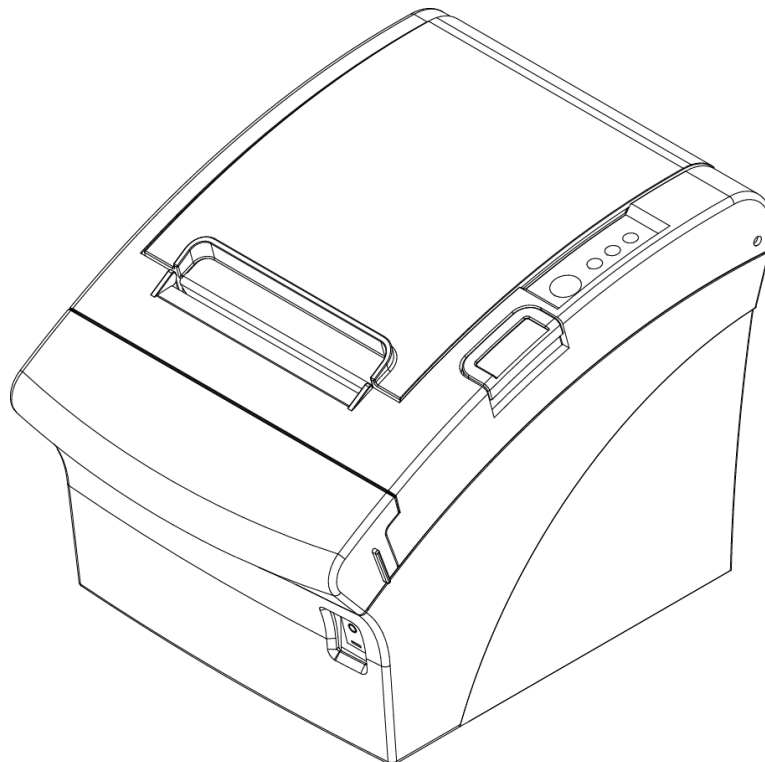


METAPACE

Software Manual Metapace T-3II SDK for Android

Rev. 1.00

Metapace T-3II



<http://www.metapace.com>

■ Table of Contents

1. About This Manual	4
2. Metapace T-3II SDK for Android Overview	5
2-1 Features	5
2-2 Functions	5
3. Operation Environment	6
3-1 Android Version.....	6
3-2 Printer Interface	6
4. Development Environment	7
4-1 System Requirement	7
4-1-1 Operating System	7
4-1-2 Eclipse IDE	7
4-2 Connecting Android Device	8
4-2-1 Network	8
4-2-2 USB	9
4-3 Importing Library and Running Sample Application	11
4-3-1 How to import Android project in Eclipse	11
4-3-2 How to run/debug the project.....	11
5. Package contents	12
5-1 Package.....	12
6. Sample Program	13
6-1 Functions	13
6-2 Environment Configuration	13
6-3 How to Use Sample Program	14
6-3-1 Search and Connect Printer	14
7. API Reference.....	15
7-1 Create printer instance	15
7-1-1 Constructor	15
7-2 Search Printer.....	17
7-2-1 findNetworkPrinters	17
7-2-2 findUsbPrinters	19
7-3 Connect Printer.....	21
7-3-1 connect.....	21
7-3-2 connect.....	23
7-3-3 connect.....	25
7-3-4 disconnect	27
7-4 Print	28
7-4-1 formFeed	28
7-4-2 lineFeed.....	29
7-4-3 print1dBarcode	31
7-4-4 printBitmap	33
7-4-5 printBitmap	35
7-4-6 printPdf417	37
7-4-7 printQrCode	39

Metapace T-3II SDK for Android

7-4-8 printQrCode	41
7-4-9 printSelfTest	43
7-4-10 printText	44
7-5 Receive Printer Response	46
7-5-1 automateStatusBack	46
7-5-2 getPrinterId	48
7-5-3 getStatus	50
7-6 NV Image	52
7-6-1 defineNvlImage	52
7-6-2 defineNvlImage	54
7-6-3 getDefinedNvlImageKeyCodes	56
7-6-4 printNvlImage	58
7-6-5 removeAllNvlImage	60
7-6-6 removeNvlImage	61
7-7 Page Mode	63
7-7-1 setAbsolutePrintPosition	63
7-7-2 setAbsoluteVerticalPrintPosition	63
7-7-3 setPageMode	63
7-7-4 setPrintArea	64
7-7-5 setPrintDirection	64
7-7-6 setStandardMode	64
7-8 Settings	66
7-8-1 setSingleByteFont	66
7-9 Miscellaneous Functions	68
7-9-1 cutPaper	68
7-9-2 cutPaper	70
7-9-3 executeDirectlo	72
7-9-4 getMacAddress	74
7-9-5 initialize	76
7-9-6 kickOutDrawer	78
7-9-7 updateFirmware	80
8. Programming	82
8-1 Programming Flow	82
8-2 Search Printer	82
8-3 Open Printer Port	83
8-4 Send Print Data	84
8-5 Close Printer Port	85

1. About This Manual

This manual has been prepared to provide the information required to configure and design Android applications using Metapace T-3II.

Metapace constantly makes improvements to the functions and quality and the specifications of the product and the contents of this manual are subject to change without prior notice for this reason.

2. Metapace T-3II SDK for Android Overview

2-1 Features

- This SDK has been designed to make printing with Metapace printer easier using Android applications.
- Android applications can easily check the status of the printer with this SDK.

2-2 Functions

- Print Settings (Alignment / Page Mode)
- Charter Data Settings (Code Page / Device Font Type)
- Character Style Settings (Bold / Reverse / Underline)
- Image Printing (Raster Bit / NV Graphics)
- One-Dimensional Barcode Printing
- Two-Dimensional Barcode Printing
- Cash Register Open Function / Melody Box Function
- Printer Command Transmission
- Printer Response Reception (Printing Result / Printer Status)

3. Operation Environment

3-1 Android Version

- Printing over Ethernet or WLAN: Android 2.2 (Froyo) or higher
- Printing over USB: Android 3.1 (Honeycomb) or higher

3-2 Printer Interface

- Ethernet
- WLAN
- USB

4. Development Environment

4-1 System Requirement

4-1-1 Operating System

- Windows XP (32-bit), Vista (32- or 64-bit), or Windows 7 (32- or 64-bit)
- Mac OS X 10.5.8 or later (x86 only)
- Linux (tested on Ubuntu Linux, Lucid Lynx)
- GNU C Library (glibc) 2.7 or later is required.
- On Ubuntu Linux, version 8.04 or later is required.
- 64-bit distributions must be capable of running 32-bit applications.

4-1-2 Eclipse IDE

- Eclipse 3.6.2 (Helios) or greater
- JDK 6 (JRE alone is not sufficient)
- Android SDK
- ADT(Android Development Tools) plugin
- Reference: <http://developer.android.com/sdk/index.html>

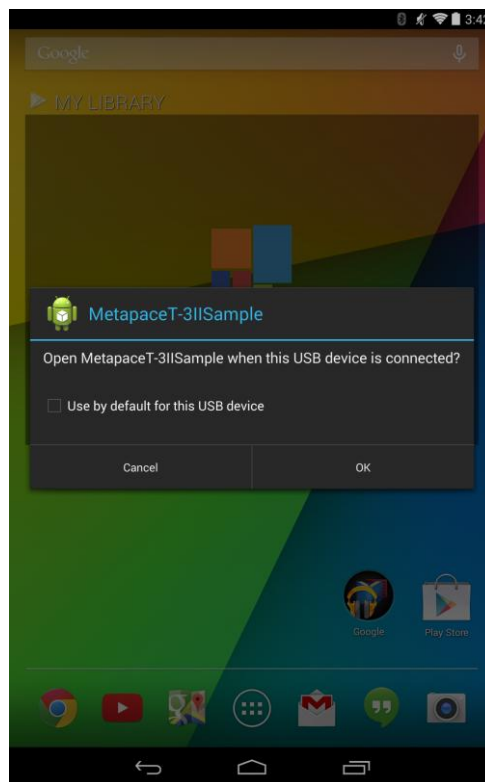
4-2 Connecting Android Device

4-2-1 Network

1. Connect the printer to the network AP (Access Point) and assign an IP address or obtain one using DHCP. Since Metapace T-3II is configured with ad-hoc network from the factory, network should be configured at least once using the Net Configuration Tool that is included in the master CD.
2. Select [Settings].
3. Wi-Fi should be turned on.
4. Connect the device to the same network that the Metapace T-3II is connected to.
5. Additional setting is not required to connect the Android device to the TCP/IP port of printer.

4-2-2 USB

1. Android device can be connected to USB peripheral devices using OS version 3.1 or higher.
2. Special driver or printer software of Metapace does not have to be installed in the Android device.
3. Type of required USB cable depends on the type of smart phone or tablet device. Most Android devices do not come with A to B USB cable and mini/micro USB cable or adapter/dock might be required. Check whether the cable works with the Android device to be used.
4. The following message may be displayed depending on Android device when Metapace T-3II is connected first time.



5. The following code should be entered to AndroidManifest.xml and res/xml/device_filter.xml in order to connect USB peripheral devices.

[AndroidManifest.xml]

```
<intent-filter>
    <action android:name="android.hardware.usb.action.USB_DEVICE_ATTACHED"
/>
</intent-filter>

<meta-data
    android:name="android.hardware.usb.action.USB_DEVICE_ATTACHED"
    android:resource="@xml/device_filter" />
```

[device_filter.xml]

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

<usb-device
    product-id="43"
    vendor-id="5380" />

</resources>
```

4-3 Importing Library and Running Sample Application

ADT (Android Development Tool) plug in should be installed in Eclipse.

4-3-1 How to import Android project in Eclipse

1. Run Eclipse.
2. Select [File – Import].
3. Select [General – Existing Project into Workspace].
4. Select [Browse] and specify the path to the MetapaceT-3IISample.

4-3-2 How to run/debug the project

1. Select [Project – Run/Debug].

5. Package contents

5-1 Package

- Doc/Manual_POS_Printer SDK for Android API Reference Guide_english
_Rev_1_xx.pdf: Manual in English
- lib/T-3II.jar: Printer library
- sample/MetapaceT-3IISample: Sample program project folder

6. Sample Program

This chapter explains how to use the sample program. (MetapaceT-3IISample)
The sample is provided as an Android application project using Eclipse.

6-1 Functions

- Search printer
- Connect printer
- Disconnect printer
- Print text
- Print image file
- Print one-dimensional bar code
- Print two-dimensional bar code
- Print in the page mode
- Cut paper
- Check printer status
- Check printer information

6-2 Environment Configuration

1. Decompress the Metapace T-3II SDK for Android package into any desired target location.
2. Select [File – Import] from Eclipse and select [General – Existing Project into Workspace] from the corresponding dialogue.
3. Select [Browse] and set the path where MetapaceT-3IISample is located.
4. Right-click on MetapaceT-3IISample in the Package Explorer and select [Properties].
5. Select [Add JARs] from the [Libraries] tab of the Java Build Path and select lib/T-3II.jar
6. Right-click on MetapaceT-3IISample in the Package Explorer and select [Run As – Android Application].
7. When the sample program is installed in the target Android device, select the Metapace T-3IISample application and run the program.

6-3 How to Use Sample Program

6-3-1 Search and Connect Printer

1. Run the sample program.
2. Select the [Option] menu and choose one of Bluetooth, Network, or USB interface to make connection.
 - A. If Network is selected, a dialog box containing the list of printer IP addresses that can be connected will be displayed.
 - B. If USB is selected, a dialog box containing the list of device information of printer connected with Android device through USB connection will be displayed.
3. Select the printer to connect from the device list dialog box.
4. Connection is established if “connected to [Printer Model Name or USB Device ID]” appears in the Title area.
5. Printer function list is activated when connection is established.

7. API Reference

This chapter describes the API provided by the Printer SDK.

7-1 Create printer instance

7-1-1 Constructor

Create an instance of T_3II to use the printer.

Methods implemented in the T_3II class are configured for asynchronous operation. When a response is to be received from the printer, other methods can be executed only after reception of the response is completed.

When multiple printers are connected, a separate instance should be created for each printer.

*** Syntax**

```
public T_3II(Context context, Handler handler, Looper looper)
```

*** Parameters**

- context: is a context instance used to access Wi-Fi service, USB service, and file system of Android.
- handler: is application handler to connect and receive print message.
- looper: Set main looper in application to create T_3II instance in a separate thread in order to avoid the collision between the handler in application and the internal handler in printer library. Set it to null to create T_3II instance from the main thread.

* Example

```
public class MainActivity extends Activity {  
  
    ...  
  
    private T_3II t_3ii;  
  
    ...  
  
    public void onCreate(Bundle savedInstanceState) {  
  
        ...  
  
        t_3ii = new T_3II(this, mHandler, null);  
  
        ...  
  
    }  
  
    private final Handler mHandler = new Handler(new Handler.Callback() {  
  
        public Boolean handleMessage(Message msg) {  
  
            ...  
  
        }  
  
    });  
  
}
```


7-2 Search Printer

7-2-1 findNetworkPrinters

This method searches the printers connected to the same network as Android device that runs the application. It sends the MESSAGE_NETWORK_DEVICE_SET message to the application handler when the search operation is completed. The message includes the IP addresses of the printers that can be connected. The return value will be null if no printer is found.

* Syntax

```
public void findNetworkPrinters(int timeout)
```

* Parameters

- timeout: timeout in searching printer (millisecond)

* Example

```
public class MainActivity extends Activity {

    ...

    private T_3II t_3ii;

    ...

    public void onCreate(Bundle savedInstanceState) {

        ...

        t_3ii = new T_3II(this, mHandler, null);
        t_3ii.findNetworkPrinters(5000);

        ...

    }

    private final Handler mHandler = new Handler(new Handler.Callback() {

        public boolean handleMessage(Message msg) {
            switch (msg.what) {
                case T_3II.MESSAGE_NETWORK_DEVICE_SET:
                    if (msg.obj != null) {
                        Set<String> ipAddressSet = (Set<String>) msg.obj;
                        for (String ipAddress : ipAddressSet) {
                            if (ipAddress.equals("192.168.0.100")) {
                                // TODO: Connect printer
                                break;
                            }
                        }
                    }
                    break;
            }
            return true;
        }
    });
}
```

7-2-2 findUsbPrinters

This method obtains the information of the printer connected by USB with the Android device that runs the application, and it sends the MESSAGE_USB_DEVICE_SET message to the application handler. The message includes the information of the printer connected by USB. The return value will be null if no USB printer is found.

* Syntax

```
public void findUsbPrinters()
```

* Example

```
public class MainActivity extends Activity {

    ...

    private T_3II t_3ii;

    ...

    public void onCreate(Bundle savedInstanceState) {

        ...

        t_3ii = new T_3II(this, mHandler, null);
        t_3ii.findUsbPrinters();

        ...

    }

    private final Handler mHandler = new Handler(new Handler.Callback() {

        public boolean handleMessage(Message msg) {
            switch (msg.what) {
                case T_3II.MESSAGE_USB_DEVICE_SET:
                    Set<UsbDevice> usbDeviceSet = (Set<UsbDevice>)
msg.obj;
                    for (UsbDevice device : usbDeviceSet) {
                        if (device.getProductId() == 0x2b) {
                            // TODO: Connect printer
                            break;
                        }
                    }
                    break;
            }
            return true;
        }
    });
}
```

7-3 Connect Printer

7-3-1 connect

This method opens the printer port of USB printer and enables communication. It sends the MESSAGE_STATE_CHANGE that includes STATE_CONNECTING in arg1 when connection process is initiated and MESSAGE_STATE_CHANGE with STATE_CONNECTED in arg1 when the connection is completed to the application handler. The messages are transmitted in the following order when this method is called.

* If connection is successful

1. MESSAGE_STATE_CHANGE (arg1: STATE_CONNECTING): Connection is being established.
2. MESSAGE_DEVICE_NAME: Connection to printer port is successful (USB device name of the printer recognized in Android device is transmitted.)
3. MESSAGE_STATE_CHANGE (arg1: STATE_CONNECTED): Communication with the printer is enabled.

* If connection fails

1. MESSAGE_TOAST: "Unable to connect device" message is sent.
2. MESSAGE_STATE_CHANGE (arg1: STATE_NONE): Printer is not connected.

* Syntax

```
public void connect()
```

* Example

```
public class MainActivity extends Activity {

    ...

    private T_3II t_3ii;

    ...

    public void onCreate(Bundle savedInstanceState) {

        ...

        t_3ii = new T_3II(this, mHandler, null);
        t_3ii.connect();

        ...

    }

    private final Handler mHandler = new Handler(new Handler.Callback() {

        public boolean handleMessage(Message msg) {
            switch (msg.what) {
                case T_3II.MESSAGE_STATE_CHANGE:
                    switch (msg.arg1) {
                        case T_3II.STATE_CONNECTING:
                            // TODO: Processing when connection to printer is being tried
                            break;

                        case T_3II.STATE_CONNECTED:
                            // TODO: Processing when printer connection is completed
                            break;

                        case T_3II.STATE_NONE:
                            // TODO: Processing when printer is not connected
                            break;
                    }
                    break;

                case T_3II.MESSAGE_DEVICE_NAME:
                    String connectedDeviceName =
                        msg.getData().getString(
                            T_3II.KEY_STRING_DEVICE_NAME);

                    break;

                case T_3II.MESSAGE_TOAST:
                    Toast.makeText(getApplicationContext(),
                        msg.getData().getString(T_3II.KEY_STRING_TOAST),
                        Toast.LENGTH_SHORT).show();

                    break;
            }
            return true;
        }
    });
}
```

7-3-2 connect

This method opens the port of printer connected with LAN or Wireless LAN and enables communication. It sends the MESSAGE_STATE_CHANGE message with STATE_CONNECTING in arg1 when connection process is initiated and MESSAGE_STATE_CHANGE with STATE_CONNECTED in arg1 when connection is completed to the application handler.

Messages are transmitted in the following order when this method is called.

* If connection is successful

1. MESSAGE_STATE_CHANGE (arg1: STATE_CONNECTING): Connection is being established
2. MESSAGE_DEVICE_NAME: Connection to printer port is successful. (Device name of the printer recognized by Android device is transmitted.)
3. MESSAGE_STATE_CHANGE (arg1: STATE_CONNECTED): Communication with printer is enabled.

* If connection fails

1. MESSAGE_TOAST: "Unable to connect device" message is sent.
2. MESSAGE_STATE_CHANGE (arg1: STATE_NONE): Printer is not connected.

* Syntax

```
public void connect(String host, int port, int timeout)
```

* Parameters

- host: IP address of the printer to connect
- port: Port number of the printer to connect (only 9100 is allowed.)
- timeout: Timeout in connecting printer (millisecond)

* Example

```
public class MainActivity extends Activity {

    ...

    private T_3II t_3ii;

    ...

    public void onCreate(Bundle savedInstanceState) {

        ...

        t_3ii = new T_3II(this, mHandler, null);
        t_3ii.connect("192.168.0.100", 9100, 5000);

        ...

    }

    private final Handler mHandler = new Handler(new Handler.Callback() {

        public boolean handleMessage(Message msg) {
            switch (msg.what) {
                case T_3II.MESSAGE_STATE_CHANGE:
                    switch (msg.arg1) {
                        case T_3II.STATE_CONNECTING:
                            // TODO: Processing when connection to printer is being
                            // tried
                            break;

                        case T_3II.STATE_CONNECTED:
                            // TODO: Processing when printer connection is completed
                            break;

                        case T_3II.STATE_NONE:
                            // TODO: Processing when printer is not connected
                            break;
                    }
                    break;

                case T_3II.MESSAGE_DEVICE_NAME:
                    String connectedDeviceName =
                        msg.getData().getString(
                            T_3II.KEY_STRING_DEVICE_NAME);

                    break;

                case T_3II.MESSAGE_TOAST:
                    Toast.makeText(getApplicationContext(),
                        msg.getData().getString(T_3II.KEY_STRING_TOAST),
                        Toast.LENGTH_SHORT).show();

                    break;
            }
            return true;
        }
    });
}
```


7-3-3 connect

This method opens the port of printer connected over USB and enables communication. It sends the MESSAGE_STATE_CHANGE message with STATE_CONNECTING in arg1 when connection process is initiated and MESSAGE_STATE_CHANGE with STATE_CONNECTED in arg1 when connection is completed to the application handler. Messages are transmitted in the following order when this method is called.

* If connection is successful

1. MESSAGE_STATE_CHANGE (arg1: STATE_CONNECTING): Connection is being established
2. MESSAGE_DEVICE_NAME: Connection to printer port is successful. (Device name of the printer recognized by Android device is transmitted.)
3. MESSAGE_STATE_CHANGE (arg1: STATE_CONNECTED): Communication with printer is enabled.

* If connection fails

1. MESSAGE_TOAST: "Unable to connect device" message is sent.
2. MESSAGE_STATE_CHANGE (arg1: STATE_NONE): Printer is not connected.

* Syntax

```
public void connect(UsbDevice device)
```

* Parameters

- device: UsbDevice instance of the device to connect. It can be received through the message received as a response to findUsbPrinters().

* Example

```
public class MainActivity extends Activity {

    ...

    private T_3II t_3ii;

    ...

    public void onCreate(Bundle savedInstanceState) {

        ...

        t_3ii = new T_3II(this, mHandler, null);
        t_3ii.findUsbPrinters();

        ...

    }

    private final Handler mHandler = new Handler(new Handler.Callback() {

        public boolean handleMessage(Message msg) {
            switch (msg.what) {
                case T_3II.MESSAGE_USB_DEVICE_SET:
                    Set<UsbDevice> usbDeviceSet = (Set<UsbDevice>) msg.obj;
                    for (UsbDevice device : usbDeviceSet) {
                        if (device.getProductId() == 0x2b) {
                            t_3ii.connect(device);
                            break;
                        }
                    }
                    return true;

                case T_3II.MESSAGE_STATE_CHANGE:
                    switch (msg.arg1) {
                        case T_3II.STATE_CONNECTING:
                            // TODO: Processing when connection to printer is being tried
                            break;

                        case T_3II.STATE_CONNECTED:
                            // TODO: Processing when printer connection is completed
                            break;

                        case T_3II.STATE_NONE:
                            // TODO: Processing when printer is not connected
                            break;
                    }
                    break;

                case T_3II.MESSAGE_DEVICE_NAME:
                    String connectedDeviceName =
                        msg.getData().getString(
                            T_3II.KEY_STRING_DEVICE_NAME);

                    break;

                case T_3II.MESSAGE_TOAST:
                    Toast.makeText(getApplicationContext(),
                        msg.getData().getString(T_3II.KEY_STRING_TOAST),
                        Toast.LENGTH_SHORT).show();

                    break;
            }
            return true;
        }
    });
}
```

7-3-4 disconnect

The method closes the port of connected printer and terminates the connection. When connection is terminated, it sends the MESSAGE_STATE_CHANGE message (arg1: STATE_NONE) to application handler.

* Syntax

```
public void disconnect()
```

* Example

```
public class MainActivity extends Activity {

    ...

    private T_3II t_3ii;

    ...

    public void onCreate(Bundle savedInstanceState) {

        ...

        t_3ii = new T_3II(this, mHandler, null);
        t_3ii.connect(null);

        ...

    }

    private final Handler mHandler = new Handler(new Handler.Callback() {

        public boolean handleMessage(Message msg) {
            switch (msg.what) {
                case T_3II.MESSAGE_STATE_CHANGE:
                    switch (msg.arg1) {
                        case T_3II.STATE_CONNECTING:
                            // TODO: Processing when connection to printer is being tried
                            break;

                        case T_3II.STATE_CONNECTED:
                            t_3ii.disconnect();
                            break;

                        case T_3II.STATE_NONE:
                            Toast.makeText(getApplicationContext(), "Printer is disconnected",
                                Toast.LENGTH_SHORT).show();
                            break;
                    }
                    break;
            }
            return true;
        }
    });
}
```

7-4 Print

7-4-1 formFeed

This method performs paper feeding in the page mode or label mode.

* Syntax

```
public void formFeed(boolean getResponse)
```

* Parameters

- `getResponse`: A message is sent to the application handler upon completion of feeding if this parameter is set to `True`, and message is not sent if it is set to `False`.

* Example

```
public class MainActivity extends Activity {  
  
    ...  
  
    private T_3II t_3ii;  
  
    ...  
  
    public void onCreate(Bundle savedInstanceState) {  
  
        ...  
  
        t_3ii = new T_3II(this, mHandler, null);  
        t_3ii.connect(null);  
  
        ...  
  
    }  
  
    private final Handler mHandler = new Handler(new Handler.Callback() {  
  
        public boolean handleMessage(Message msg) {  
            switch (msg.what) {  
                case T_3II.MESSAGE_STATE_CHANGE:  
                    if (msg.arg1 == STATE_CONNECTED) {  
                        // TODO: Printing processing  
                        t_3ii.formFeed(true);  
                    }  
                    break;  
  
                case MESSAGE_PRINT_COMPLETE:  
                    t_3ii.disconnect();  
                    break;  
            }  
            return true;  
        }  
    });  
}
```

7-4-2 lineFeed

This method feeds the paper by the specified number of lines.

*** Syntax**

```
public void lineFeed(int lines, boolean getResponse)
```

*** Parameters**

- lines: number of lines to feed
- getResponse: A message is sent to the application handler upon completion of feeding if this parameter is set to True, and message is not sent if it is set to False.

* Example

```
public class MainActivity extends Activity {

    ...

    private T_3II t_3ii;

    ...

    public void onCreate(Bundle savedInstanceState) {

        ...

        t_3ii = new T_3II(this, mHandler, null);
        t_3ii.connect(null);

        ...

    }

    private final Handler mHandler = new Handler(new Handler.Callback() {

        public boolean handleMessage(Message msg) {
            switch (msg.what) {
                case MESSAGE_STATE_CHANGE:
                    if (msg.arg1 == STATE_CONNECTED) {
                        t_3ii.lineFeed(5, true);
                    }
                    break;

                case MESSAGE_PRINT_COMPLETE:
                    t_3ii.disconnect();
                    break;
            }
            return true;
        }
    });
}
```

7-4-3 print1dBarcode

This method prints one dimensional barcode.

* Syntax

```
public void print1dBarcode(  
    String data, int barCodeSystem, int alignment, int width, int height,  
    int characterPosition, boolean getResponse)
```

* Parameters

- data: barcode data to print
- barCodeSystem: barcode system

Code	Value	Description
BAR_CODE_UPC_A	65	UPC-A
BAR_CODE_UPC_E	66	UPC-E
BAR_CODE_EAN13	67	EAN13
BAR_CODE_EAN8	68	EAN8
BAR_CODE_CODE39	69	CODE93
BAR_CODE_ITF	70	ITF
BAR_CODE_CODABAR	71	CODABAR
BAR_CODE_CODE93	72	CODE93
BAR_CODE_CODE128	73	CODE128

- alignment: barcode alignment

Code	Value	Description
ALIGNMENT_LEFT	0	Align to left
ALIGNMENT_CENTER	1	Align to center
ALIGNMENT_RIGHT	2	Align to right

- width: width of barcode (1 ~ 6)
- height: height of barcode (1 ~ 255)
- characterPosition: position to print barcode data string

Code	Value	Description
HRI_CHARACTER_NOT_PRINTED	0	Character string is not printed
HRI_CHARACTER_ABOVE_BAR_CODE	1	Character string is printed above barcode
HRI_CHARACTER_BELOW_BAR_CODE	2	Character string is printed below barcode
HRI_CHARACTER_ABOVE_AND_BELOW_BAR_CODE	3	Character string is printed above and below barcode

Metapace T-3II SDK for Android

- `getResponse`: A message is sent to the application handler upon completion of printing if this parameter is set to `True`, and message is not sent if it is set to `False`.

* Example

```
public class MainActivity extends Activity {

    ...

    private T_3II t_3ii;

    ...

    public void onCreate(Bundle savedInstanceState) {

        ...

        t_3ii = new T_3II(this, mHandler, null);
        t_3ii.connect(null);

        ...

    }

    private final Handler mHandler = new Handler(new Handler.Callback() {

        public boolean handleMessage(Message msg) {
            switch (msg.what) {
                case T_3II.MESSAGE_STATE_CHANGE:
                    if (msg.arg1 == T_3II.STATE_CONNECTED) {
                        t_3ii.print1dBarcode("012345678905",
                            T_3II.BAR_CODE_UPC_A,
                            T_3II.ALIGNMENT_LEFT,3, 162,
                            T_3II.HRI_CHARACTER_ABOVE_BAR_CODE, true);
                    }
                    break;

                    case T_3II.MESSAGE_PRINT_COMPLETE:
                        t_3ii.disconnect();
                        break;
            }
            return true;
        }
    });
}
```


7-4-4 printBitmap

This method converts Bitmap instance to black and white image and prints the image.

* Syntax

```
public void printBitmap(  
    Bitmap bitmap, int alignment, int width, int level, boolean getResponse)
```

* Parameter

- bitmap: Bitmap instance to print
- alignment: Image alignment

Code	Value	Description
ALIGNMENT_LEFT	0	Align to left
ALIGNMENT_CENTER	1	Align to center
ALIGNMENT_RIGHT	2	Align to right

- width: width of image to print

Code	Value	Description
BITMAP_WIDTH_FULL	-1	Image is enlarged or reduced to the maximum printing width and printed.
BITMAP_WIDTH_NONE	0	Image is printed without resizing or reduced to the maximum printing width if the image is wider than the maximum width.
Integer		Enter integer number directly

- level: brightness level of the image to print (13 ~ 88)
- getResponse: A message is sent to the application handler upon completion of printing if this parameter is set to True, and message is not sent if it is set to False.

* Example

```
public class MainActivity extends Activity {

    ...

    private T_3II t_3ii;

    ...

    public void onCreate(Bundle savedInstanceState) {

        ...

        t_3ii = new T_3II(this, mHandler, null);
        t_3ii.connect(null);

        ...

    }

    private final Handler mHandler = new Handler(new Handler.Callback() {

        public boolean handleMessage(Message msg) {
            switch (msg.what) {
                case T_3II.MESSAGE_STATE_CHANGE:
                    if (msg.arg1 == T_3II.STATE_CONNECTED) {
                        BitmapDrawable drawable =
                            (BitmapDrawable)
getResources().getDrawable(R.drawable.logo);
                        Bitmap bitmap = drawable.getBitmap();
                        t_3ii.printBitmap(bitmap, T_3II.ALIGNMENT_LEFT,
                            T_3II.BITMAP_WIDTH_FULL,    50,
true);
                    }
                    break;

                case T_3II.MESSAGE_PRINT_COMPLETE:
                    t_3ii.disconnect();
                    break;
            }
            return true;
        }
    });
}
```

7-4-5 printBitmap

This method converts the image file located in the specified path to black and white image and prints the image.

* Syntax

```
public void printBitmap(  
    String pathName, int alignment, int width, int level, boolean getResponse)
```

* Parameter

- pathName: absolute path of the image file to print
- alignment: Image alignment

Code	Value	Description
ALIGNMENT_LEFT	0	Align to left
ALIGNMENT_CENTER	1	Align to center
ALIGNMENT_RIGHT	2	Align to right

- width: width of image to print

Code	Value	Description
BITMAP_WIDTH_FULL	-1	Image is enlarged or reduced to the maximum printing width and printed.
BITMAP_WIDTH_NONE	0	Image is printed without resizing or reduced to the maximum printing width if the image is wider than the maximum width.
Integer		Enter integer number directly

- level: brightness level of the image to print (13 ~ 88)
- getResponse: A message is sent to the application handler upon completion of printing if this parameter is set to True, and message is not sent if it is set to False.

* Example

```
public class MainActivity extends Activity {

    ...

    private T_3II t_3ii;

    ...

    public void onCreate(Bundle savedInstanceState) {

        ...

        t_3ii = new T_3II(this, mHandler, null);
        t_3ii.connect(null);

        ...

    }

    private final Handler mHandler = new Handler(new Handler.Callback() {

        public boolean handleMessage(Message msg) {
            switch (msg.what) {
                case T_3II.MESSAGE_STATE_CHANGE:
                    if (msg.arg1 == T_3II.STATE_CONNECTED) {
                        String pathName =
Environment.getExternalStorageDirectory().getAbsolutePath() +
                                "/logo.png";
                        t_3ii.printBitmap(pathName,
                                T_3II.ALIGNMENT_LEFT,
                                T_3II.BITMAP_WIDTH_FULL,    50,
true);
                                }
                                break;

                case T_3II.MESSAGE_PRINT_COMPLETE:
                    t_3ii.disconnect();
                    break;
            }
            return true;
        }
    });
}
```

7-4-6 printPdf417

This method prints PDF417.

* Syntax

```
public void printPdf417(  
    String data, int alignment, int width, int height, boolean getResponse)
```

* Parameter

- data: barcode data to print
- alignment: barcode printing alignment

Code	Value	Description
ALIGNMENT_LEFT	0	Align to left
ALIGNMENT_CENTER	1	Align to center
ALIGNMENT_RIGHT	2	Align to right

- width: width of barcode to print (2 ~ 3)
- height: height of barcode to print (2 ~ 8)
- getResponse: A message is sent to the application handler upon completion of printing if this parameter is set to True, and message is not sent if it is set to False.

* Example

```
public class MainActivity extends Activity {

    ...

    private T_3II t_3ii;

    ...

    public void onCreate(Bundle savedInstanceState) {

        ...

        t_3ii = new T_3II(this, mHandler, null);
        t_3ii.connect(null);

        ...

    }

    private final Handler mHandler = new Handler(new Handler.Callback() {

        public boolean handleMessage(Message msg) {
            switch (msg.what) {
                case T_3II.MESSAGE_STATE_CHANGE:
                    if (msg.arg1 == T_3II.STATE_CONNECTED) {
                        t_3ii.printPdf417("www.metapace.com",
                            T_3II.ALIGNMENT_LEFT, 3, 8, true);
                    }
                    break;

                case T_3II.MESSAGE_PRINT_COMPLETE:
                    t_3ii.disconnect();
                    break;
            }
            return true;
        }
    });
}
```

7-4-7 printQRCode

This method prints QR Code.

* Syntax

```
public void printQRCode(  
    String data, int alignment, int model, int size, boolean getResponse)
```

* Parameter

- data: barcode data to print
- alignment: barcode printing alignment

Code	Value	Description
ALIGNMENT_LEFT	0	Align to left
ALIGNMENT_CENTER	1	Align to center
ALIGNMENT_RIGHT	2	Align to right

- model: QR Code model to print

Code	Value	Description
QR_CODE_MODEL1	48	QR Code model 1
QR_CODE_MODEL2	49	QR Code model 2

- size: size of barcode to print (1 ~ 8)
- getResponse: A message is sent to the application handler upon completion of printing if this parameter is set to True, and message is not sent if it is set to False.

* Example

```
public class MainActivity extends Activity {

    ...

    private T_3II t_3ii;

    ...

    public void onCreate(Bundle savedInstanceState) {

        ...

        t_3ii = new T_3II(this, mHandler, null);
        t_3ii.connect(null);

        ...

    }

    private final Handler mHandler = new Handler(new Handler.Callback() {

        public boolean handleMessage(Message msg) {
            switch (msg.what) {
                case T_3II.MESSAGE_STATE_CHANGE:
                    if (msg.arg1 == T_3II.STATE_CONNECTED) {
                        t_3ii.printQRCode("www.metapace.com",
                            T_3II.ALIGNMENT_LEFT,
                            T_3II.QR_CODE_MODEL2, 8, true);
                    }
                    break;

                case T_3II.MESSAGE_PRINT_COMPLETE:
                    t_3ii.disconnect();
                    break;
            }
            return true;
        }
    });
}
```


7-4-8 printQRCode

This method prints QR Code.

* Syntax

```
public void printQRCode(  
    String data, int alignment, int model, int size, int errorCorrectionLevel,  
    boolean getResponse)
```

* Parameter

- data: barcode data to print
- alignment: barcode printing alignment

Code	Value	Description
ALIGNMENT_LEFT	0	Align to left
ALIGNMENT_CENTER	1	Align to center
ALIGNMENT_RIGHT	2	Align to right

- model: QR Code model to print

Code	Value	Description
QR_CODE_MODEL1	48	QR Code model 1
QR_CODE_MODEL2	49	QR Code model 2

- size: size of barcode to print (1 ~ 8)
- errorCorrectionLevel: error correction level of barcode to print

Code	Value	Description
QR_CODE_ERROR_CORRECTION_LEVEL_L	48	Error correction level L
QR_CODE_ERROR_CORRECTION_LEVEL_M	49	Error correction level M
QR_CODE_ERROR_CORRECTION_LEVEL_Q	50	Error correction level Q
QR_CODE_ERROR_CORRECTION_LEVEL_H	51	Error correction level H

- getResponse: A message is sent to the application handler upon completion of printing if this parameter is set to True, and message is not sent if it is set to False.

* Example

```
public class MainActivity extends Activity {

    ...

    private T_3II t_3ii;

    ...

    public void onCreate(Bundle savedInstanceState) {

        ...

        t_3ii = new T_3II(this, mHandler, null);
        t_3ii.connect(null);

        ...

    }

    private final Handler mHandler = new Handler(new Handler.Callback() {

        public boolean handleMessage(Message msg) {
            switch (msg.what) {
                case T_3II.MESSAGE_STATE_CHANGE:
                    if (msg.arg1 == T_3II.STATE_CONNECTED) {
                        t_3ii.printQRCode("www.metapace.com",
                            T_3II.ALIGNMENT_LEFT,
                            T_3II.QR_CODE_MODEL2, 8,
                            T_3II.QR_CODE_ERROR_CORRECTION_LEVEL_L,
                            true);
                    }
                    break;

                case T_3II.MESSAGE_PRINT_COMPLETE:
                    t_3ii.disconnect();
                    break;
            }
            return true;
        }
    });
}
```

7-4-9 printSelfTest

This page prints Self-Test page. Printer settings and current code page are printed.

* Syntax

```
public void printSelfTest(boolean getResponse)
```

* Example

```
public class MainActivity extends Activity {  
  
    ...  
  
    private T_3II t_3ii;  
  
    ...  
  
    public void onCreate(Bundle savedInstanceState) {  
  
        ...  
  
        t_3ii = new T_3II(this, mHandler, null);  
        t_3ii.connect(null);  
  
        ...  
  
    }  
  
    private final Handler mHandler = new Handler(new Handler.Callback() {  
  
        public boolean handleMessage(Message msg) {  
            switch (msg.what) {  
                case T_3II.MESSAGE_STATE_CHANGE:  
                    if (msg.arg1 == T_3II.STATE_CONNECTED) {  
                        t_3ii.printSelfTest(true);  
                    }  
                    break;  
  
                case T_3II.MESSAGE_PRINT_COMPLETE:  
                    t_3ii.disconnect();  
                    break;  
            }  
            return true;  
        }  
    });  
}
```

7-4-10 printText

This method prints character string.

*** Syntax**

```
public void printText(String text, int alignment, int attribute, int size, boolean getResponse)
```

*** Parameters**

- text: character string to print
- alignment: character string alignment

Code	Value	Description
ALIGNMENT_LEFT	0	Align to left
ALIGNMENT_CENTER	1	Align to center
ALIGNMENT_RIGHT	2	Align to right

- attribute: attributes of the character string. When more than one parameter is configured, each option should be combined through bitwise OR operation.

Code	Value	Description
TEXT_ATTRIBUTE_FONT_A	0	font type A (12X24)
TEXT_ATTRIBUTE_FONT_B	1	font type B (9X17)
TEXT_ATTRIBUTE_FONT_C	2	font type C (9X24)
TEXT_ATTRIBUTE_UNDERLINE1	4	Underline with 1 dot thickness
TEXT_ATTRIBUTE_UNDERLINE2	8	Underline with 2 dots thickness
TEXT_ATTRIBUTE_EMPHASIZED	16	Bold
TEXT_ATTRIBUTE_REVERSE	32	Reversed

- size: size of character string to print. The width and height parameters should be combined through bitwise OR operation.

Code	Value	Description
TEXT_SIZE_HORIZONTAL1	0	1X magnification on width
TEXT_SIZE_HORIZONTAL2	16	2X magnification on width
TEXT_SIZE_HORIZONTAL3	32	3X magnification on width
TEXT_SIZE_HORIZONTAL4	48	4X magnification on width
TEXT_SIZE_HORIZONTAL5	64	5X magnification on width
TEXT_SIZE_HORIZONTAL6	80	6X magnification on width
TEXT_SIZE_HORIZONTAL7	96	7X magnification on width
TEXT_SIZE_HORIZONTAL8	112	8X magnification on width
TEXT_SIZE_VERTICAL1	0	1X magnification on height
TEXT_SIZE_VERTICAL2	1	2X magnification on height
TEXT_SIZE_VERTICAL3	2	3X magnification on height
TEXT_SIZE_VERTICAL4	3	4X magnification on height
TEXT_SIZE_VERTICAL5	4	5X magnification on height
TEXT_SIZE_VERTICAL6	5	6X magnification on height
TEXT_SIZE_VERTICAL7	6	7X magnification on height
TEXT_SIZE_VERTICAL8	7	8X magnification on height

Metapace T-3II SDK for Android

- `getResponse`: A message is sent to the application handler upon completion of printing if this parameter is set to `True`, and message is not sent if it is set to `False`.

* Example

```
public class MainActivity extends Activity {

    ...

    private T_3II t_3ii;

    ...

    public void onCreate(Bundle savedInstanceState) {

        ...

        t_3ii = new T_3II(this, mHandler, null);
        t_3ii.connect(null);

        ...

    }

    private final Handler mHandler = new Handler(new Handler.Callback() {

        public boolean handleMessage(Message msg) {
            switch (msg.what) {
                case T_3II.MESSAGE_STATE_CHANGE:
                    if (msg.arg1 == T_3II.STATE_CONNECTED) {
                        t_3ii.printText("printText\n",
                                T_3II.ALIGNMENT_LEFT,
                                T_3II.TEXT_ATTRIBUTE_FONT_A |
                                T_3II.TEXT_ATTRIBUTE_UNDERLINE1,
                                T_3II.TEXT_SIZE_HORIZONTAL1 |
                                T_3II.TEXT_SIZE_VERTICAL1, true);
                    }
                    break;

                case T_3II.MESSAGE_PRINT_COMPLETE:
                    t_3ii.disconnect();
                    break;
            }
            return true;
        }

    });
}
```

7-5 Receive Printer Response

7-5-1 automateStatusBack

This method enables or disables automatic status check function of printer. When it is activated, the MESSAGE_READ message (arg1: PROCESS_AUTO_STATUS_BACK) is sent to the application handler whenever there is any change in the printer status. The arg2 includes the following values if there is any error in printer.

Code	Value	Description
AUTO_STATUS_COVER_OPEN	32	Printer cover is open.
AUTO_STATUS_NO_PAPER	12	No printer paper

* Syntax

```
public void automateStatusBack(boolean isEnable)
```

* Example

```
public class MainActivity extends Activity {

    ...

    private T_3II t_3ii;

    ...

    public void onCreate(Bundle savedInstanceState) {

        ...

        t_3ii = new T_3II(this, mHandler, null);
        t_3ii.connect();

        ...

    }

    public void onDestroy() {

        ...

        if (t_3ii != null) {
            t_3ii.automateStatusBack(false);
            t_3ii.disconnect();
        }

        ...

    }

    private final Handler mHandler = new Handler(new Handler.Callback() {

        public boolean handleMessage(Message msg) {
            switch (msg.what) {
                case T_3II.MESSAGE_STATE_CHANGE:
                    if (msg.arg1 == T_3II.STATE_CONNECTED) {
                        t_3ii.automateStatusBack(true);
                    }
                    break;

                case T_3II.MESSAGE_READ:
                    StringBuffer buffer = new StringBuffer(0);
                    if (msg.arg1 == T_3II.PROCESS_AUTO_STATUS_BACK) {
                        if ((msg.arg2 & T_3II.AUTO_STATUS_COVER_OPEN) ==
                            T_3II.AUTO_STATUS_COVER_OPEN) {
                            buffer.append("Cover is open.\n");
                        }
                        if ((msg.arg2 & T_3II.AUTO_STATUS_NO_PAPER) ==
                            T_3II.AUTO_STATUS_NO_PAPER) {
                            buffer.append("Paper end sensor: no paper present.\n");
                        }
                    }
                    if (buffer.capacity() > 0) {
                        Toast.makeText(getApplicationContext(), buffer.toString(),
                                    Toast.LENGTH_SHORT).show();
                    } else {
                        Toast.makeText(getApplicationContext(), "No error.",
                                    Toast.LENGTH_SHORT).show();
                    }
                    break;
            }
            return true;
        }
    });
}
```

7-5-2 getPrinterId

This method checks the printer information. When the information is available, MESSAG_READ message (arg1: PROCESS_GET_PRINTER_ID) is sent to the application handler.

* Syntax

```
public void getPrinterId(int idType)
```

* Parameters

- idType: Information of the printer to check

Code	Value	Description
PRINTER_ID_FIRMWARE_VERSION	65	Firmware version
PRINTER_ID_MANUFACTURER	66	Manufacturer (Metapace Identsystem GmbH)
PRINTER_ID_PRINTER_MODEL	67	Printer model name
PRINTER_ID_CODE_PAGE	69	Current code page

* Example

```
public class MainActivity extends Activity {

    ...

    private T_3II t_3ii;

    ...

    public void onCreate(Bundle savedInstanceState) {

        ...

        t_3ii = new T_3II(this, mHandler, null);
        t_3ii.connect(null);

        ...

    }

    public void onDestroy() {

        ...

        if (t_3ii != null) {
            t_3ii.disconnect();
        }

        ...

    }

    private final Handler mHandler = new Handler(new Handler.Callback() {

        public boolean handleMessage(Message msg) {
            switch (msg.what) {
                case T_3II.MESSAGE_STATE_CHANGE:
                    if (msg.arg1 == T_3II.STATE_CONNECTED) {
                        t_3ii.getPrinterId(

T_3II.PRINTER_ID_FIRMWARE_VERSION);
                    }
                    break;

                case T_3II.MESSAGE_READ:
                    if (msg.arg1 == T_3II.PROCESS_GET_PRINTER_ID) {
                        Bundle data = msg.getData();
                        Toast.makeText(getApplicationContext(),

data.getString(T_3II.KEY_STRING_PRINTER_ID),
                                Toast.LENGTH_SHORT).show();
                    }
                    break;
            }
            return true;
        }
    });
}
```

7-5-3 getStatus

This method gets the status of printer. When the status is obtained, the MESSAGE_READ message (arg1: PROCESS_GET_STATUS) is sent to the application handler.

The following values can be returned in arg2.

Code	Value	Description
STATUS_NORMAL	0	Normal
STATUS_COEVER_OPEN	4	Cover is open
STATUS_PAPER_NEAR_END	12	Printer paper reaches to near end state
STATUS_PAPER_NOT_PRESENT	96	No printer paper

* Syntax

```
public void getStatus()
```

* Example

```
public class MainActivity extends Activity {

    ...

    private T_3II t_3ii;

    ...

    public void onCreate(Bundle savedInstanceState) {

        ...

        t_3ii = new T_3II(this, mHandler, null);
        t_3ii.connect();

        ...

    }

    public void onDestroy() {

        ...

        if (t_3ii != null) {
            t_3ii.disconnect();
        }

        ...

    }

    private final Handler mHandler = new Handler(new Handler.Callback() {

        public boolean handleMessage(Message msg) {
            switch (msg.what) {
                case T_3II.MESSAGE_STATE_CHANGE:
                    if (msg.arg1 == T_3II.STATE_CONNECTED) {
                        t_3ii.getStatus();
                    }
                    break;

                case T_3II.MESSAGE_READ:
                    if (msg.arg1 == T_3II.PROCESS_GET_STATUS) {
                        if (msg.arg2 == T_3II.STATUS_NORMAL) {
                            Toast.makeText(getApplicationContext(), "No error",
                                Toast.LENGTH_SHORT).show();
                        } else {
                            StringBuffer buffer = new StringBuffer();
                            if ((msg.arg2 & T_3II.STATUS_COVER_OPEN) ==
                                T_3II.STATUS_COVER_OPEN) {
                                buffer.append("Cover is open.\n");
                            }
                            if ((msg.arg2 & T_3II.STATUS_PAPER_NOT_PRESENT) ==
                                T_3II.STATUS_PAPER_NOT_PRESENT) {
                                buffer.append("Paper end sensor: paper not present.\n");
                            }
                            Toast.makeText(getApplicationContext(), buffer.toString(),
                                Toast.LENGTH_SHORT).show();
                        }
                    }
                    break;
            }
            return true;
        }
    });
}
```

7-6 NV Image

7-6-1 defineNvlImage

This method saves the image in the non-volatile memory area of printer. When image is saved, the MESSAGE_WRITE message (arg1: PROCESS_DEFINE_NV_IMAGE) is sent to the application handler.

* Syntax

```
public void defineNvlImage(Bitmap bitmap, int width, int level, int keyCode)
```

* Paramters

- bitmap: Bitmap instance to save
- width: width of image to save

Code	Value	Description
BITMAP_WIDTH_FULL	-1	Image is enlarged or reduced to the maximum printing width and saved.
BITMAP_WIDTH_NONE	0	Image is saved without resizing or reduced to the maximum printing width if the image is wider than the maximum width.
Integer		Enter integer number directly

- level: brightness level of image to save (13 ~ 88)
- keyCode: address of memory area to save image. (0 ~ 255)

* Example

```
public class MainActivity extends Activity {

    ...

    private T_3II t_3ii;

    ...

    public void onCreate(Bundle savedInstanceState) {

        ...

        t_3ii = new T_3II(this, mHandler, null);
        t_3ii.connect(null);

        ...

    }

    private final Handler mHandler = new Handler(new Handler.Callback() {

        public boolean handleMessage(Message msg) {
            switch (msg.what) {
                case T_3II.MESSAGE_STATE_CHANGE:
                    if (msg.arg1 == T_3II.STATE_CONNECTED) {
                        BitmapDrawable drawable =
                            (BitmapDrawable)
getResources().getDrawable(R.drawable.logo);
                        Bitmap bitmap = drawable.getBitmap();
                        t_3ii.defineNvImage(bitmap,
                            T_3II.BITMAP_WIDTH_FULL, 50, 0);
                    }
                    break;

                case T_3II.MESSAGE_WRITE:
                    if (msg.arg1 == T_3II.PROCESS_DEFINE_NV_IMAGE) {
                        t_3ii.disconnect();
                    }
                    break;
            }
            return true;
        }
    });
}
```

7-6-2 defineNvlImage

This method saves the image in the non-volatile memory area of printer. When image save operation is completed, the MESSAGE_WRITE message (arg1: PROCESS_DEFINE_NV_IMAGE) is sent to the application handler.

* Syntax

```
public void defineNvlImage(String pathName, int width, int level, int keyCode)
```

* Paramters

- pathName: absolute path of the image file to save in printer
- width: width of image to save

Code	Value	Description
BITMAP_WIDTH_FULL	-1	Image is enlarged or reduced to the maximum printing width and saved.
BITMAP_WIDTH_NONE	0	Image is saved without resizing or reduced to the maximum printing width if the image is wider than the maximum width.
Integer		Enter integer number directly

- level: brightness level of image to save (13 ~ 88)
- keyCode: address of memory area to save image (0 ~ 255)

* Example

```
public class MainActivity extends Activity {

    ...

    private T_3II t_3ii;

    ...

    public void onCreate(Bundle savedInstanceState) {

        ...

        t_3ii = new T_3II(this, mHandler, null);
        t_3ii.connect(null);

        ...

    }

    private final Handler mHandler = new Handler(new Handler.Callback() {

        public boolean handleMessage(Message msg) {
            switch (msg.what) {
                case T_3II.MESSAGE_STATE_CHANGE:
                    if (msg.arg1 == T_3II.STATE_CONNECTED) {
                        String pathName =
Environment.getExternalStorageDirectory().getAbsolutePath() +
                                "/logo.png";
                        t_3ii.defineNvImage(pathName,
                                T_3II.BITMAP_WIDTH_FULL, 50, 0);
                    }
                    break;

                case MESSAGE_WRITE:
                    if (msg.arg1 == T_3II.PROCESS_DEFINE_NV_IMAGE) {
                        t_3ii.disconnect();
                    }
                    break;
            }
            return true;
        }
    });
}
```

7-6-3 getDefinedNvImageKeyCodes

This method obtains the address of image saved in the non-volatile memory area of printer. When address is obtained, the MESSAGE_READ message (arg1: PROCESS_GET_NV_IMAGE_KEY_CODES) is sent to the application handler. If there is any image stored in the area, the address list of the images is returned as integer array or null is returned if there is no image.

* Syntax

```
public void getDefinedNvImageKeyCodes()
```


* Example

```
public class MainActivity extends Activity {

    ...

    private T_3II t_3ii;

    ...

    public void onCreate(Bundle savedInstanceState) {

        ...

        t_3ii = new T_3II(this, mHandler, null);
        t_3ii.connect(null);

        ...

    }

    public void onDestroy() {

        ...

        if (t_3ii != null) {
            t_3ii.disconnect();
        }

        ...

    }

    private final Handler mHandler = new Handler(new Handler.Callback() {

        public boolean handleMessage(Message msg) {
            switch (msg.what) {
                case T_3II.MESSAGE_STATE_CHANGE:
                    if (msg.arg1 == STATE_CONNECTED) {
                        t_3ii.getDefinedNvImageKeyCodes();
                    }
                    break;

                case T_3II.MESSAGE_READ:
                    if (msg.arg1 ==
                        T_3II.PROCESS_GET_NV_IMAGE_KEY_CODES) {
                        Bundle data = msg.getData();
                        int[] keyCodes =
                            data.getIntArray(T_3II.NV_IMAGE_KEY_CODES);
                        if (keyCodes != null) {
                            // TODO: Image address processing
                        }
                    }
                    break;
            }
            return true;
        }
    });
}
```

7-6-4 printNvlImage

This method prints image stored in the non-volatile memory area of printer.

*** Syntax**

```
public void printNvlImage(int keyCode, boolean getResponse)
```

*** Parameters**

- keyCode: address code of NV image to print
- getResponse: MESSAGE_PRINT_COMPLETE A message is sent to the
- application handler upon completion of printing if this parameter is set to True, and message is not sent if it is set to False.

* Example

```
public class MainActivity extends Activity {

    ...

    private T_3II t_3ii;

    public void onCreate(Bundle savedInstanceState) {

        ...

        t_3ii = new T_3II(this, mHandler, null);
        t_3ii.connect(null);

        ...

    }

    public void onDestroy() {

        ...

        if (t_3ii != null) {
            t_3ii.disconnect();
        }

        ...

    }

    private final Handler mHandler = new Handler(new Handler.Callback() {

        public boolean handleMessage(Message msg) {
            switch (msg.what) {
                case T_3II.MESSAGE_STATE_CHANGE:
                    if (msg.arg1 == T_3II.STATE_CONNECTED) {
                        t_3ii.getDefinedNvlImageKeyCodes();
                    }
                    break;

                case T_3II.MESSAGE_READ:
                    if (msg.arg1 ==
                        T_3II.PROCESS_GET_NV_IMAGE_KEY_CODES) {
                        Bundle data = msg.getData();
                        int[] keyCodes =
                            data.getIntArray(T_3II.NV_IMAGE_KEY_CODES);
                        if (keyCodes != null) {
                            for (int i = 0; i < keyCodes.length; i++) {
                                t_3ii.printNvlImage(keyCodes[i], false);
                            }
                        }
                    }
                    break;
            }
            return true;
        }
    });
}
```

7-6-5 removeAllNvlImage

This method deletes all image data stored in the non-volatile memory area of printer.

* Syntax

```
public void printAllNvlImage()
```

* Example

```
public class MainActivity extends Activity {  
  
    ...  
  
    private T_3II t_3ii;  
  
    ...  
  
    public void onCreate(Bundle savedInstanceState) {  
  
        ...  
  
        t_3ii = new T_3II(this, mHandler, null);  
        t_3ii.connect(null);  
  
        ...  
    }  
  
    public void onDestroy() {  
  
        ...  
  
        if (t_3ii != null) {  
            t_3ii.disconnect();  
        }  
  
        ...  
    }  
  
    private final Handler mHandler = new Handler(new Handler.Callback() {  
  
        public boolean handleMessage(Message msg) {  
            switch (msg.what) {  
                case T_3II.MESSAGE_STATE_CHANGE:  
                    if (msg.arg1 == T_3II.STATE_CONNECTED) {  
                        t_3ii.removeAllNvlImage();  
                        break;  
                    }  
                    break;  
            }  
            return true;  
        }  
    });  
}
```

7-6-6 removeNvlImage

This function deletes image data stored in the non-volatile memory area of printer.

*** Syntax**

```
public void removeNvlImage(int keyCode)
```

*** Parameters**

- keyCode: address code of the NV image to delete

* Example

```
public class MainActivity extends Activity {

    ...

    private T_3II t_3ii;

    ...

    public void onCreate(Bundle savedInstanceState) {

        ...

        t_3ii = new T_3II(this, mHandler, null);
        t_3ii.connect(null);

        ...

    }

    public void onDestroy() {

        ...

        if (t_3ii != null) {
            t_3ii.disconnect();
        }

        ...

    }

    private final Handler mHandler = new Handler(new Handler.Callback() {

        public boolean handleMessage(Message msg) {
            switch (msg.what) {
                case T_3II.MESSAGE_STATE_CHANGE:
                    if (msg.arg1 == STATE_CONNECTED) {
                        t_3ii.getDefinedNvImageKeyCodes()
                    }
                    break;

                case T_3II.MESSAGE_READ:
                    if (msg.arg1 ==
                        T_3II.PROCESS_GET_NV_IMAGE_KEY_CODES) {
                        Bundle data = msg.getData();
                        int[] keyCodes =
                            data.getIntArray(T_3II.NV_IMAGE_KEY_CODES);
                        if (keyCodes != null) {
                            t_3ii.removeNvImage(keyCodes[0]);
                        }
                    }
                    break;
            }
            return true;
        }
    });
}
```

7-7 Page Mode

7-7-1 setAbsolutePrintPosition

This method sets the horizontal position in the page mode.

* Syntax

```
public void setAbsolutePrintPosition(int position)
```

* Parameters

- position: horizontal position to set

7-7-2 setAbsoluteVerticalPrintPosition

This method sets the vertical position in the page mode.

* Syntax

```
public void setAbsoluteVerticalPrintPosition(int position)
```

* Parameters

- position: vertical position to set

7-7-3 setPageMode

This function switches the mode to page mode.

* Syntax

```
public void setPageMode()
```

7-7-4 setPrintArea

This function sets the printing area in the page mode.

* Syntax

```
public void setPrintArea(int x, int y, int width, int height)
```

* Parametes

- x: origin of the horizontal printing area
- y: origin of the vertical printing area
- width: width of printing area
- height: height of printing area

7-7-5 setPrintDirection

This method sets the printing direction in the page mode.

* Syntax

```
public void setPrintDirection(int direction)
```

* Parameter

- direction: direction of printing. Refer to the following table for possible options.

Code	Value	Description
DIRECTION_0_DEGREE_ROTATION	0	Print without rotation
DIRECTION_90_DEGREE_ROTATION	1	Print clockwise 90° rotated image
DIRECTION_180_DEGREE_ROTATION	2	Print clockwise 180° rotated image
DIRECTION_270_DEGREE_ROTATION	3	Print clockwise 270° rotated image

7-7-6 setStandardMode

This method closes the page mode and switches to the standard mode.

* Syntax

```
public void setStandardMode()
```


* Example

```
public class MainActivity extends Activity {

    ...

    private T_3II t_3ii;

    public void onCreate(Bundle savedInstanceState) {

        ...

        t_3ii = new T_3II(this, mHandler, null);
        t_3ii.connect(null);

        ...

    }

    public void onDestroy() {

        ...

        if (t_3ii != null) {
            t_3ii.disconnect();
        }

        ...

    }

    private final Handler mHandler = new Handler(new Handler.Callback() {

        public boolean handleMessage(Message msg) {
            switch (msg.what) {
                case MESSAGE_STATE_CHANGE:
                    if (msg.arg1 == STATE_CONNECTED) {
                        t_3ii.setPageMode();
                        t_3ii.setPrintDirection(
T_3II.DIRECTION_180_DEGREE_ROTATION);
                        t_3ii.setPrintArea(0, 0, 384, 840);
                        t_3ii.setAbsoluteVerticalPrintPosition(100);
                        t_3ii.printBitmap(bitmap,
                                    T_3II.ALIGNMENT_CENTER,
                                    T_3II.BITMAP_WIDTH_FULL, 88, false);
                        t_3ii.formFeed(true);
                        t_3ii.setStandardMode();
                    }
                    break;

                case MESSAGE_PRINT_COMPLETE:
                    t_3ii.disconnect();
                    break;
            }
            return true;
        }

    });
}
```

7-8 Settings

7-8-1 setSingleByteFont

This method sets the single byte font.

* Syntax

```
public void setSingleByte(int codePage)
```

* Parameters

- codePage: code page to set

Code	Value	Description
CODE_PAGE_437_USA	0	Page 0 437(USA standard Europe)
CODE_PAGE_KATAKANA	1	Page 1 Katakana
CODE_PAGE_850_MULTILINGUAL	2	Page 2 850 (Multilingual)
CODE_PAGE_860_PORTUGUESE	3	Page 3 860 (Portuguese)
CODE_PAGE_863_CANADIAN_FRENCH	4	Page 4 863 (Canadian-French)
CODE_PAGE_865_NORDIC	5	Page 5 865 (Nordic)
CODE_PAGE_1252_LATIN1	16	Page 16 1252 (Latin I)
CODE_PAGE_866_CYRILLIC2	17	Page 17 866 (Cyrillic #2)
CODE_PAGE_852_LATIN2	18	Page 18 852 (Latin 2)
CODE_PAGE_858_EURO	19	Page 19 858 (Euro)
CODE_PAGE_862_HEBREW_DOS_CODE	21	Page 21 862 (Hebrew DOS code)
CODE_PAGE_864_ARABIC	22	Page 22 864 (Arabic)
CODE_PAGE_THAI42	23	Page 23 Thai42
CODE_PAGE_1253_GREEK	24	Page 24 1253 (Greek)
CODE_PAGE_1254_TURKISH	25	Page 25 1254 (Turkish)
CODE_PAGE_1257_BALTIC	26	Page 26 1257 (Baltic)
CODE_PAGE_FARSI	27	Page 27 Farsi
CODE_PAGE_1251_CYRILLIC	28	Page 28 1251 (Cyrillic)
CODE_PAGE_737_GREEK	29	Page 29 737 (Greek)
CODE_PAGE_775_BALTIC	30	Page 30 775 (Baltic)
CODE_PAGE_THAI14	31	Page 31 Thai14
CODE_PAGE_1255_HEBREW_NEW_CODE	33	Page 33 1255 (Hebrew Newcode)
CODE_PAGE_THAI11	34	Page 34 Thai11
CODE_PAGE_THAI18	35	Page 35 Thai18
CODE_PAGE_855_CYRILLIC	36	Page 36 855 (Cyrillic)
CODE_PAGE_857_TURKISH	37	Page 37 857 (Turkish)
CODE_PAGE_928_GREEK	38	Page 38 928 (Greek)
CODE_PAGE_THAI16	39	Page 39 Thai16
CODE_PAGE_1256_ARABIC	40	Page 40 1256 (ARB)
CODE_PAGE_1258_VIETNAM	41	Page 41 1258 (Vietnam)
CODE_PAGE_KHMER_CAMBODIA	42	Page 42 Khmer (Cambodia)
CODE_PAGE_1250_CZECH	43	Page 47 1250 (Czech)

* Example

```
public class MainActivity extends Activity {

    ...

    private T_3II t_3ii;

    ...

    public void onCreate(Bundle savedInstanceState) {

        ...

        t_3ii = new T_3II(this, mHandler, null);
        t_3ii.connect(null);

        ...

    }

    public void onDestroy() {

        ...

        if (t_3ii != null) {
            t_3ii.disconnect();
        }

        ...

    }

    private final Handler mHandler = new Handler(new Handler.Callback() {

        public boolean handleMessage(Message msg) {
            switch (msg.what) {
                case T_3II.MESSAGE_STATE_CHANGE:
                    if (msg.arg1 == T_3II.STATE_CONNECTED) {
                        t_3ii.setSingleByteFont(
                            T_3II.CODE_PAGE_437_USA);
                    }
                    break;
            }
            return true;
        }
    });
}
```

7-9 Miscellaneous Functions

7-9-1 cutPaper

This method cuts paper.

*** Syntax**

```
public void cutPaper(boolean getResponse)
```

*** Paramters**

- **getResponse**: When this parameter is set to True, the MESSAGE_PRINT_COMPLETE message is sent to the application handler when paper cut operation is completed. If it is set to False, message is not sent to the application handler after paper is cut.

* Example

```
public class MainActivity extends Activity {

    ...

    private T_3II t_3ii;

    ...

    public void onCreate(Bundle savedInstanceState) {

        ...

        t_3ii = new T_3II(this, mHandler, null);
        t_3ii.connect(null);

        ...

    }

    private final Handler mHandler = new Handler(new Handler.Callback() {

        public boolean handleMessage(Message msg) {
            switch (msg.what) {
                case T_3II.MESSAGE_STATE_CHANGE:
                    if (msg.arg1 == T_3II.STATE_CONNECTED) {
                        // TODO: Set getResponse parameter to False and
                        print t_3ii.cutPaper(true);
                    }
                    break;

                case T_3II.MESSAGE_PRINT_COMPLETE:
                    t_3ii.disconnect();
                    break;
            }
            return true;
        }
    });
}
```

7-9-2 cutPaper

This method feeds the paper by the specified number of lines and cut the paper.

*** Syntax**

```
public void cutPaper(int feeds, Boolean getResponse)
```

*** Paramters**

- feeds: number of lines to feed before cutting paper
- getResponse: When this parameter is set to True, the MESSAGE_PRINT_COMPLETE message is sent to the application handler when paper cut operation is completed. If it is set to False, message is not sent to the application handler after paper is cut.

* Example

```
public class MainActivity extends Activity {

    ...

    private T_3II t_3ii;

    ...

    public void onCreate(Bundle savedInstanceState) {

        ...

        t_3ii = new T_3II(this, mHandler, null);
        t_3ii.connect(null);

        ...

    }

    private final Handler mHandler = new Handler(new Handler.Callback() {

        public boolean handleMessage(Message msg) {
            switch (msg.what) {
                case T_3II.MESSAGE_STATE_CHANGE:
                    if (msg.arg1 == T_3II.STATE_CONNECTED) {
                        // TODO: Set getResponse parameter to False and
                        print
                        t_3ii.cutPaper(10, true);
                    }
                    break;

                case T_3II.MESSAGE_PRINT_COMPLETE:
                    t_3ii.disconnect();
                    break;
            }
            return true;
        }
    });
}
```

7-9-3 executeDirectIo

This method sends command directly to printer. If response is to be received from the printer, the MESSAGE_READ message (arg1: PROCESS_EXECUTE_DIRECT_IO) should be sent to the application handler.

*** Syntax**

```
public void executeDirectIo(byte[] command, boolean hasResponse)
```

*** Parameters**

- command: command to send to the printer
- hasResponse: Set this True if response to the command is to be received from the printer, or False if not.

* Example

```
public class MainActivity extends Activity {

    ...

    private T_3II t_3ii;

    ...

    public void onCreate(Bundle savedInstanceState) {

        ...

        t_3ii = new T_3II(this, mHandler, null);
        t_3ii.connect(null);

        ...

    }

    public void onDestroy() {

        ...

        if (t_3ii != null) {
            t_3ii.disconnect();
        }

        ...

    }

    private final Handler mHandler = new Handler(new Handler.Callback() {

        public boolean handleMessage(Message msg) {
            switch (msg.what) {
                case T_3II.MESSAGE_STATE_CHANGE:
                    if (msg.arg1 == T_3II.STATE_CONNECTED) {
                        byte[] command = new byte[] {0x10, 0x04, 0x02};
                        t_3ii.executeDirectIo(command, true);
                    }
                    break;

                case T_3II.MESSAGE_READ:
                    if (msg.arg1 == T_3II.PROCESS_EXECUTE_DIRECT_IO) {
                        Bundle data = msg.getData();
                        byte[] response =
                            data.getBytes(T_3II.KEY_STRING_DIRECT_IO);
                        // TODO: response time
                    }
                    break;
            }
            return true;
        }
    });
}
```

7-9-4 getAddress

This method obtains and returns network MAC address of the printer connected with LAN or wireless LAN.

* Syntax

```
public String getAddress()
```

* Example

```
public class MainActivity extends Activity {

    ...

    private T_3II t_3ii;

    ...

    public void onCreate(Bundle savedInstanceState) {

        ...

        t_3ii = new T_3II(this, mHandler, null);
        t_3ii.connect("192.168.0.100", 9100, 5000);

        ...

    }

    public void onDestroy() {

        ...

        if (t_3ii != null) {
            t_3ii.disconnect();
        }

        ...

    }

    private final Handler mHandler = new Handler(new Handler.Callback() {

        public boolean handleMessage(Message msg) {
            switch (msg.what) {
                case T_3II.MESSAGE_STATE_CHANGE:
                    if (msg.arg1 == T_3II.STATE_CONNECTED) {
                        String macAddress = t_3ii.getMacAddress();
                        Toast.makeText(getApplicationContext(),
macAddress,
                                                                    Toast.LENGTH_SHORT).show();
                    }
                    break;
            }
            return true;
        }
    });
}
```

7-9-5 initialize

This method initializes the printer settings to a state the same as after booting. The data in the printer buffer is initialized but the data in the printer receive buffer is not. NV image stored in the printer is not initialized. If the printer is in the page mode, all data in the print area is removed and the printer is initialized to the standard mode.

* Syntax

```
public void initialize()
```

* Example

```
public class MainActivity extends Activity {

    ...

    private T_3II t_3ii;

    ...

    public void onCreate(Bundle savedInstanceState) {

        ...

        t_3ii = new T_3II(this, mHandler, null);
        t_3ii.connect();

        ...

    }

    public void onDestroy() {

        ...

        if (t_3ii != null) {
            t_3ii.disconnect();
        }

        ...

    }

    private final Handler mHandler = new Handler(new Handler.Callback() {

        public boolean handleMessage(Message msg) {
            switch (msg.what) {
                case T_3II.MESSAGE_STATE_CHANGE:
                    if (msg.arg1 == T_3II.STATE_CONNECTED) {
                        t_3ii.initialize();
                    }
                    break;
            }
            return true;
        }

    });
}
```

7-9-6 kickOutDrawer

This method opens cash drawer or run melody box.

* Syntax

```
public void kickOutDrawer(int connectorPin)
```

* Parameter

- connectorPin: connector pin of cash register or melody box

Code	Value	Description
DRAWER_CONNECTOR_PIN2	0	Connector pin 2
DRAWER_CONNECTOR_PIN5	1	Connector pin 5

* Example

```
public class MainActivity extends Activity {

    ...

    private T_3II t_3ii;

    ...

    public void onCreate(Bundle savedInstanceState) {

        ...

        t_3ii = new T_3II(this, mHandler, null);
        t_3ii.connect();

        ...

    }

    public void onDestroy() {

        ...

        if (t_3ii != null) {
            t_3ii.disconnect();
        }

        ...

    }

    private final Handler mHandler = new Handler(new Handler.Callback() {

        public boolean handleMessage(Message msg) {
            switch (msg.what) {
                case T_3II.MESSAGE_STATE_CHANGE:
                    if (msg.arg1 == T_3II.STATE_CONNECTED) {

                        t_3ii.kickOutDrawer(DRAWER_CONNECTOR_PIN5);
                    }
                    break;
            }
            return true;
        }
    });
}
```

7-9-7 updateFirmware

This method sends the specified firmware binary file to the printer and updates the printer firmware. Printer should be rebooted after updating printer firmware.

*** Syntax**

```
public void updateFirmware(String binaryFilePath)
```

*** Parameters**

- binaryFilePath: absolute path fo the firmware binary file to send to printer

* Example

```
public class MainActivity extends Activity {

    ...

    private T_3II t_3ii;

    ...

    public void onCreate(Bundle savedInstanceState) {

        ...

        t_3ii = new T_3II(this, mHandler, null);
        t_3ii.connect();

        ...

    }

    private final Handler mHandler = new Handler(new Handler.Callback() {

        public boolean handleMessage(Message msg) {
            switch (msg.what) {
                case T_3II.MESSAGE_STATE_CHANGE:
                    if (msg.arg1 == T_3II.STATE_CONNECTED) {
                        String binaryFilePath =

Environment.getExternalStorageDirectory().getAbsolutePath() +
                                "/firmware_binary.fls";
                        t_3ii.updateFirmware(binaryFilePath);

                        try {
                            Thread.sleep(5000);
                        } catch (InterruptedException e) {
                            // TODO Auto-generated catch block
                            e.printStackTrace();
                        }

                        t_3ii.disconnect();
                    }
                    break;
            }
            return true;
        }
    });
}
```

8. Programming

8-1 Programming Flow

Applications should be programmed in the following sequence.

1. Search printer
2. Open printer port
3. Send print data
4. Close printer port

8-2 Search Printer

Search available printers. This step can be skipped if the printer to connect is manually specified. Refer to the following codes.

```
...

private T_3II t_3ii;

...

    t_3ii = new T_3II(this, mHandler, null);
    t_3ii.findBluetoothPrinters();           // When connecting over Bluetooth
    // t_3ii.findNetworkPrinters(5000);      // When connecting over Network
    // t_3ii.findUsbPrinters();              // When connecting over USB

    private final Handler mHandler = new Handler(new Handler.Callback() {
        public boolean handleMessage(Message msg) {

            ...

        }
    })
```

8-3 Open Printer Port

Use the connect method to create printer instance and open the port. Refer to the following codes.

```
private final Handler mHandler = new Handler(new Handler.Callback() {
    public boolean handleMessage(Message msg) {
        switch (msg.what) {
            case T_3II.MESSAGE_BLUETOOTH_DEVICE_SET:
                if (msg.obj != null) {
                    Set<BluetoothDevice> bluetoothDeviceSet =
                        (Set<BluetoothDevice>) msg.obj;
                    BluetoothDevice[] bluetoothDevices =
                        bluetoothDeviceSet.toArray(
                            new
                                BluetoothDevice[bluetoothDeviceSet.size()]);
                    t_3ii.connect(bluetoothDevices[i].getAddress());
                }
                break;

            case T_3II.MESSAGE_USB_DEVICE_SET:
                if (msg.obj != null) {
                    Set<String> networkDeviceSet = (Set<String>) msg.obj;
                    String[] networkDevices =
                        networkDeviceSet.toArray(new
                            String[networkDeviceSet.size()]);
                    t_3ii.connect(networkDevices[i].getAddress(), 9100, 5000);
                }
                break;

            case T_3II.MESSAGE_NETWORK_DEVICE_SET:
                if (msg.obj != null) {
                    Set<UsbDevice> usbDeviceSet = (Set<UsbDevice>)
                        msg.obj;
                    UsbDevice[] usbDevices =
                        usbDeviceSet.toArray(new
                            UsbDevice[usbDeviceSet.size()]);
                    t_3ii.connect(usbDevices[i]);
                }
                break;
        }
        return true;
    }
}
```

8-4 Send Print Data

Application handler sends print data to the printer after receiving connection completion message. Refer to the following codes.

```
private final Handler mHandler = new Handler(new Handler.Callback() {
    public boolean handleMessage(Message msg) {
        switch (msg.what) {
            case T_3II.MESSAGE_STATE_CHANGE:
                if (msg.arg1 == T_3II.STATE_CONNECTED) {
                    t_3ii.printText("Text to print\n",
                                    T_3II.ALIGNMENT_LEFT,
                                    T_3II.TEXT_ATTRIBUTE_FONT_A,
                                    T_3II.TEXT_SIZE_HORIZONTAL1 |
                                    T_3II.TEXT_SIZE_VERTICAL1, false);
                }
                break;
        }
        return true;
    }
});
```

8-5 Close Printer Port

Close the printer connection when application is shutdown if connection with the printer is maintained while application is running. If printer is to be connected on demand, close the connection after receiving print completion message. Refer to the following codes.

```
private final Handler mHandler = new Handler(new Handler.Callback() {  
    public boolean handleMessage(Message msg) {  
        switch (msg.what) {  
            case T_3II.MESSAGE_PRINT_COMPLETE:  
                t_3ii.disconnect();  
                break;  
        }  
        return true;  
    }  
})
```