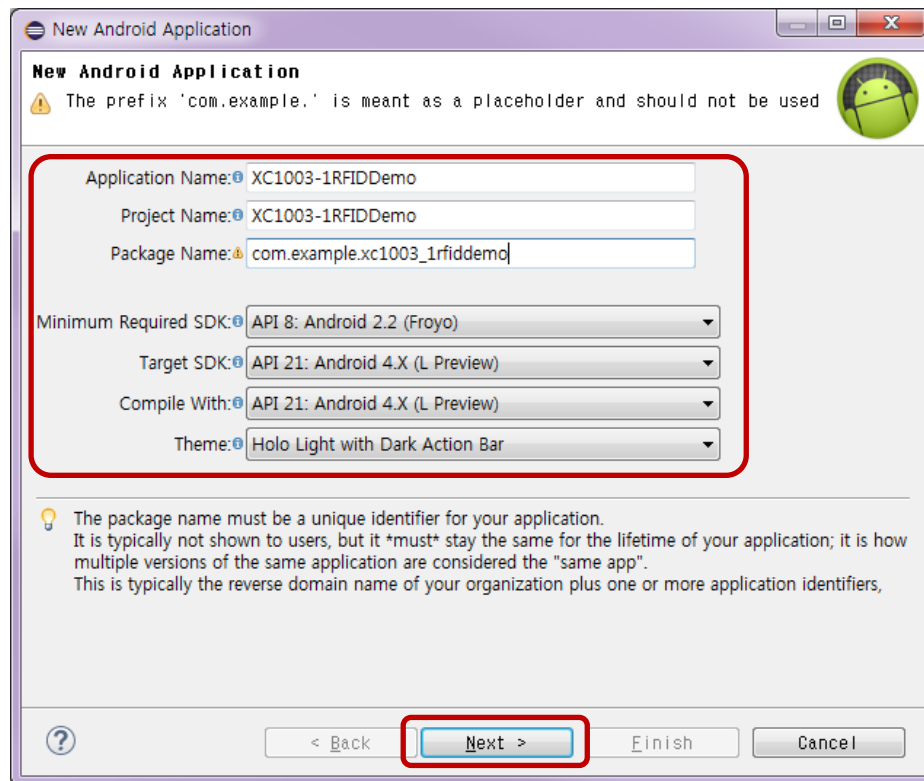
Run Eclipse to begin building application program steps for XC1003 RFID Android.



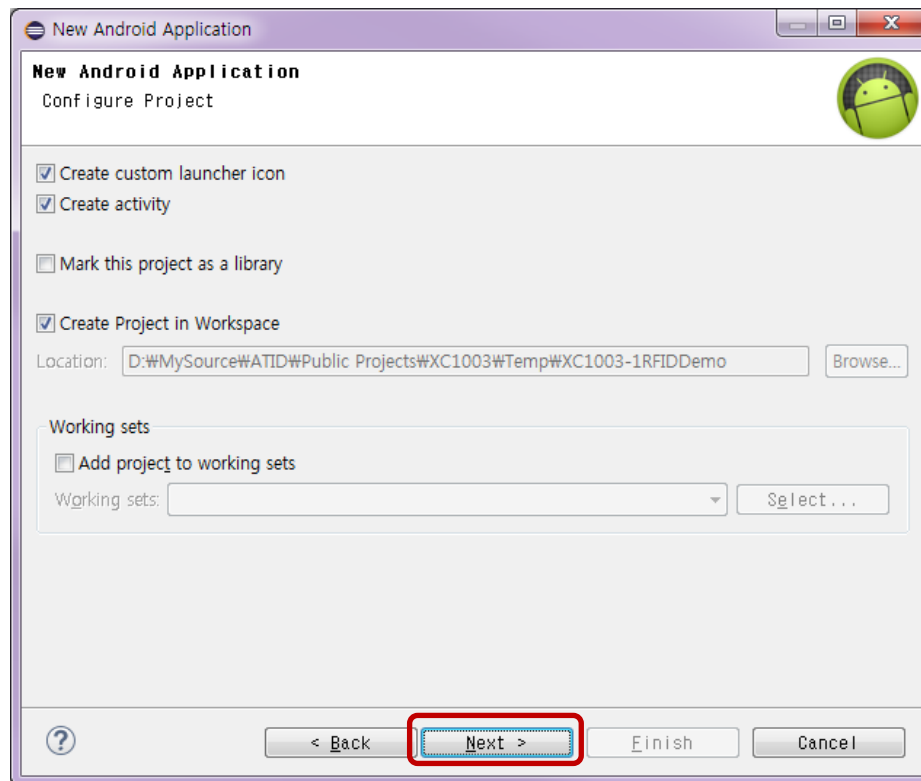
When above dialogue box is shown, locate the project folder and click "OK".



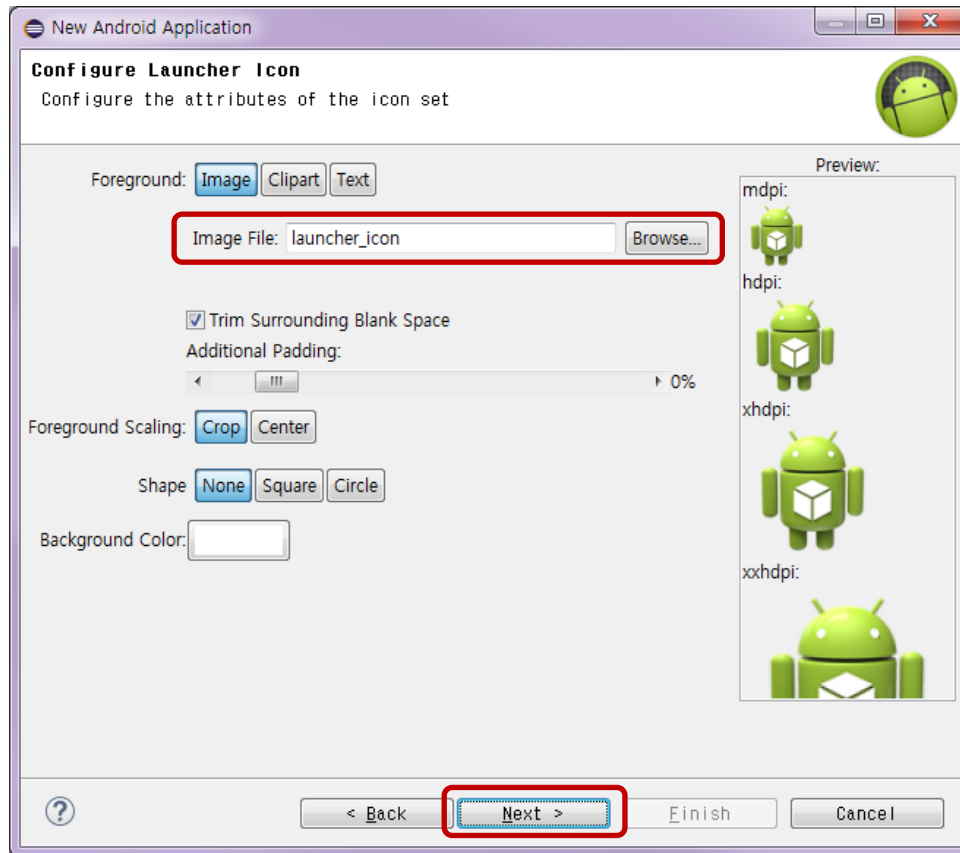
Select "File" → "New" → "Android Application Project" from the file menu.



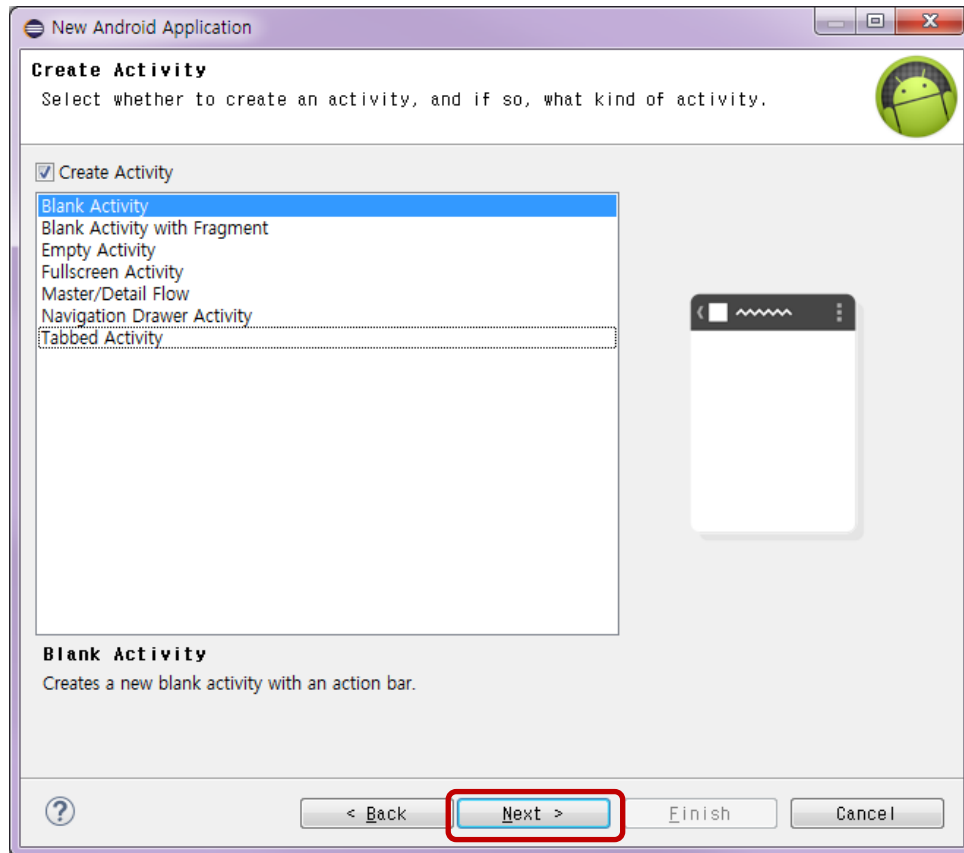
When New Android Application window is displayed, fill in "Application Name", "Project Name" and "Package Name". Select "Minimum Required SDK", "Target SDK", "Compile With" and "Theme" then, click "Next" button to proceed.



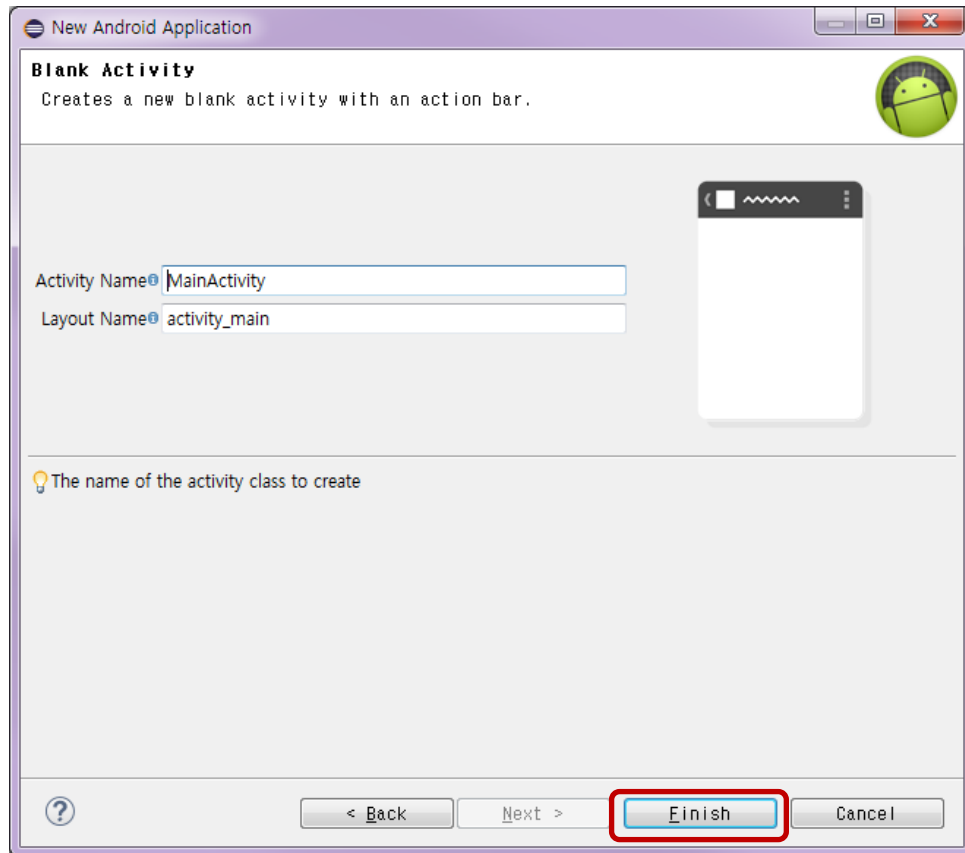
Click "Next" to continue.



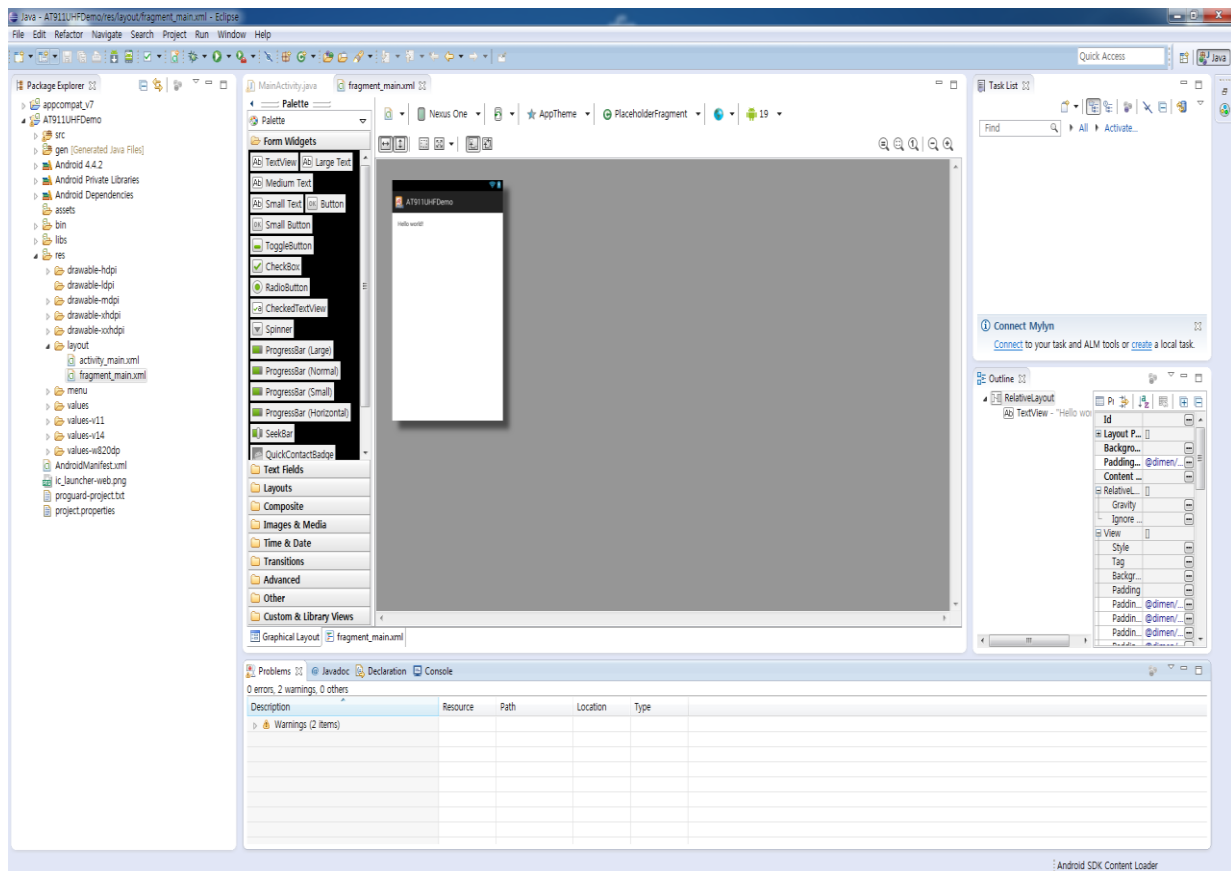
Select the Icon that would be used in Application and click "Next" to proceed.



Select Application의 Theme and click "Next" to proceed.



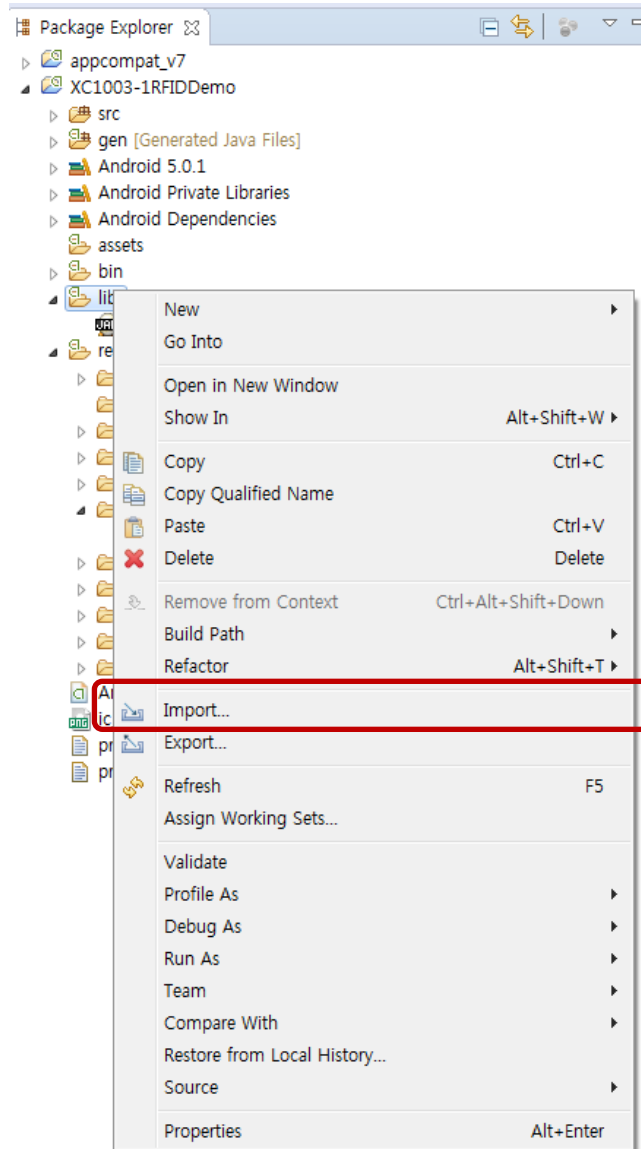
Fill in "Activity Name" and "Layout Name", then click "Finish".



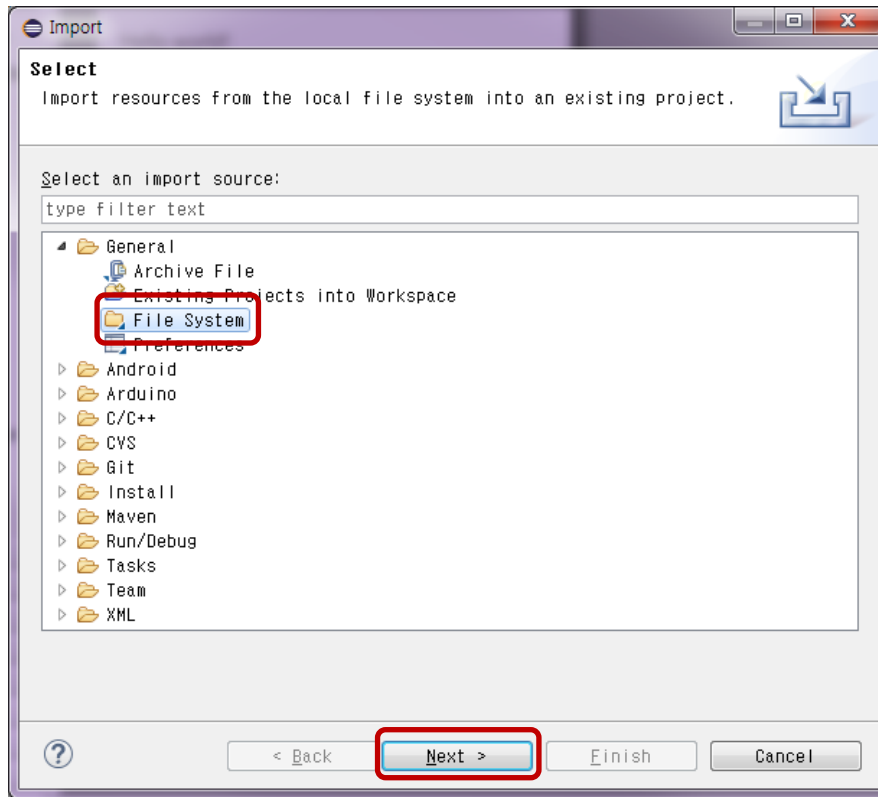
Eclipse screen is shown as above when the project creation is completed.

2.2. Development Environment

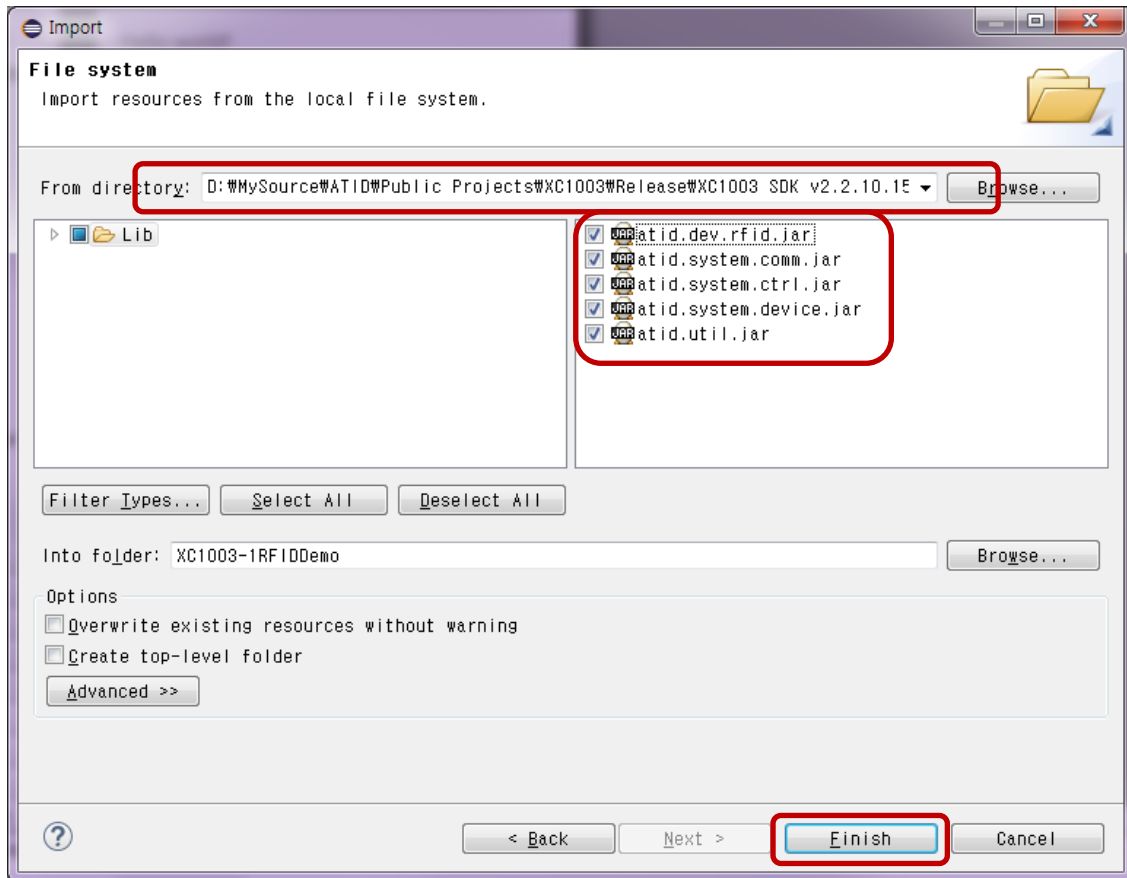
XC1003 RFID SDK Jar file is required to develop XC1003 RFID Android application program.



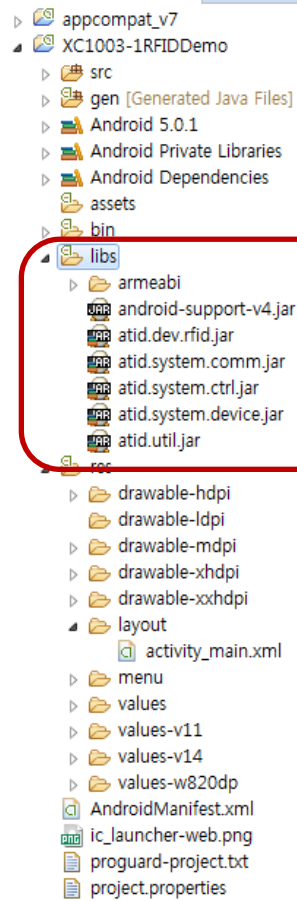
To import XC1003 RFID SDK jar file, select libs folder from Package Explorer. Right-click on the folder and select "Import" from the pop-up menu.



Select "General" → "File System" and click "Next" to proceed.



Select Android folder under Library folder and you can see atid.dev.rfid.jar file. Check all the checkboxes and click "Finish". atid.dev.rfid.jar file will be copied to libs folder created by the project.



You can see the jar file is copied to Package Explorer.

3. Programing Guide

3.1. Create and Synchronization

3.1.1. Create Reader

To use XC1003 RFID via XC1003 RFID SDK Library, the instance of ATRfidReader must be created first. Creating the instance can be checked from the onCreate method in MinActivity.java file of the sample source.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    Log.i(TAG, "INFO. onCreate()");

    // Setup always wake up
    if (this.wakeLock == null) {
        PowerManager powerManager = (PowerManager)
getSystemService(POWER_SERVICE);
        this.wakeLock = powerManager.newWakeLock(
            PowerManager.FULL_WAKE_LOCK, APP_NAME);
        this.wakeLock.acquire();
    }

    // Display Version
    String packageName = getPackageName();
    String versionName = "";
    try {
        versionName =
getPackageManager().getPackageInfo(packageName,
            PackageManager.GET_META_DATA).versionName;
    } catch (NameNotFoundException e) {
    }
    this.txtDemoVersion.setText(versionName);

    enableButtons(true);

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

In XC1003, it is recommended to create from the onResume method to control the instance of UHF Device.

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

    Log.i(TAG, "EVENT. onResume()");
}
```

```
// Initialize Scanner Instance
if ((this.reader = ATRfidManager.getInstance()) == null) {
    // Show Error Alert Dialog
    AlertDialog.Builder alert = new AlertDialog.Builder(this);
    alert.setOnCancelListener(new
DialogInterface.OnCancelListener() {

        @Override
        public void onCancel(DialogInterface dialog) {
            finish();
        }
    });
    alert.setPositiveButton(R.string.ok_button,
        new DialogInterface.OnClickListener() {

            @Override
            public void onClick(DialogInterface
dialog, int which) {

                finish();
            }
        });
    alert.setIcon(android.R.drawable.ic_dialog_alert);
    alert.setTitle(R.string.device_error);
    alert.setMessage(R.string.filaed_to_device);
    alert.show();
    return;
}

this.txtFirmwareVersion.setText(this.reader.getFirmwareVersion());
}
```

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

    switch (requestCode) {
        case INVENTORY_VIEW:
        case READ_MEMORY_VIEW:
        case WRITE_MEMORY_VIEW:
        case LOCK_MEMORY_VIEW:
        case OPTION_VIEW:
            enableButtons(true);
            break;
    }
    super.onActivityResult(requestCode, resultCode, data);
}
txtLogo.setTextColor(getResources().getColor(R.color.connected_device));
}
```

You must redefine onActivityResult method and then call onActivityResult method of Reader class for it to function properly. OpenDeviceListActivity method calls Activity via Intent from the inside. And the result value is returned via onActivityResult of Main Activity. To execute this within

		XC1003 RFID Program Developers Guide					
Android Developer Guide					Company	ATID Co.,Ltd	
Doc.		Writer	Y.H.Park	Date	2015-04-23	Version	v1.0

XC1003 RFID SDK Library, it needs to call OnAcitivityResult. Also, add RfidReaderEventListener for the synchronization with the reader.

3.1.2. Destroy Reader

To manage the Life Cycle of Main Activity and Reader, overridden methods like onCreate, onDestroy, onActivityResult etc. can sometimes be redefined. And if required, the developers can call the same methods from its redefined methods to synchronize the reader.

```
@Override
protected void onDestroy() {

    Log.i(TAG, "EVENT. onDestroy()");

    // Setup basic wake up state
    if (this.wakeLock != null) {
        this.wakeLock.release();
        this.wakeLock = null;
    }
    super.onDestroy();
}
```

```
@Override
protected void onPause() {
    Log.i(TAG, "EVENT. onPause()");

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

    super.onPause();
}
```

3.2. Event Handler

To receive reader event from XC1003 RFID, declare RfidReaderEventListener to class and receive Event Handler in parameter. Reader object creates the events that are received by the reader. Classes of event are divided into onReaderStateChanged, onReaderDetectTag, onReaderResult. The following assigns Handler objects that are embodied in MainActivity.java.

```
public class InventoryView extends AccessView implements RfidReaderEventListener,
    OnClickListener, OnItemSelectedListener {
    @Override
    protected void onResume() {
        super.onResume();

        Log.i(TAG, "EVENT. onResume()");

        // Initialize RFID Reader
        getReader().setEventListener(this);

        // Initialize Power Gain
        int i = 0;
        RangeValue powerRange = getReader().getPowerRange();
        for (i = powerRange.max; i >= powerRange.min; i--) {
            this.adpPower.addItem(powerRange.max - i, "" + i + " dBm");
        }
        this.adpPower.notifyDataSetChanged();
        this.spnPower.setSelection(this.adpPower
            .indexOf(getReader().getPower()));

        // Enable Widgets
        enableWidgets(true);
    }

    @Override
    protected void onPause() {
        Log.i(TAG, "EVENT. onPause()");

        super.onPause();
    }

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

        enableWidgets(true);

        super.onActivityResult(requestCode, resultCode, data);
    }
}
```

onReaderStateChange method receives events that are related to Actions in XC1003. Refer to the following and the sample program for the detail.

```
@Override
public void onReaderStateChanged(ReaderState state) {
    enableWidgets(true);
}
```

onReaderDetectTag method creates events when XC1003 Tag is detected.

```
@Override
public void onReaderDetectTag(String tag) {
    // Add Tag List
    this.adpTags.addItem(tag);
    // Display Tag Count
    this.txtCount.setText("" + this.adpTags.getCount());
    // Play Beep
    beep();
}
```

onReaderResult method checks the results when XC1003 Memory Access, Tag Access etc. are carried out.

```
@Override
public void onReaderResult(ResultCode code) {
    Log.e(TAG, "ERROR. Failed to onReaderResult(" + code + ")");
}
```

3.3. Action Operation

3.3.1. Inventory and StopOperation

Use inventory6cTag, inventory 6bTag, readEpc6cTag, readEpc6cTag methods for the Inventory of Tag. The details on using Inventory method can be found in InventoryView.java file.

```
// Start Action
private void startAction() {

    Log.i(TAG, "INFO. Start Action()");

    if (getContinuousMode()) {
        // Multi Tag Inventory...
        if (!mReader.inventory6cTag()) {
            Log.e(TAG, "ERROR. Failed to non-masked inventory 6C
tag");

            //enableWidgets(true);
            return;
        }
    } else {
        // Single Tag Inventory...
        if (!mReader.readEpc6cTag()) {
            Log.e(TAG, "ERROR. Failed to non-masked read EPC 6C tag");
            //enableWidgets(true);
            return;
        }
    }

    mAction=true;
    enableWidgets(false);
    //adpTags.start();
}

// Stop Action
private void stopAction() {
    Log.i(TAG, "INFO. Stop Action()");
    mAction=false;

    mReader.stop();
    //adpTags.shutdown();

    //enableWidgets(false);
    enableWidgets(true);
}
```