BIXOLON

API Reference Guide

BXL SDK for UPOS Compliant Android

Ver. 2.26

www.bixolon.com

Table of Contents

Copyright

BIXOLON Co., Ltd. maintains ongoing efforts to enhance and upgrade the functions and quality of all our products.
In the following, product specifications and/or user manual content may be changed without prior notice.

Caution

Some semiconductor devices are easily damaged by static electricity. You should turn the printer “OFF”, before you connect or remove the cables on the rear side, in order to guard the printer against the static electricity. If the printer is damaged by the static electricity, you should turn the printer “OFF”.

1. About This Manual

This SDK manual describes the library required for developing applications for Android.

It additionally describes how to use SDK, specifications, and restrictions.

2. Support OS and Interface

2-1 Operating System

• This software supports the following operating systems.
• Android 6.0 or later is required.

2-2 Supported Devices and Interfaces

Models	Interface	DPI	Max Printable Width
SPP-R200II	Bluetooth / WLAN / USB	203 dpi	384 dots
SPP-R200III	Bluetooth / WLAN / USB	203 dpi	384 dots
SPP-R210	Bluetooth / WLAN / USB	203 dpi	384 dots
SPP-R215	Bluetooth / USB	203 dpi	384 dots
SPP-R220	Bluetooth / BLE / WLAN / USB	203 dpi	384 dots
SPP-C200	Bluetooth / BLE / USB	203 dpi	384 dots
SPP-C300	Bluetooth / BLE / USB	203 dpi	576 dots
SPP-R300	Bluetooth / WLAN / USB	203 dpi	576 dots
SPP-R310	Bluetooth / WLAN / USB	203 dpi	576 dots
SPP-R400	Bluetooth / WLAN / USB	203 dpi	832 dots
SPP-R410	Bluetooth / BLE / WLAN / USB	203 dpi	832 dots
SPP-R418	Bluetooth / BLE / WLAN / USB	203 dpi	832 dots
SRP-350plusIII	Bluetooth / WLAN / Ethernet / USB	180 dpi	512 dots
SRP-352plusIII	Bluetooth / WLAN / Ethernet / USB	203 dpi	576 dots
SRP-350plusV	Bluetooth / WLAN / Ethernet / USB	180 dpi	512 dots
SRP-352plusV	Bluetooth / WLAN / Ethernet / USB	203 dpi	576 dots
SRP-350III	Ethernet / USB	180 dpi	512 dots
SRP-352III	Ethernet / USB	203 dpi	576 dots
SRP-350V	Ethernet / USB	180 dpi	512 dots
SRP-352V	Ethernet / USB	203 dpi	576 dots
SRP-F310II	Bluetooth / WLAN / Ethernet / USB	180 dpi	512 dots
SRP-F312II	Bluetooth / WLAN / Ethernet / USB	203 dpi	576 dots
SRP-F313II	Bluetooth / WLAN / Ethernet / USB	203 dpi	640 dots
SRP-380	Bluetooth / WLAN / Ethernet / USB	180 dpi	512 dots
SRP-382	Bluetooth / WLAN / Ethernet / USB	203 dpi	576 dots
SRP-383	Bluetooth / WLAN / Ethernet / USB	300 dpi	864 dots
SRP-380plus	Bluetooth / WLAN / Ethernet / USB	180 dpi	512 dots
SRP-382plus	Bluetooth / WLAN / Ethernet / USB	203 dpi	576 dots
SRP-330II	Ethernet / USB	180 dpi	512 dots
SRP-332II	Ethernet / USB	203 dpi	576 dots
SRP-330III	Ethernet / USB	180 dpi	512 dots
SRP-332III	Ethernet / USB	203 dpi	576 dots
SRP-S200	Bluetooth / WLAN / Ethernet / USB	203 dpi	434 dots
SRP-S300	Bluetooth / WLAN / Ethernet / USB	203 dpi	576 dots

SRP-S300II	Bluetooth / WLAN / Ethernet / USB	203 dpi	576 dots
SRP-S320	Bluetooth / WLAN / Ethernet / USB	203 dpi	576 dots
SRP-S3000	WLAN / Ethernet / USB	203 dpi	576 dots
SRP-340II	Ethernet / USB	180 dpi	512 dots
SRP-342II	Ethernet / USB	203 dpi	576 dots
STP-103III	USB	203 dpi	384 dots
SRP-275III	Ethernet / USB	160 x 144 dpi	400 dots
SRP-Q200	Bluetooth / WLAN / Ethernet / USB	203 dpi	434 dots
SRP-Q300	Bluetooth / WLAN / Ethernet / USB	180 dpi	512 dots
SRP-Q302	Bluetooth / WLAN / Ethernet / USB	203 dpi	576 dots
SRP-Q300II	Bluetooth / WLAN / Ethernet / USB	203 dpi	576 dots
SRP-QE300	Ethernet / USB	180 dpi	512 dots
SRP-QE302	Ethernet / USB	203 dpi	576 dots
SRP-E300	Ethernet / USB	180 dpi	512 dots
SRP-E302	Ethernet / USB	203 dpi	576 dots
SRP-B300	Ethernet / USB	180 dpi	546 dots
		203 dpi	576 dots
BK3-3	Ethernet / USB	203 dpi	576 dots
BK3R-3	Ethernet / USB	203 dpi	576 dots
BK3-2	USB	203 dpi	432 dots
BK5-3	USB	203 dpi	640 dots
BK5R-3	USB	203 dpi	640 dots
SPP-100II	USB	180 dpi	384 dots
G30	Bluetooth / WLAN / Ethernet / USB	203 dpi	576 dots

※ BLE: Bluetooth Low Energy

3. Development Environment

3-1 System Requirements

• Java Development Kit (JDK) 7
• Android Stuido
• Android SDK Tools
• Reference: http://developer.android.com/sdk/index.html

3-2 Connecting Android Device

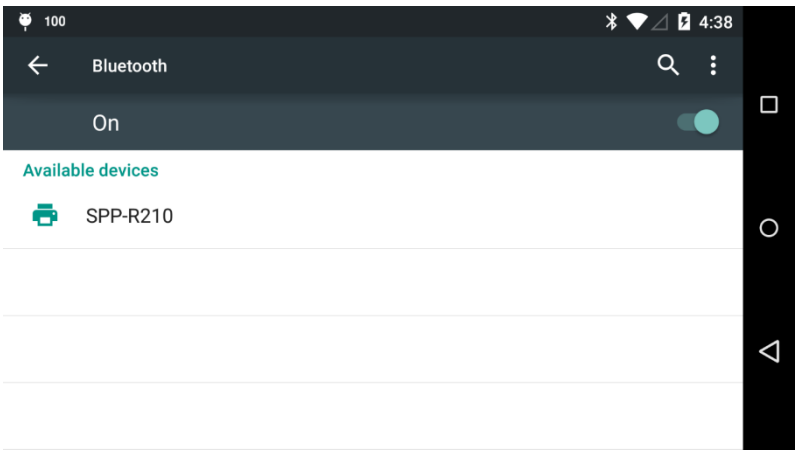
• The following screen was captured from an Nexus 5 smart phone.
The screen and field names might be different for different Android operating systems or devices.

3-2-1 Bluetooth

1. Select [Settings].

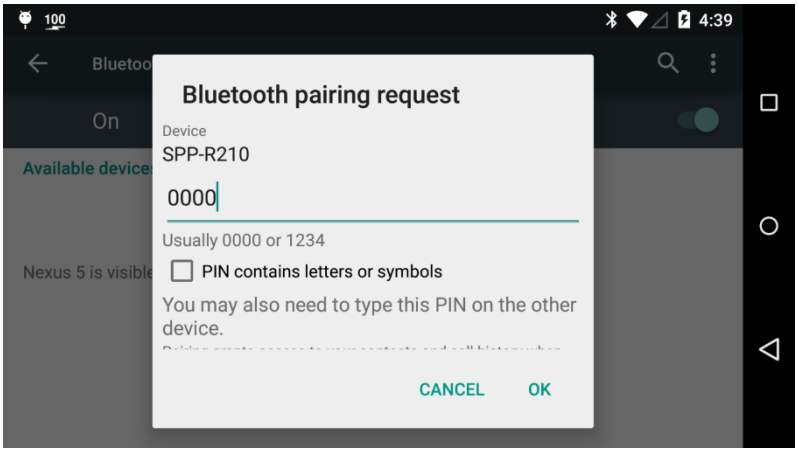
2. Bluetooth should be enabled and the printer power should be on.

3. Select [Bluetooth] for settings.



4. Select [Scan]. Search the printer to connect and perform pairing operation.

5. Enter PIN code. Default PIN code of BIXOLON is “0000”.



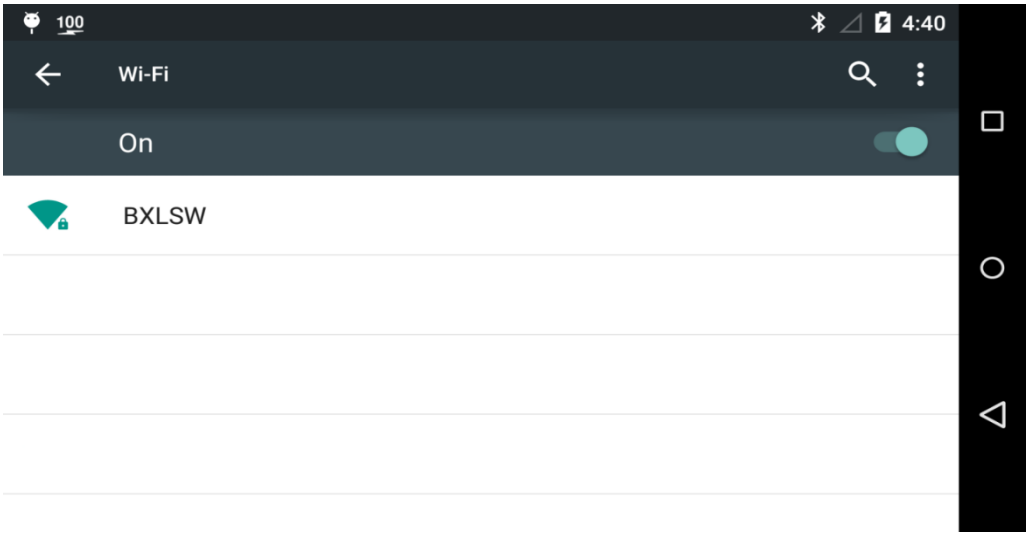
3-2-2 Network

1. Connect the printer to the network AP (Access Point) and assign an IP address or obtain one using DHCP. As BIXOLON's printer is initially set to Ad-hoc, it needs to be set up first with our Net Configuration Tool. The Net Configuration Tool can be downloaded from the BIXOLON website.
   (Refer to the Net Configuration Tool manual for details on settings)

2. Select [Settings].

3. Wi-Fi should be turned on.

4. Connect the device to the same network that the BIXOLON printer is connected to.



5. Additional setting is not required to connect the Android device to the TCP/IP port of printer.

3-2-3 Wi-Fi Direct

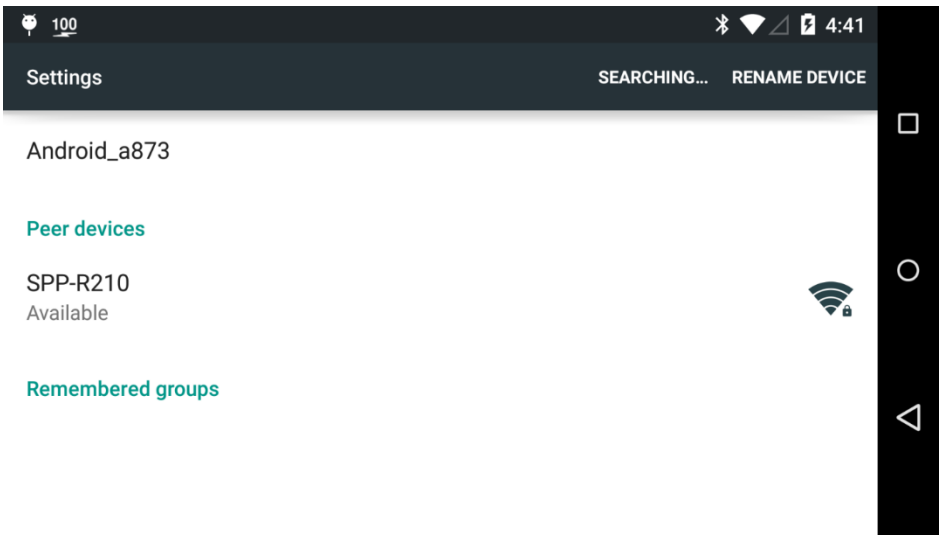
1. Android version should be 4.0 or higher in order to connect with peripheral devices using Wi-Fi Direct.

2. No special driver or printer software is required on Android device.

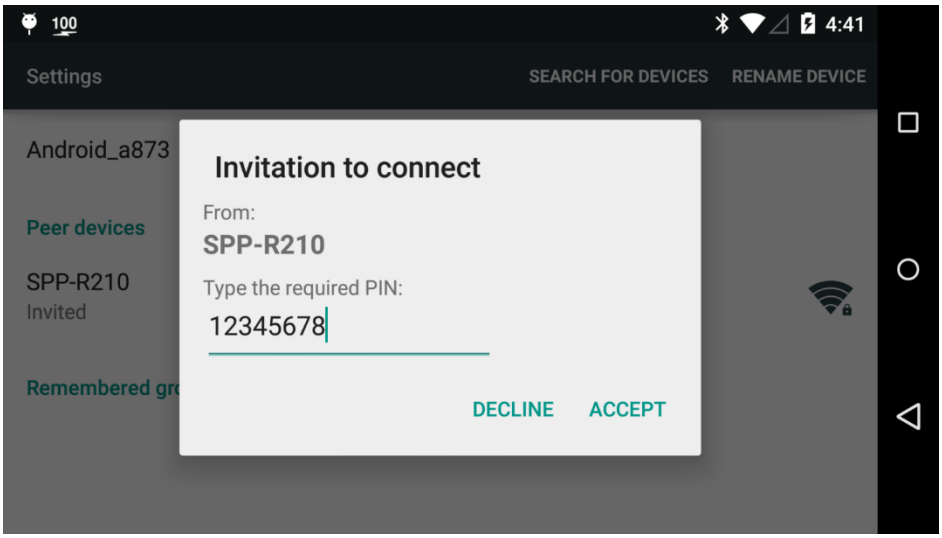
3. Select [Settings].

4. Wi-Fi should be turned on.

5. Select [Wifi Direct].



6. Select the printer from search list and connect it.
   Default PIN code of Wi-Fi Direct is “12345678”.



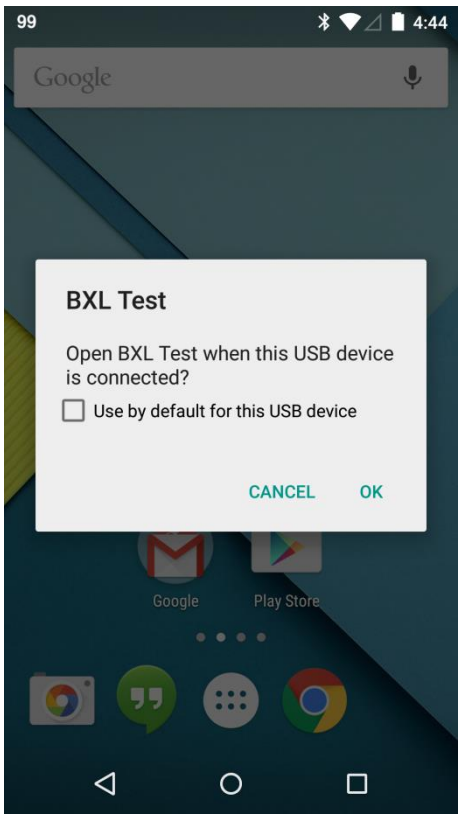
3-2-4 USB

1. Android version should 3.1 or higher to connect USB peripheral devices.

2. No special driver or printer software is required on Android device.

3. Required USB cable depends on specific smartphone or tablet.
   Be sure that the Android device to be used supports USB connection and use an appropriate cable.

4. The following message may pop up for some Android devices the first time the printer is connected to the device.



5. To connect a USB peripheral, the following code should be entered to AndroidManifest.xml and res/xml/device_filter.xml in BXLTest provided as a sample.

[AndroidManifest.xml]

```
...
<uses-feature android:name="android.hardware.usb.host" />
...

<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]

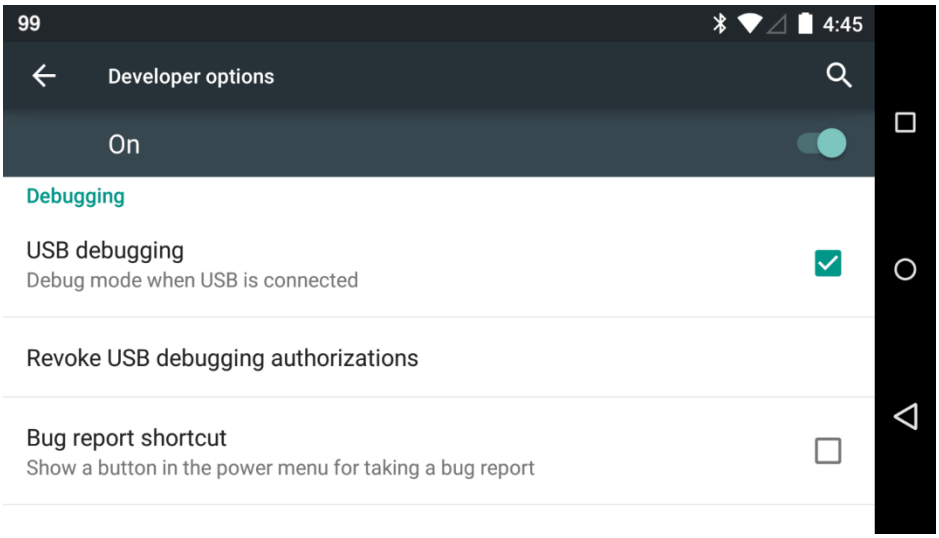
```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <usb-device
        class="7"
        protocol="2"
        subclass="1" />

</resources>
```

3-2-5 Setting Android Device Developer Options

1. Select [Settings].

2. Select [Developer options].

3. Enable [USB debugging].



4. Package Contents

4-1 Manual

Manual location/ Name	Description
docs/Manual_BXL_SDK_for_Android_UPOS_compliant_API_Reference_Guide_ENG_Ver_x.xx	Manual in English
docs/Manual_BXL_SDK_for_Android_UPOS_compliant_API_Reference_Guide_KOR_Ver_x.xx	Manual in Korean

4-2 Library

Library location/ Name	Description
libs/bixolon_printer_Vxxx.jar	Implementation of JavaPOS service component layers / printer setting library
libs/libcommon_Vxxx.jar	Printer control core library
libs/pdf/bixolon_pdf.aar	Library for PDF print
libs/jniLibs/ABI type/libbxl_common.so	Printer control native library

4-3 Sample source cord

Sample location/ Name	Description
samples/BixolonSample	Printer/MSR/SCR/CashDrawer control sample application

★ A sample reference class for better understanding of SDK usage is provided.
- BixolonPrinter.java(src\main\java\com\bixolon\sample\PrinterControl)

5. Constant Value (Defines)

5-1 JposException

• When an error occurs during execution of a specific function in a method, it throws a JposException exception. If an exception occurs, the contents of the error can be checked through the "printStackTrace" function.

[Example]

```
import jpos.JposConst;


try
{
      ......
}
catch(JposException e)
{
      // Error
      e.printStackTrace();
}
```

5-2 Event

• Each Event is defined in the JposConst and POSPrinterConst class.

5-2-1 StatusUpdate Event

A StatusUpdate event occurs whenever the printer status changes.

Code	Value	Description
JPOS_SUE_POWER_ONLINE	2001	Printer Power on
JPOS_SUE_POWER_OFF_OFFLINE	2004	Printer Power off
PTR_SUE_COVER_OPEN	11	Cover Open
PTR_SUE_COVER_OK	12	Cover OK
PTR_SUE_REC_EMPTY	24	Receipt Paper Empty
PTR_SUE_REC_NEAREMPTY	25	Receipt Paper Near Empty
PTR_SUE_REC_PAPEROK	26	Receipt Paper OK
PTR_SUE_IDLE	1001	Printer Idle
PTR_SUE_OFF_LINE	53	Printer off-line
PTR_SUE_ON_LINE	54	Printer on-line
PTR_SUE_BAT_OK	55	Printer battery normal
PTR_SUE_BAT_LOW	56	Printer battery low

5-2-2 Error Event

Code	Value	Description
JPOS_EPTR_COVER_OPEN	201	Cover Open
JPOS_EPTR_REC_EMPTY	203	Paper Empty
JPOS_EPTR_OFF_LINE	217	Printer off-line

5-2-3 OutputComplete Event

Generates a print completion event. However, it must be used in Async mode.

5-2-4 Data Event

Receives MSR Track information data.

5-2-5 DirectIO Event

Receives direct I/O response data.

[Event Example]

```
try
{
	POSPrinter posPrinter = new POSPrinter(this);
	MSR msr = new MSR();
	posPrinter.addErrorListener(this);
	posPrinter.addStatusUpdateListener(this);
	posPrinter.addOutputCompleteListener(this);
	posPrinter.addDirectIOListener(this);
	msr.addDataListener(this);
	posPrinter.setAsyncMode(true);
}
catch(JposException e)
{
	// Error
	e.printStackTrace();
}

@Override
public void outputCompleteOccurred(final OutputCompleteEvent e)
{
	runOnUiThread(new Runnable()
	{
		@Override
		public void run()
		{
			Toast.makeText(MainActivity.this, "complete print", Toast.LENGTH_SHORT).show();
		}
	});
}

@Override
public void dataOccurred(DataEvent arg0)
{
	// TODO Auto-generated method stub
	runOnUiThread(new Runnable()
	{
		@Override
		public void run()
		{
			try
			{
				String strData = new String(msr.getTrack1Data());
				strData += new String(msr.getTrack2Data());
				strData += new String(msr.getTrack3Data());
				Toast.makeText(MainActivity.this, strData, Toast.LENGTH_SHORT).show();
			}
			catch(JposException e)
			{
				Toast.makeText(MainActivity.this, e.getMessage(), Toast.LENGTH_SHORT).show();
			}
		}
	});
```

```
}

@Override
public void directIOOccurred(DirectIOEvent directIOEvent) {
runOnUiThread(new Runnable()
    {
        @Override
        public void run()
        {
            Toast.makeText(MainActivity.this, new String((byte[]) directIOEvent.getObject(),
    Toast.LENGTH_SHORT).show();
        }
    });
}

@Override
public void errorOccurred(final ErrorEvent arg0)
{
    // TODO Auto-generated method stub
    runOnUiThread(new Runnable()
    {
        @Override
    public void run()
    {
        switch (arg0.getErrorCodeExtended())
       {
        case POSPrinterConst.JPOS_EPTR_COVER_OPEN:
            return "Cover open";
        case POSPrinterConst.JPOS_EPTR_REC_EMPTY:
            return "Paper empty";
        case JposConst.JPOS_SUE_POWER_OFF_OFFLINE:
            return "Power off";
        default:
            return "Unknown";
      }
    }
    });
}

@Override
public void statusUpdateOccurred(final StatusUpdateEvent arg0)
{
    // TODO Auto-generated method stub
    runOnUiThread(new Runnable()
    {
        @Override
        public void run()
        {
                switch (arg0.getStatus())
                {
                    case JposConst.JPOS_SUE_POWER_ONLINE:
                        return "Power on";
                    case JposConst.JPOS_SUE_POWER_OFF_OFFLINE:
```

```
                        return "Power off";
                    case POSPrinterConst.PTR_SUE_COVER_OPEN:
                        return "Cover Open";
                    case POSPrinterConst.PTR_SUE_COVER_OK:
                        return "Cover OK";
                    case POSPrinterConst.PTR_SUE_REC_EMPTY:
                        return "Receipt Paper Empty";
                    case POSPrinterConst.PTR_SUE_REC_NEAREMPTY:
                        return "Receipt Paper Near Empty";
                    case POSPrinterConst.PTR_SUE_REC_PAPEROK:
                        return "Receipt Paper OK";
                    case POSPrinterConst.PTR_SUE_IDLE:
                        return "Printer Idle";
                    case POSPrinterConst.PTR_SUE_BAT_LOW:
                        return "Battery-Low";
                    case POSPrinterConst.PTR_SUE_BAT_OK:
                        return "Battery-OK";
                    default:
                        return "Unknown";
                }
            }
        });
}
```

5-3 EscapeSequence

• Values for specifying options such as font and thickness in the text to be printed.
It is in string form and is added before print data.

Define	Description
String ESCAPE_CHARACTERS = new String(new byte[] {0x1b, 0x7c})	**Escape Characters**
ESCAPE_CHARACTERS + "N"	Normal
ESCAPE_CHARACTERS + "aM"	Font A (12x24)
ESCAPE_CHARACTERS + "bM"	Font B (9x17)
ESCAPE_CHARACTERS + "cM"	Font C (9x24)
ESCAPE_CHARACTERS + "lA"	Left justify
ESCAPE_CHARACTERS + "cA"	Center
ESCAPE_CHARACTERS + "rA"	Right justify
ESCAPE_CHARACTERS + "bC"	Bold
ESCAPE_CHARACTERS + "!bC"	Disabled bold
ESCAPE_CHARACTERS + "uC"	Underline
ESCAPE_CHARACTERS + "!uC"	Disabled underline
ESCAPE_CHARACTERS + "rvC"	Reverse video
ESCAPE_CHARACTERS + "!rvC"	Disabled reverse video
ESCAPE_CHARACTERS + "1C"	Single high and wide
ESCAPE_CHARACTERS + "2C"	Double wide
ESCAPE_CHARACTERS + "3C"	Double high
ESCAPE_CHARACTERS + "4C"	Double high and wide
ESCAPE_CHARACTERS + "1hC"	Scale 1 time horizontally
ESCAPE_CHARACTERS + "2hC"	Scale 2 times horizontally
ESCAPE_CHARACTERS + "3hC"	Scale 3 times horizontally
ESCAPE_CHARACTERS + "4hC"	Scale 4 times horizontally
ESCAPE_CHARACTERS + "5hC"	Scale 5 times horizontally
ESCAPE_CHARACTERS + "6hC"	Scale 6 times horizontally
ESCAPE_CHARACTERS + "7hC"	Scale 7 times horizontally
ESCAPE_CHARACTERS + "8hC"	Scale 8 times horizontally
ESCAPE_CHARACTERS + "1vC"	Scale 1 time vertically
ESCAPE_CHARACTERS + "2vC"	Scale 2 times vertically
ESCAPE_CHARACTERS + "3vC"	Scale 3 times vertically
ESCAPE_CHARACTERS + "4vC"	Scale 4 times vertically
ESCAPE_CHARACTERS + "5vC"	Scale 5 times vertically
ESCAPE_CHARACTERS + "6vC"	Scale 6 times vertically
ESCAPE_CHARACTERS + "7vC"	Scale 7 times vertically
ESCAPE_CHARACTERS + "8vC"	Scale 8 times vertically
ESCAPE_CHARACTERS + "[code]B"	Print NV Image [code] : 0 ~ 255(image code)

ESCAPE_CHARACTERS + "[percentage]fP"	Feed Cut [percentage] : 100 Full cut [percentage] : 0~90 Partial cut

[Example]

```
try
{
	POSPrinter posPrinter = new POSPrinter(this);
	posPrinter.open(logicalDeviceName);
	posPrinter.claim(5000);
	posPrinter.setDeviceEnabled(true)

	String ESCAPE_CHARACTERS = new String(new byte[] {0x1b, 0x7c})

	// in bold
	String data = ESCAPE_SEQUENCE + "bC" + “Bixolon\n”
	posPrinter.printNormal(POSPrinterConst.PTR_S_RECEIPT, data);

	// not in bold
	String data = ESCAPE_SEQUENCE + "!bC" + “Bixolon\n”
	posPrinter.printNormal(POSPrinterConst.PTR_S_RECEIPT, data);

	// NV Image
	String data = ESCAPE_SEQUENCE + "0B";
	posPrinter.printNormal(POSPrinterConst.PTR_S_RECEIPT, data);

	// Feed cut
	String data = ESCAPE_SEQUENCE + "90fP";
	posPrinter.printNormal(POSPrinterConst.PTR_S_RECEIPT, data);
}
catch(JposException e)
{
	// Error
	e.printStackTrace();
}
```

5-4 Transaction Print

• Definitions of values that can be used when setting Transaction mode.

Code	Value	Description
PTR_TP_TRANSACTION	11	Initializes the buffer to empty status and starts Transaction Mode
PTR_TP_NORMAL	12	Terminates the transaction mode and outputs the accumulated data in the buffer

5-5 Alignment

• Definitions of the values required to specify alignment

[For Barcode]

Code	Value	Description
PTR_BC_LEFT	-1	Align to left
PTR_BC_CENTER	-2	Align to center
PTR_BC_RIGHT	-3	Align to right

[For Image]

Code	Value	Description
PTR_BM_LEFT	-1	Align to left
PTR_BM_CENTER	-2	Align to center
PTR_BM_RIGHT	-3	Align to right

5-6 Barcode type

• Definitions of the values required to specify barcode type when barcode is printed

Code	Value	Description
PTR_BCS_UPCA	101	UPCA
PTR_BCS_UPCE	102	UPCE
PTR_BCS_JAN13	103	JAN13
PTR_BCS_EAN13	103	EAN13
PTR_BCS_JAN8	104	JAN8
PTR_BCS_EAN8	104	EAN8
PTR_BCS_Code93	105	Code93
PTR_BCS_ITF	106	Interleaved 2 of 5
PTR_BCS_Codabar	107	Codabar
PTR_BCS_Code93	108	Code93
PTR_BCS_Code128	109	Code 128 ※ Special Character of Code128Special Charaters Special Charaters / Ascii Represntation Code A / {A Code B / {B Code C / {C
PTR_BCS_GS1DATABAR	110	GS1 DataBar Omnidirectional
PTR_BCS_PDF417	201	PDF 417
PTR_BCS_QRCODE	202	QR Code
PTR_BCS_MAXICODE	203	MAXI Code
PTR_BCS_DATAMATRIX	204	Data Matrix
PTR_BCS_EAN128	120	EAN 128

5-7 Barcode Text Location

• In case of barcodes supporting text printing, it specifies if the barcode text is printed, or the location of printing.

Code	Value	Description
PTR_BC_TEXT_NONE	-11	Does not print the text. Prints the barcode only.
PTR_BC_TEXT_ABOVE	-12	Prints the text at the top of the barcode.
PTR_BC_TEXT_BELOW	-13	Prints the text at the bottom of the barcode.

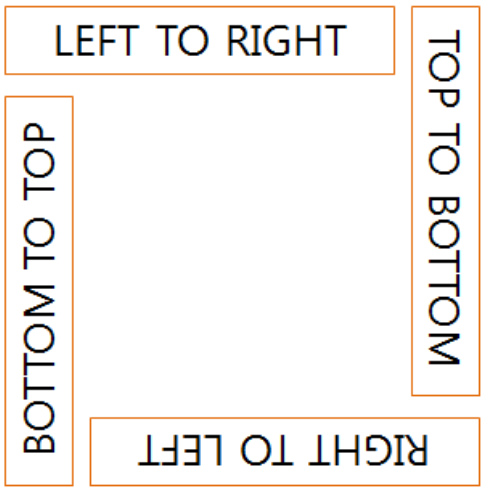
5-8 Device Model Name

Code	Description
PRODUCT_NAME_SPP_R210	SPP-R210
PRODUCT_NAME_SPP_R220	SPP-R220
PRODUCT_NAME_SPP_R200II	SPP-R200II
PRODUCT_NAME_SPP_R200III	SPP-R200III
PRODUCT_NAME_SPP_C200	SPP-C200
PRODUCT_NAME_SPP_C300	SPP-C300
PRODUCT_NAME_SPP_R300	SPP-R300
PRODUCT_NAME_SPP_R310	SPP-R310
PRODUCT_NAME_SPP_R400	SPP-R400
PRODUCT_NAME_SPP_R410	SPP-R410
PRODUCT_NAME_SPP_R418	SPP-R418
PRODUCT_NAME_SRP_350III	SRP-350III
PRODUCT_NAME_SRP_352III	SRP-352III
PRODUCT_NAME_SRP_350V	SRP-350V
PRODUCT_NAME_SRP_352V	SRP-352V
PRODUCT_NAME_SRP_330II	SRP-330II
PRODUCT_NAME_SRP_332II	SRP-332II
PRODUCT_NAME_SRP_330III	SRP-330III
PRODUCT_NAME_SRP_332III	SRP-332III
PRODUCT_NAME_SRP_340II	SRP-340II
PRODUCT_NAME_SRP_342II	SRP-342II
PRODUCT_NAME_SRP_350PLUSIII	SRP-350plusIII
PRODUCT_NAME_SRP_352PLUSIII	SRP-352plusIII
PRODUCT_NAME_SRP_350PLUSV	SRP-350plusV
PRODUCT_NAME_SRP_352PLUSV	SRP-352plusV
PRODUCT_NAME_SRP_380	SRP-380
PRODUCT_NAME_SRP_382	SRP-382
PRODUCT_NAME_SRP_383	SRP-383
PRODUCT_NAME_SRP_380II	SRP-380II
PRODUCT_NAME_SRP_382II	SRP-382II
PRODUCT_NAME_SRP_S200	SRP-S200
PRODUCT_NAME_SRP_S300	SRP-S300
PRODUCT_NAME_SRP_S300II	SRP-S300II
PRODUCT_NAME_SRP_S320	SRP-S320
PRODUCT_NAME_SRP_Q200	SRP-Q200
PRODUCT_NAME_SRP_S3000	SRP-S3000
PRODUCT_NAME_SRP_Q300	SRP-Q300
PRODUCT_NAME_SRP_Q302	SRP-Q302
PRODUCT_NAME_SRP_Q300II	SRP-Q300II

PRODUCT_NAME_SRP_F310II	SRP-F310II
PRODUCT_NAME_SRP_F312II	SRP-F312II
PRODUCT_NAME_SRP_F313II	SRP-F313II
PRODUCT_NAME_STP_103III	STP-103III
PRODUCT_NAME_SRP_275III	SRP-275III
PRODUCT_NAME_SRP_QE300	SRP-QE300
PRODUCT_NAME_SRP_QE302	SRP-QE302
PRODUCT_NAME_SRP_E300	SRP-E300
PRODUCT_NAME_SRP_E302	SRP-E302
PRODUCT_NAME_SRP_B300	SRP-B300
PRODUCT_NAME_BK3_3	BK3-3
PRODUCT_NAME_BK3_2	BK3-2
PRODUCT_NAME_BK5_3	BK5-3
PRODUCT_NAME_SPP_100II	SPP-100II
PRODUCT_NAME_G30	G30

5-9 Print Direction in Page Mode

Code	Value	Description
PTR_PD_LEFT_TO_RIGHT	1	Prints left to right
PTR_PD_BOTTOM_TO_TOP	2	Prints from bottom to top
PTR_PD_RIGHT_TO_LEFT	3	Prints right to left
PTR_PD_TOP_TO_BOTTOM	4	Prints from top to bottom



5-10 MSR Encryption

Code	Value	Description
MSR_DE_NONE	1	Disables data encryption algorithm
MSR_DE_3DEA_DUKPT	2	Enables data encryption algorithm

5-11 SCR Mode

Code	Value	Description
SC_CMODE_ISO	1	ISO Mode
SC_CMODE_EMV	2	EMV Mode

5-12 Character Set

Code	Value	Description
CS_437_USA_STANDARD_EUROPE	437	PC437
CS_737_GREEK	737	PC737
CS_775_BALTIC	775	PC775
CS_850_MULTILINGUAL	850	PC850
CS_852_LATIN2	852	PC852
CS_855_CYRILLIC	855	PC855
CS_857_TURKISH	857	PC857
CS_858_EURO	858	PC858
CS_860_PORTUGUESE	860	PC860
CS_862_HEBREW_DOS_CODE	862	PC862
CS_863_CANADIAN_FRENCH	863	PC863
CS_864_ARABIC	864	PC864
CS_865_NORDIC	865	PC865
CS_866_CYRILLIC2	866	PC866
CS_928_GREEK	928	PC928
CS_1250_CZECH	1250	WPC1250
CS_1251_CYRILLIC	1251	WPC1251
CS_1252_LATIN1	1252	WPC1252
CS_1253_GREEK	1253	WPC1253
CS_1254_TURKISH	1254	WPC1254
CS_1255_HEBREW_NEW_CODE	1255	WPC1255
CS_1256_ARABIC	1256	WPC1256
CS_1257_BALTIC	1257	WPC1257
CS_1258_VIETNAM	1258	WPC1258
CS_FARSI	7065	FARSI
CS_KATAKANA	7565	KATAKANA
CS_KHMER_CAMBODIA	7572	KHMER
CS_THAI11	8411	THAI11
CS_THAI14	8414	THAI14
CS_THAI16	8416	THAI16
CS_THAI18	8418	THAI18
CS_THAI42	8442	THAI42
CS_KS5601	5601	KS5601
CS_BIG5	6605	BIG5
CS_GB2312	2312	GB2312
CS_SHIFT_JIS	8374	SHIFT-JIS
CS_TCVN_3_1	3031	TCVN-3(1)
CS_TCVN_3_2	3032	TCVN-3(2)

6. Functions by Class

6-1 BXL Config Loader Class

• This is a class to save device setting information to be connected. The setting information manages device information through the BXLConfigLoader Class. The setting information includes the device name, product name, interface, etc., and if the information is not saved normally, the device cannot be connected. Before calling the Open function, this class must be called to save the setting information.

The device which is not saved with BXLConfigLoader cannot be connected.

6-1-1 Open File()

Opens existing saved setting file.

[Syntax]
void openFile() throws Exception

[Example]

```
import com.bxl.config.editor.BXLConfigLoader;
import jpos.config.JposEntry;

try
{
    BXLConfigLoader bxlConfigLoader = new BXLConfigLoader(this);
    bxlConfigLoader.openFile();
}
catch(Exception e)
{
    // Error
    e.printStackTrace();
}
```

6-1-2 New File()

Creates a new setting save file. Normally, it is performed when an openFile exception occurs.

[Syntax]
void newFile() throws Exception

[Example]

```
import com.bxl.config.editor.BXLConfigLoader;
import jpos.config.JposEntry;

try
{
	BXLConfigLoader bxlConfigLoader = new BXLConfigLoader(this);
	bxlConfigLoader.openFile();
}
catch(Exception e)
{
	e.printStackTrace();
	bxlConfigLoader.newFile();
}
```

6-1-3 Get Entries()

Obtains saved setting information.

[Syntax]
List<JposEntry> getEntries() throws Exception

[Return Values]

Value	Description
List container of JposEntry	Listing of objects

[Example]

```
import com.bxl.config.editor.BXLConfigLoader;
import jpos.config.JposEntry;

try
{
	for (Object entry : bxlConfigLoader.getEntries())
	{
		JposEntry jposEntry = (JposEntry)entry;
		String strLogicalname = jposEntry.getLogicalName();
	}
}
catch(Exception e)
{
	e.printStackTrace();
}
```

6-1-4 Add Entry()

Adds device connection information.

[Syntax]
void addEntry(String logicalName, int deviceCategory, String productName,
int deviceBus, String address) throws IllegalArgumentException

[Parameters]
• String logicalName: Enter the logical name(nickname) of the device.
• int deviceCategory: Select the type of device.

Code	Value	Description
DEVICE_CATEGORY_CASH_DRAWER	0	Cash Drawer
DEVICE_CATEGORY_MSR	1	MSR
DEVICE_CATEGORY_POS_PRINTER	2	POS Printer
DEVICE_CATEGORY_SMART_CARD_RW	3	SCR

• String productName: Device model name(Refer to “5-8 Device Model Name”)
• int deviceBus: Select the interface type.

Code	Value	Description
DEVICE_BUS_BLUETOOTH	0	Bluetooth
DEVICE_BUS_ETHERNET	1	Ethernet
DEVICE_BUS_USB	2	USB
DEVICE_BUS_WIFI	3	WiFi
DEVICE_BUS_WIFI_DIRECT	4	WiFi-Driect
DEVICE_BUS_BLUETOOTH_LE	5	Bluetooth Low Energy

• String address: Enter the MAC or IP address of the device.
(BT: MAC Address, Netwrok: IP Address)

[Example]

```
import com.bxl.config.editor.BXLConfigLoader;
import jpos.config.JposEntry;

try
{
    BXLConfigLoader bxlConfigLoader = new BXLConfigLoader(this);
    bxlConfigLoader.addEntry("SPP-R200III",
                             BXLConfigLoader.DEVICE_CATEGORY_POS_PRINTER,
                             BXLConfigLoader.PRODUCT_NAME_SPP_R200III,
                             BXLConfigLoader.DEVICE_BUS_BLUETOOTH,
                             "74:F0:7D:E4:11:AF");
}
catch(Exception e)
{
    e.printStackTrace();
}
```

6-1-5 removeEntry()

Deletes saved setting information.

[Syntax]
boolean removeEntry(String logicalName)

[Parameters]
• String logicalName: Enter the logical name (nickname) of the device.

[Return Values]

Value	Description
true	Returned on success
false	Returned on failure

[Example]

```
import com.bxl.config.editor.BXLConfigLoader;
import jpos.config.JposEntry;

try
{
	for (Object entry : bxlConfigLoader.getEntries())
	{
		JposEntry jposEntry = (JposEntry)entry;
		String strLogicalname = jposEntry.getLogicalName();

		bxlConfigLoader.removeEntry(strLogicalname);
	}
}
catch(Exception e)
{
	e.printStackTrace();
}
```

6-1-6 Save File()

Saves changed information through addEntry and removeEntry.

[Syntax]
void saveFile()

[Example]

```
import com.bxl.config.editor.BXLConfigLoader;
import jpos.config.JposEntry;

try
{
    for (Object entry : bxlConfigLoader.getEntries())
    {
        JposEntry jposEntry = (JposEntry)entry;
        String strLogicalname = jposEntry.getLogicalName();

        bxlConfigLoader.removeEntry(strLogicalname);
    }

    bxlConfigLoader.addEntry("SPP-R200III",
                             BXLConfigLoader.DEVICE_CATEGORY_POS_PRINTER,
                             BXLConfigLoader.PRODUCT_NAME_SPP_R200III,
                             BXLConfigLoader.DEVICE_BUS_BLUETOOTH,
                             "74:F0:7D:E4:11:AF");

    bxlConfigLoader.saveFile();
}
catch(Exception e)
{
    e.printStackTrace();
}
```

The device added by addEntry function must be saved using the saveFile function.

6-2 POS Printer Class

• This is a class for POS printer control. Using this class, operations can be performed such as connecting/disconnecting printer and executing print jobs. It generates a JposException when an error occurs while performing a specific function. (Refer to "5-1 JposException")

6-2-1 Open()

It Initiates the use of printer class and includes initialization operations such as memory allocation. It must be first performed to call a Method above Claim. Devices not saved via the BXLConfigLoader Class will not be opened.

[Syntax]
void open(String logicalDeviceName) throws JposException

[Parameters]
• String logicalDeviceName : Enter the name of the device to be opened.

[Example]

```
import com.bxl.config.editor.BXLConfigLoader;
import jpos.config.JposEntry;
import jpos.POSPrinter;
import jpos.POSPrinterConst;

try
{
    POSPrinter posPrinter = new POSPrinter(this);
    posPrinter.open("SPP-R200III");
}
catch(JposException e)
{
    e.printStackTrace();
}
```

6-2-2 Claim()

It attempts to actually open the port included in the device information, and includes some initialization operations such as memory allocation initialization.
It must be first performed to enable the use of device.

[Syntax]
void claim(int timeout) throws JposException

[Parameters]
• int timeout: Attempt to open the port for the time specified in this parameter.

[Example]

```
import com.bxl.config.editor.BXLConfigLoader;
import jpos.config.JposEntry;
import jpos.POSPrinter;
import jpos.POSPrinterConst;

try
{
	POSPrinter posPrinter = new POSPrinter(this);
	posPrinter.open("SPP-R200III");
	posPrinter.claim(3000);
}
catch(JposException e)
{
	e.printStackTrace();
}
```

6-2-3 Set Device Enabled()

It includes whether the device will be used or not.
If the value of DeviceEnabled is disabled, the function may not be available.

[Syntax]
void setDeviceEnabled(boolean deviceEnabled) throws JposException

[Parameters]
• boolean deviceEnabled: Enter the device activation status.

Value	Description
true	Enabled
false	Disabled

[Example]

```
import com.bxl.config.editor.BXLConfigLoader;
import jpos.config.JposEntry;
import jpos.POSPrinter;
import jpos.POSPrinterConst;

try
{
	POSPrinter posPrinter = new POSPrinter(this);
	posPrinter.open("SPP-R200III");
	posPrinter.claim(5000);
	posPrinter.setDeviceEnabled(true);
}
catch(JposException e)
{
	e.printStackTrace();
}
```

6-2-4 Release()

Physically terminates use of the port of the claimed Device.
Operations such as freeing up memory can be performed.

[Syntax]
void release() throws JposException

[Example]

```
import com.bxl.config.editor.BXLConfigLoader;
import jpos.config.JposEntry;
import jpos.POSPrinter;
import jpos.POSPrinterConst;

try
{
    posPrinter.release();
}
catch(JposException e)
{
    e.printStackTrace();
}
```

6-2-5 Close()

Terminates use of the opened device.
Some operations such as freeing up memory may be performed.

[Syntax]
void close() throws JposException

[Example]

```
import com.bxl.config.editor.BXLConfigLoader;
import jpos.config.JposEntry;
import jpos.POSPrinter;
import jpos.POSPrinterConst;

try
{
    posPrinter.close();
    posPrinter.setDeviceEnabled(false);
}
catch(JposException e)
{
    e.printStackTrace();
}
```

6-2-6 Check Health()

Checks that the device is operating normally.
It can be used when Open / Claim / DeviceEnabled is normally performed.

[Syntax]
void checkHealth(int level) throws JposException

[Parameters]
• int level: fixed Value JposConst.JPOS_CH_INTERNAL

[Example]

```
import com.bxl.config.editor.BXLConfigLoader;
import jpos.config.JposEntry;
import jpos.POSPrinter;
import jpos.POSPrinterConst;

try
{
	POSPrinter posPrinter = new POSPrinter(this);
	posPrinter.open("SPP-R200III");
	posPrinter.claim(5000);
	posPrinter.setDeviceEnabled(true);

	posPrinter.checkHealth(JposConst.JPOS_CH_INTERNAL);
}
catch(JposException e)
{
	e.printStackTrace();
}
```

6-2-7 Set Async Mode()

Includes whether or not to use asynchronous mode.
If the asynchronous mode is true, the related method operates in asynchronous mode. If false, the related method operates in synchronous mode.

[Syntax]
void setAsyncMode(boolean asyncMode) throws JposException

[Parameters]
• boolean asyncMode: Select whether or not to use asynchronous mode.

Value	Description
true	asynchronous mode
false	synchronous mode

[Example]

```
import com.bxl.config.editor.BXLConfigLoader;
import jpos.config.JposEntry;
import jpos.POSPrinter;
import jpos.POSPrinterConst;

try
{
	POSPrinter posPrinter = new POSPrinter(this);
	posPrinter.open("SPP-R200III");
	posPrinter.claim(5000);
	posPrinter.setDeviceEnabled(true);

	posPrinter.setAsyncMode(true);
}
catch(JposException e)
{
	e.printStackTrace();
}
```

6-2-8 Set Character Set()

Sets the character set of the printer.

[Syntax]
void setCharacterSet(int characterSet) throws JposException

[Parameters]
• int characterSet: Select the character set to be set in the the printer.
(Refer to “5-12 Character Set”)

[Example]

```
import com.bxl.config.editor.BXLConfigLoader;
import jpos.config.JposEntry;
import jpos.POSPrinter;
import jpos.POSPrinterConst;
import com.bxl.BXLConst;

try
{
	POSPrinter posPrinter = new POSPrinter(this);
	posPrinter.open(“SPP-R200III”);
	posPrinter.claim(5000);
	posPrinter.setDeviceEnabled(true);

	posPrinter.setCharacterSet(BXLConst.CS_437_USA_STANDARD_EUROPE);
}
catch(JposException e)
{
	e.printStackTrace();
}
```

6-2-9 Set Character Encoding()

Sets the encoding of the data to be sent to the printer.

[Syntax]
void setCharacterEncoding(int characterEncoding) throws JposException

[Parameters]
• int characterEncoding: Select the data encoding type.

Code	Value	Description
CE_ASCII	0	ASCII(default)
CE_UTF8	1	UTF-8

[Example]

```
import com.bxl.config.editor.BXLConfigLoader;
import jpos.config.JposEntry;
import jpos.POSPrinter;
import jpos.POSPrinterConst;
import com.bxl.BXLConst;

try
{
	POSPrinter posPrinter = new POSPrinter(this);
	posPrinter.open("SPP-R200III");
	posPrinter.claim(5000);
	posPrinter.setDeviceEnabled(true);

	posPrinter.setCharacterEncoding(BXLConst.CE_ASCII);
}
catch(JposException e)
{
	e.printStackTrace();
}
```

6-2-10 Cut Paper()

Cuts the paper in the models with Auto Cutter.

[Syntax]
void cutPaper(int percentage) throws JposException

[Parameters]
• int percentage: Select the Full cut / Partial cut.
It works only in the models with Auto Cutter.

Value	Description
100	Full cut
90	Partial cut

[Example]

```
import com.bxl.config.editor.BXLConfigLoader;
import jpos.config.JposEntry;
import jpos.POSPrinter;
import jpos.POSPrinterConst;
import com.bxl.BXLConst;

try
{
	POSPrinter posPrinter = new POSPrinter(this);
	posPrinter.open("SPP-R200III");
	posPrinter.claim(5000);
	posPrinter.setDeviceEnabled(true);

	posPrinter.cutPaper(100);
}
catch(JposException e)
{
	e.printStackTrace();
}
```

6-2-11 Print BarCode()

Prints barcodes.

[Syntax]
void printBarCode(int station, String data, int symbology, int height, int width,
int alignment, int textPosition) throws JposException

[Parameters]
• int station: fixed Value PTR_S_RECEIPT
• String data: The data to be included in the barcode. The data allowed by the Barcode type may differ.
• int symbology: Select the type of barcode. (Refer to “5-6 Barcord type”)
• int height: Specify the height of the barcode.
• int width: Specify the width of the barcode.
• int alignment: Select the alignment of the barcode. (Refer to “5-5 Alignment”)
• int textPosition: Determine the postion of the text to be printed with the barcode. (Refer to “5-7 Barcode Text Location”)

[Example]

```
import com.bxl.config.editor.BXLConfigLoader;
import jpos.config.JposEntry;
import jpos.POSPrinter;
import jpos.POSPrinterConst;
import com.bxl.BXLConst;

try
{
    POSPrinter posPrinter = new POSPrinter(this);
    posPrinter.open(“SPP-R200III”);
    posPrinter.claim(5000);
    posPrinter.setDeviceEnabled(true);

    posPrinter.printBarCode(POSPrinterConst.PTR_S_RECEIPT,
                            “123456789”,
                            POSPrinterConst.PTR_BCS_QRCODE,
                            8,
                            8,
                            POSPrinterConst.PTR_BC_CENTER,
                            POSPrinterConst.PTR_BC_TEXT_BELOW);
}
catch(JposException e)
{
    e.printStackTrace();
}
```

6-2-12 Print Bitmap()

Prints image. (file printing)

[Syntax]
void printBitmap(int station, String fileName, int width, int alignment)
throws JposException

[Parameters]
• int station: Set image printing options (total of 4 bytes)

Index	Description
First byte	Fixed Value: PTR_S_RECEIPT
Second byte	brightness: 0 ~ 100
Third byte	Whether to use compression algorithm (0x00: None, 0x01: RLE, 0x02: LZMA)
Fourth byte	Whether to use dither 0x00: None, 0x01: FloydSteinberg

• String filename: Specify the path to the image file.
• int width: Specify the image width.
• int alignment: Select the image alignment. (Refer to “5-5 Alignment”)

[Example]

```
import com.bxl.config.editor.BXLConfigLoader;
import jpos.config.JposEntry;
import jpos.POSPrinter;
import jpos.POSPrinterConst;
import com.bxl.BXLConst;

try
{
	POSPrinter posPrinter = new POSPrinter(this);
	posPrinter.open("SPP-R200III");
	posPrinter.claim(5000);
	posPrinter.setDeviceEnabled(true);

	ByteBuffer buffer = ByteBuffer.allocate(4);
	buffer.put((byte) POSPrinterConst.PTR_S_RECEIPT);
	buffer.put((byte) 80); // brightness
	buffer.put((byte) 0x01); // compress
	buffer.put((byte) 0x00);

	posPrinter.printBitmap(buffer.getInt(0),
				imagePath,
				posPrinter.getRecLineWidth(),
				POSPrinterConst.PTR_BM_LEFT);
}
catch(JposException e)
{
	e.printStackTrace();
}
```

6-2-13 print Bitmap()

Prints image. (Bitmap data)

[Syntax]
void printBitmap(int station, Bitmap bitmap, int width, int alignment)
throws JposException

[Parameters]
• int station: Set image printing options (total of 4 bytes)

Index	Description
First byte	Fixed Value: PTR_S_RECEIPT
Second byte	brightness: 0 ~ 100
Third byte	Whether to use compression algorithm (0x00: None, 0x01: RLE, 0x02: LZMA)
Fourth byte	Whether to use dither 0x00: None, 0x01: FloydSteinberg

• Bitmap bitmap: Type the image data in the bitmap format.
• int width: Specify the image width.
• int alignment: Select the image alignment. (Refer to “5-5 Alignment”)

[Example]

```
import com.bxl.config.editor.BXLConfigLoader;
import jpos.config.JposEntry;
import jpos.POSPrinter;
import jpos.POSPrinterConst;
import com.bxl.BXLConst;

try
{
	POSPrinter posPrinter = new POSPrinter(this);
	posPrinter.open("SPP-R200III");
	posPrinter.claim(5000);
	posPrinter.setDeviceEnabled(true);

	ByteBuffer buffer = ByteBuffer.allocate(4);
	buffer.put((byte) POSPrinterConst.PTR_S_RECEIPT);
	buffer.put((byte) 80); // brightness
	buffer.put((byte) 0x01); // compress
	buffer.put((byte) 0x00);

	posPrinter.printBitmap(buffer.getInt(0),
				BitmapData,
				posPrinter.getRecLineWidth(),
				POSPrinterConst.PTR_BM_LEFT);
}
catch(JposException e)
{
	e.printStackTrace();
}
```

6-2-14 Print Bitmap()

Prints image. (Bitmap data)

[Syntax]
void printBitmap(int station, Bitmap bitmap, int width, int alignment,
int thresholdValue) throws
JposException

[Parameters]
• int station: Set image printing options (total of 4 bytes)

Index	Description
First byte	Fixed Value: PTR_S_RECEIPT
Second byte	brightness: 0 ~ 100
Third byte	Whether to use compression algorithm (0x00: None, 0x01: RLE, 0x02: LZMA)
Fourth byte	Whether to use dither 0x00 : None, 0x01 : FloydSteinberg

• Bitmap bitmap: Type the image data in the bitmap format.
• int width: Specify the image width.
• int alignment: Select the image alignment. (Refer to “5-5 Alignment”)
• int thresholdValue: Specifies the threshold to apply to each pixel. Only pixel values that are less then the threshold are printed. ($0 \leq$ thresholdValue ≤ 255)

[Example]

```
import com.bxl.config.editor.BXLConfigLoader;
import jpos.config.JposEntry;
import jpos.POSPrinter;
import jpos.POSPrinterConst;
import com.bxl.BXLConst;

try
{
    POSPrinter posPrinter = new POSPrinter(this);
    posPrinter.open("SPP-R200III");
    posPrinter.claim(5000);
    posPrinter.setDeviceEnabled(true);

    ByteBuffer buffer = ByteBuffer.allocate(4);
    buffer.put((byte) POSPrinterConst.PTR_S_RECEIPT);
    buffer.put((byte) 80); // brightness
    buffer.put((byte) 0x01); // compress
    buffer.put((byte) 0x00);

    posPrinter.printBitmap(buffer.getInt(0),
                           BitmapData,
                           posPrinter.getRecLineWidth(),
                           POSPrinterConst.PTR_BM_LEFT,
                           200);
}
catch(JposException e)
{
    e.printStackTrace();
}
```

6-2-15 Print Normal()

Prints text.

[Syntax]
void printNormal(int station, String data) throws JposException

[Parameters]
• int station: fixed Value PTR_S_RECEIPT
• String data: Specify the data to be printed. Printable characters and escape sequences, carriage returns, line feeds Data are allowed.

[Example]

```
import com.bxl.config.editor.BXLConfigLoader;
import jpos.config.JposEntry;
import jpos.POSPrinter;
import jpos.POSPrinterConst;
import com.bxl.BXLConst;

try
{
	POSPrinter posPrinter = new POSPrinter(this);
	posPrinter.open("SPP-R200III");
	posPrinter.claim(5000);
	posPrinter.setDeviceEnabled(true);

	posPrinter.printNormal(POSPrinterConst.PTR_S_RECEIPT, "Print Data\n");
}
catch(JposException e)
{
	e.printStackTrace();
}
```

6-2-16 Set Page Mode Print Area()

Specifies the page mode area.

[Syntax]
void setPageModePrintArea(String area) throws JposException

[Parameters]
• String area: Specifies the print area.
"X coordinate of area, y coordinate of area, width of area, height of area"
ex) "0, 0, 576, 1600"

[Example]

```
import com.bxl.config.editor.BXLConfigLoader;
import jpos.config.JposEntry;
import jpos.POSPrinter;
import jpos.POSPrinterConst;
import com.bxl.BXLConst;

try
{
	POSPrinter posPrinter = new POSPrinter(this);
	posPrinter.open("SPP-R200III");
	posPrinter.claim(5000);
	posPrinter.setDeviceEnabled(true);

	posPrinter.setPageModePrintArea("0, 0, 576, 1600");
}
catch(JposException e)
{
	e.printStackTrace();
}
```

6-2-17 Set Page Mode Print Direction()

Specifies the direction of printing in Page Mode.

[Syntax]
void setPageModePrintDirection(int direction) throws JposException

[Parameters]
• int direction: Specify the direction of printing.
(Refer to “Print Direction in Page Mode”)

[Example]

```
import com.bxl.config.editor.BXLConfigLoader;
import jpos.config.JposEntry;
import jpos.POSPrinter;
import jpos.POSPrinterConst;
import com.bxl.BXLConst;

try
{
	POSPrinter posPrinter = new POSPrinter(this);
	posPrinter.open(“SPP-R200III”);
	posPrinter.claim(5000);
	posPrinter.setDeviceEnabled(true);

	posPrinter.setPageModePrintArea(“0, 0, 576, 1600”);
	posPrinter.setPageModePrintDirection(POSPrinterConst.PTR_PD_LEFT_TO_RIGHT);
}
catch(JposException e)
{
	e.printStackTrace();
}
```

6-2-18 Page Mode Print()

Sets the printer to page mode.

[Syntax]
void pageModePrint(int control) throws JposException

[Parameters]
• int control: Specify the page mode

Code	Value	Description
PTR_PM_PAGE_MODE	1	Enable page mode
PTR_PM_NORMAL	2	Change to the normal mode and the data stored in the page mode buffer is printed.

[Example]

```
import com.bxl.config.editor.BXLConfigLoader;
import jpos.config.JposEntry;
import jpos.POSPrinter;
import jpos.POSPrinterConst;
import com.bxl.BXLConst;

try
{
	POSPrinter posPrinter = new POSPrinter(this);
	posPrinter.open("SPP-R200III");
	posPrinter.claim(5000);
	posPrinter.setDeviceEnabled(true);

	posPrinter.setPageModePrintArea("0, 0, 576, 1600");
	posPrinter.setPageModePrintDirection(POSPrinterConst.PTR_PD_LEFT_TO_RIGHT);
	posPrinter.setPageModePrint(POSPrinterConst.PTR_PM_PAGE_MODE);
}
catch(JposException e)
{
	e.printStackTrace();
}
```

6-2-19 Set Page Mode Horizontal Position()

Specifies the print start position (Horizontal).

[Syntax]
void setPageModeHorizontalPosition(int position) throws JposException

[Parameters]
• int position: print start position (Horizontal)

[Example]

```
import com.bxl.config.editor.BXLConfigLoader;
import jpos.config.JposEntry;
import jpos.POSPrinter;
import jpos.POSPrinterConst;
import com.bxl.BXLConst;

try
{
    POSPrinter posPrinter = new POSPrinter(this);
    posPrinter.open("SPP-R200III");
    posPrinter.claim(5000);
    posPrinter.setDeviceEnabled(true);

    ByteBuffer buffer = ByteBuffer.allocate(4);
    buffer.put((byte) POSPrinterConst.PTR_S_RECEIPT);
    buffer.put((byte) 80); // brightness
    buffer.put((byte) 0x01); // compress
    buffer.put((byte) 0x00);

    // specifying of print area
    posPrinter.setPageModePrintArea("0, 0, 576, 1600");
    // specifying of print direction
    posPrinter.setPageModePrintDirection(POSPrinterConst.PTR_PD_LEFT_TO_RIGHT);
    // change to printer page mode
    posPrinter.setPageModePrint(POSPrinterConst.PTR_PM_PAGE_MODE);

    // specifying of width/height print position
    posPrinter.setPageModeHorizontalPosition(0);
    posPrinter.setPageModeVerticalPosition(0);
    // transmitting of data to be printed (image)
    posPrinter.printBitmap(buffer.getInt(0), imagePath, 384, POSPrinterConst.PTR_BM_LEFT);

    // specifying of width/height position
    posPrinter.setPageModeHorizontalPosition(100);
    posPrinter.setPageModeVerticalPosition(100);
    // transmitting of data to be printed (text)
    posPrinter.printNormar(POSPrinterConst.PTR_S_RECEIPT, "Print Data\n");

    // print start
    posPrinter.setPageModePrint(POSPrinterConst.PTR_PM_PAGE_NORMAL);
```

```
}
catch(JposException e)
{
	e.printStackTrace();
}
```

6-2-20 Set Page Mode Vertical Position()

Specifies the print start position (Vertical).

[Syntax]
void setPageModeVerticalPosition(int position) throws JposException

[Parameters]
• int position: print start position (Vertical)

[Example]

```
import com.bxl.config.editor.BXLConfigLoader;
import jpos.config.JposEntry;
import jpos.POSPrinter;
import jpos.POSPrinterConst;
import com.bxl.BXLConst;

try
{
    POSPrinter posPrinter = new POSPrinter(this);
    posPrinter.open("SPP-R200III");
    posPrinter.claim(5000);
    posPrinter.setDeviceEnabled(true);

    ByteBuffer buffer = ByteBuffer.allocate(4);
    buffer.put((byte) POSPrinterConst.PTR_S_RECEIPT);
    buffer.put((byte) 80); // brightness
    buffer.put((byte) 0x01); // compress
    buffer.put((byte) 0x00);

    // specifying of print area
    posPrinter.setPageModePrintArea("0, 0, 576, 1600");
    // specifying of print direction
    posPrinter.setPageModePrintDirection(POSPrinterConst.PTR_PD_LEFT_TO_RIGHT);
    // change to printer page mode
    posPrinter.setPageModePrint(POSPrinterConst.PTR_PM_PAGE_MODE);

    // specifying of width/height print position
    posPrinter.setPageModeHorizontalPosition(0);
    posPrinter.setPageModeVerticalPosition(0);
    // transmitting of data to be printed (image)
    posPrinter.printBitmap(buffer.getInt(0), imagePath, 384, POSPrinterConst.PTR_BM_LEFT);

    // specifying of width/height position
    posPrinter.setPageModeHorizontalPosition(100);
    posPrinter.setPageModeVerticalPosition(100);
    // transmitting of data to be printed (text)
    posPrinter.printNormar(POSPrinterConst.PTR_S_RECEIPT, "Print Data\n");
```

```
        // print start
        posPrinter.setPageModePrint(POSPrinterConst.PTR_PM_PAGE_NORMAL);
}
catch(JposException e)
{
        e.printStackTrace();
}
```

6-2-21 Transaction Print()

Prints using the Transaction Mode.

[Syntax]
public void transactionPrint(int station, int control) throws JposException

[Parameters]
• int station: fixed Value PTR_S_RECEIPT
• int control: Transaction Mode(Refer to “5-4 Transaction Print”)

[Example]

```
import com.bxl.config.editor.BXLConfigLoader;
import jpos.config.JposEntry;
import jpos.POSPrinter;
import jpos.POSPrinterConst;
import com.bxl.BXLConst;

try
{
	POSPrinter posPrinter = new POSPrinter(this);
	posPrinter.open(“SPP-R200III”);
	posPrinter.claim(5000);
	posPrinter.setDeviceEnabled(true);

	// Transaction mode start
	posPrinter.transactionPrint(POSPrinterConst.PTR_TP_TRANSACTION);

	// transmitting of data to be printed (image)
	posPrinter.printBitmap(buffer.getInt(0), imagePath, 384, POSPrinterConst.PTR_BM_LEFT);
	// transmitting of data to be printed (text)
	posPrinter.printNormar(POSPrinterConst.PTR_S_RECEIPT, “Print Data\n”);

	// Transaction mode Termination (Start printing)
	posPrinter.transactionPrint(POSPrinterConst.PTR_TP_NORMAL);
}
catch(JposException e)
{
	e.printStackTrace();
}
```

6-2-22 Display String()

Print the text with BCD-3000.
This time, BCD-3000 should be connected to SRP-Q300 DK port.

[Syntax]
public void displayString(String data) throws JposException

[Parameters]
• String data: Text data to be printed with BCD-3000.

[Example]

```
import com.bxl.config.editor.BXLConfigLoader;
import jpos.config.JposEntry;
import jpos.POSPrinter;
import jpos.POSPrinterConst;
import com.bxl.BXLConst;

try
{
	POSPrinter posPrinter = new POSPrinter(this);
	posPrinter.open("SRP-Q300");
	posPrinter.claim(5000);
	posPrinter.setDeviceEnabled(true);

	posPrinter.displayString("BIXOLON Customer Display 3000");
}
catch(JposException e)
{
	e.printStackTrace();
}
```

6-2-23 Clean Screen()

Clean BCD-3000 screen.
This time, BCD-3000 should be connected to SRP-Q300 DK port.

[Syntax]
public void cleanScreen() throws JposException

[Example]

```
import com.bxl.config.editor.BXLConfigLoader;
import jpos.config.JposEntry;
import jpos.POSPrinter;
import jpos.POSPrinterConst;
import com.bxl.BXLConst;

try
{
	POSPrinter posPrinter = new POSPrinter(this);
	posPrinter.open("SRP-Q300");
	posPrinter.claim(5000);
	posPrinter.setDeviceEnabled(true);

	posPrinter.displayString("BIXOLON Customer Display 3000");
	posPrinter.cleanScreen();
}
catch(JposException e)
{
	e.printStackTrace();
}
```

6-2-24 Store Image File()

Save the image data in the image buffer of BCD-3000.
This time, BCD-3000 should be connected to SRP-Q300 DK port.

[Syntax]
public void storeImageFile(String filename, int width, int height, int imageNumber) throws JposException

[Parameters]
• String filename: Specify the path of image file.
• int width: Specify the width of image (1 ~ 160)
• int height: Specify the height of image (1 ~ 32)
• int imageNumber: Specify the number of the image data to be saved (1 ~ 5)

[Example]

```
import com.bxl.config.editor.BXLConfigLoader;
import jpos.config.JposEntry;
import jpos.POSPrinter;
import jpos.POSPrinterConst;
import com.bxl.BXLConst;

try
{
	POSPrinter posPrinter = new POSPrinter(this);
	posPrinter.open("SRP-Q300");
	posPrinter.claim(5000);
	posPrinter.setDeviceEnabled(true);

	posPrinter.storeImageFile(imagePath, 160, 32, 1);
}
catch(JposException e)
{
	e.printStackTrace();
}
```

6-2-25 Display Image()

Print the image saved in the image buffer of BCD-3000.
This time, BCD-3000 should be connected to SRP-Q300 DK port.

[Syntax]
public void displayImage(int imageNumber, int xPos, int yPos) throws JposException

[Parameters]
• int imageNumber: Specify the number of the image data to be printed (1 ~ 5)
• int xPos: Input X coordinate to print the image on (0 ~ 159)
• int yPos: Input Y coordinate to print the image on (0 ~ 31)
※ The image will not be displayed when it is out of the printing area.

[Example]

```
import com.bxl.config.editor.BXLConfigLoader;
import jpos.config.JposEntry;
import jpos.POSPrinter;
import jpos.POSPrinterConst;
import com.bxl.BXLConst;

try
{
	POSPrinter posPrinter = new POSPrinter(this);
	posPrinter.open("SRP-Q300");
	posPrinter.claim(5000);
	posPrinter.setDeviceEnabled(true);

	posPrinter.storeImageFile(imagePath, 160, 32, 1);
	posPrinter.displayImage(1, 0, 0);
}
catch(JposException e)
{
	e.printStackTrace();
}
```

6-2-26 Clear Image()

Delete the image saved in the image buffer of BCD-3000.
This time, BCD-3000 should be connected to SRP-Q300 DK port.

[**Syntax**]
public void clearImage(boolean isAll, int imageNumber) throws JposException

[**Parameters**]
• boolean isAll: Select image buffer clear mode.

Value	Description
true	Delete all the images in image buffer.
false	Delete only the image of the specified number. The image number to be deleted should be input in imageNumber

• int imageNumber: Specify the number of the image data to be deleted (1 ~ 5)

[Example]

```
import com.bxl.config.editor.BXLConfigLoader;
import jpos.config.JposEntry;
import jpos.POSPrinter;
import jpos.POSPrinterConst;
import com.bxl.BXLConst;

try
{
	POSPrinter posPrinter = new POSPrinter(this);
	posPrinter.open("SRP-Q300");
	posPrinter.claim(5000);
	posPrinter.setDeviceEnabled(true);

	posPrinter.storeImageFile(imagePath, 160, 32, 1);
	posPrinter.displayImage(1, 0, 0);
	posPrinter.clearImage(false, 1);
}
catch(JposException e)
{
	e.printStackTrace();
}
```

6-2-27 DirectIO()

Transmits custom data to the printer.
Transfers the data-driven response value to the directIOOccurred Event.

[Syntax]
public void directIO(int command, int[] data, Object object) throws JposException

[Parameters]
• int command: Selects the type of direct I/O command.

Value	Description
1	Transmit custom data.
2	Check battery status.
3	Power off time (mobile printer only): 0 to 90 (seconds)

• int[] data: Specifies the power off time (command = 3)
• Object object: Enters a custom command.

[Example]

```
import com.bxl.config.editor.BXLConfigLoader;
import jpos.config.JposEntry;
import jpos.POSPrinter;
import jpos.POSPrinterConst;
import com.bxl.BXLConst;

try
{
    POSPrinter posPrinter = new POSPrinter(this);
    posPrinter.addDirectIOListener(this);
    posPrinter.open("SRP-Q300");
    posPrinter.claim(5000);
    posPrinter.setDeviceEnabled(true);

    byte[] data = new byte{0x1d, 0x49, 0x43};
    posPrinter.directIO(1, null, data);
}
catch(JposException e)
{
    e.printStackTrace();
}

@Override
public void directIOOccurred(DirectIOEvent directIOEvent) {
    runOnUiThread(new Runnable()
    {
        @Override
        public void run()
        {
            Toast.makeText(MainActivity.this, new String((byte[]) directIOEvent.getObject(),
    Toast.LENGTH_SHORT).show();
```

```
            }
        });
}
```

6-2-28 Mark Feed()

Feeds the paper to the next print position.

[Syntax]
public void markFeed(int type) throws JposException

[Parameters]
• int type: Specifies the mark type (fixed value: 0)

[Example]

```
import com.bxl.config.editor.BXLConfigLoader;
import jpos.config.JposEntry;
import jpos.POSPrinter;
import jpos.POSPrinterConst;
import com.bxl.BXLConst;

try
{
	POSPrinter posPrinter = new POSPrinter(this);
	posPrinter.open("SPP-R310");
	posPrinter.claim(5000);
	posPrinter.setDeviceEnabled(true);

	posPrinter.markFeed(0);
}
catch(JposException e)
{
	e.printStackTrace();
}
```

6-2-29 Set Bitmap()

This method saves the image in the non-volatile memory area.

[Syntax]
void setBitmap(int bitmapNumber, int station, String fileName, int width,
int alignment) throws JposException

[Parameters]
• int bitmapNumber: address of memory area to save image. (0 ~ 255)
• int station: Set image printing options (total of 4 bytes)

Index	Description
First byte	Fixed Value: PTR_S_RECEIPT
Second byte	brightness: 0 ~ 100
Third byte	Reserved(0x00)
Fourth byte	Reserved(0x00)

• String filename: Specify the path to the image file.
• int width: Specify the image width.
• int alignment: Fixed Value: -1

[Example]

```
import com.bxl.config.editor.BXLConfigLoader;
import jpos.config.JposEntry;
import jpos.POSPrinter;
import jpos.POSPrinterConst;
import com.bxl.BXLConst;

try
{
    POSPrinter posPrinter = new POSPrinter(this);
    posPrinter.open("SPP-R200III");
    posPrinter.claim(5000);
    posPrinter.setDeviceEnabled(true);

    ByteBuffer buffer = ByteBuffer.allocate(4);
    buffer.put((byte) POSPrinterConst.PTR_S_RECEIPT);
    buffer.put((byte) 80); // brightness
    buffer.put((byte) 0x00); // Reserved
    buffer.put((byte) 0x00); // Reserved

    // Define NV image : Memory code 0
    posPrinter.setBitmap(0,
                         buffer.getInt(0),
                         imagePath,
                         posPrinter.getRecLineWidth(),
                         -1);

    // Print NV image : Memory code 0
    String data = ESCAPE_SEQUENCE + "0B";
```

```
        posPrinter.printNormal(POSPrinterConst.PTR_S_RECEIPT, data);
}
catch(JposException e)
{
        e.printStackTrace();
}
```

6-2-30 Print Svg()

Prints SVG file.

[Syntax]
void printSvg(int station, String fileName, int width, int alignment, int rotate)
throws JposException

[Parameters]
• int station: Set SVG printing options (total of 4 bytes)

Index	Description
First byte	Fixed Value: PTR_S_RECEIPT
Second byte	brightness: 0 ~ 100
Third byte	Whether to use compression algorithm (RLE) (0x01: RLE)
Fourth byte	Reserved(0x00)

• String filename: Specify the path to the SVG file.
• int width: Specify the SVG width.
• int alignment: Select the SVG alignment. (Refer to “5-5 Alignment”)
• int rotate: Select the SVG rotate.

Value	Description
0	No rotate
90	90° rotate
180	180° rotate
270	270° rotate

[Example]

```
import com.bxl.config.editor.BXLConfigLoader;
import jpos.config.JposEntry;
import jpos.POSPrinter;
import jpos.POSPrinterConst;
import com.bxl.BXLConst;

try
{
    POSPrinter posPrinter = new POSPrinter(this);
    posPrinter.open(“SPP-R410”);
    posPrinter.claim(5000);
    posPrinter.setDeviceEnabled(true);

    ByteBuffer buffer = ByteBuffer.allocate(4);
    buffer.put((byte) POSPrinterConst.PTR_S_RECEIPT);
    buffer.put((byte) 50); // brightness
    buffer.put((byte) 0x01); // compress
    buffer.put((byte) 0x00);

    posPrinter.printSvg(buffer.getInt(0),
                        imagePath,
```

```
                                posPrinter.getRecLineWidth(),
                                POSPrinterConst.PTR_BM_LEFT,
                                90);
}
catch(JposException e)
{
      e.printStackTrace();
}
```

6-2-31 Update Frimware()

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]
void updateFirmware(String firmwareFileName) throws JposException

[Parameters]
• String firmwareFileName: Absolute path fo the firmware binary file to send to printer.

[Example]

```
import com.bxl.config.editor.BXLConfigLoader;
import jpos.config.JposEntry;
import jpos.POSPrinter;
import jpos.POSPrinterConst;
import com.bxl.BXLConst;

try
{
	POSPrinter posPrinter = new POSPrinter(this);
	posPrinter.open("SPP-R200III");
	posPrinter.claim(5000);
	posPrinter.setDeviceEnabled(true);

	posPrinter.updateFirmware("fw.fls");
}
catch(JposException e)
{
	e.printStackTrace();
}
```

6-2-32 Eject Paper()

Ejects paper.

Available with the BK3-3 and BK5-3 Presenter Accessory model only.

[Syntax]
void ejectPaper(int mode) throws JposException

[Parameters]
• int mode: sets the paper ejection mode.

Value	Description
3	Automatically calculates the length and ejects the paper. Then, paper is detected by the Presenter out sensor.
6	Completely ejects the paper from the Presenter sensor. Once ejected, paper is not detected by the Presenter out sensor.

[Example]

```
import com.bxl.config.editor.BXLConfigLoader;
import jpos.config.JposEntry;
import jpos.POSPrinter;
import jpos.POSPrinterConst;
import com.bxl.BXLConst;

try
{
	POSPrinter posPrinter = new POSPrinter(this);
	posPrinter.open("BK3-3");
	posPrinter.claim(5000);
	posPrinter.setDeviceEnabled(true);

	posPrinter. ejectPaper(3);
}
catch(JposException e)
{
	e.printStackTrace();
}
```

6-2-33 Get Presenter Status()

Checks the status of the Presenter.

Available with the BK3-3 and BK5-3 Presenter Accessory model only.

[**Syntax**]
byte getPresenterStatus() throws JposException

[**Return Values**]
• Presenter status: 1 byte

Value	Description
-1	Failed to check the status
Other values	Presenter sensor status (Refer to the Table [Presenter Status])

Bit	Hex	Description
0	00	Paper near-end sensor: Paper adequate.
	01	Paper near-end sensor: Paper near end.
1	00	Not used. Fixed to Off
2	00	Paper end sensor: paper present
	04	Paper end sensor: paper not present
3	00	Presenter out sensor: paper out
	08	Presenter out sensor: paper in
4	00	Not used. Fixed to Off
5	00	Not used. Fixed to Off
6	00	Not used. Fixed to Off
7	00	No jam
	80	Paper jam

[Presenter status]

[Example]

```
import com.bxl.config.editor.BXLConfigLoader;
import jpos.config.JposEntry;
import jpos.POSPrinter;
import jpos.POSPrinterConst;
import com.bxl.BXLConst;

try
{
	POSPrinter posPrinter = new POSPrinter(this);
	posPrinter.open("BK3-3");
	posPrinter.claim(5000);
	posPrinter.setDeviceEnabled(true);

	byte[] status = posPrinter. ejectPaper(3);
	if(status != -1) {
		// status
	}
}
catch(JposException e) {
	e.printStackTrace();
}
```

6-2-34 Print PDF File()

Print the PDF file.

[Syntax]
void printPDFFile(int station, Uri uri, int width, int alignment, int pgae)
throws JposException

[Parameters]
• int station: Set the image print options (consisting of 4 bytes in total)

Index	Explanation
First byte	Fixed value: PTR_S_RECEIPT
Second byte	Brightness: 0 ~ 100
Third byte	Whether to use a compression algorithm 0x00: no compression, 0x01: RLE, 2: LZMA
Fourth byte	Whether to use dither 0x00: not used, 0x01: FloydSteinberg

• Uri uri: Enter the PDF file path in URI format.
• int width: Specifies the image width.
• int alignment: Choose image alignment. (“5-5 Alignment” Reference)
• int page: Specifies the PDF page number to be printed.

[Example]

```
import com.bxl.config.editor.BXLConfigLoader;
import jpos.config.JposEntry;
import jpos.POSPrinter;
import jpos.POSPrinterConst;
import com.bxl.BXLConst;

try
{
	POSPrinter posPrinter = new POSPrinter(this);
	posPrinter.open("SPP-R200III");
	posPrinter.claim(5000);
	posPrinter.setDeviceEnabled(true);

	ByteBuffer buffer = ByteBuffer.allocate(4);
	buffer.put((byte) POSPrinterConst.PTR_S_RECEIPT);
	buffer.put((byte) 80); // brightness
	buffer.put((byte) 0x01); // compress
	buffer.put((byte) 0x00);


	Uri uri = Uri.parse("/document/primary:Download/PDF_For_Print.pdf");
	posPrinter.printPDFFile(buffer.getInt(0),
					uri,
					posPrinter.getRecLineWidth(),
					POSPrinterConst.PTR_BM_LEFT,
					1);
}
catch(JposException e)
{
	e.printStackTrace();
}
```

6-2-35 Print PDF File()

Print the PDF file.

[Syntax]
void printPDFFile(int station, Uri uri, int width, int alignment, int startPgae, int endPage) throws JposException

[Parameters]
• int station: Set the image print options (consisting of 4 bytes in total)

Index	Explanation
First byte	Fixed value: PTR_S_RECEIPT
Second byte	Brightness: 0 ~ 100
Third byte	Whether to use a compression algorithm 0x00: no compression, 0x01: RLE, 2: LZMA
Fourth byte	Whether to use dither 0x00: not used, 0x01: FloydSteinberg

• Uri uri: Enter the PDF file path in URI format.
• int width: Specifies the image width.
• int alignment: Choose image alignment. (“5-5 Alignment” Reference)
• int startPage: Specifies the page number from which to start printing.
• int endPage: Specifies the page number to end printing.

[Example]

```
import com.bxl.config.editor.BXLConfigLoader;
import jpos.config.JposEntry;
import jpos.POSPrinter;
import jpos.POSPrinterConst;
import com.bxl.BXLConst;

try
{
	POSPrinter posPrinter = new POSPrinter(this);
	posPrinter.open("SPP-R200III");
	posPrinter.claim(5000);
	posPrinter.setDeviceEnabled(true);

	ByteBuffer buffer = ByteBuffer.allocate(4);
	buffer.put((byte) POSPrinterConst.PTR_S_RECEIPT);
	buffer.put((byte) 80); // brightness
	buffer.put((byte) 0x01); // compress
	buffer.put((byte) 0x00);


	Uri uri = Uri.parse("/document/primary:Download/PDF_For_Print.pdf");
	posPrinter.printPDFFile(buffer.getInt(0),
					uri,
					posPrinter.getRecLineWidth(),
					POSPrinterConst.PTR_BM_LEFT,
					1,
					3);
}
catch(JposException e)
{
	e.printStackTrace();
}
```

6-2-36 Print PDF File()

Print the PDF file.

[Syntax]
void printPDFFile(int station, Uri uri, int width, int alignment, int startPgae,
int endPage, int thresholdValue) throws JposException

[Parameters]
• int station: Set the image print options (consisting of 4 bytes in total)

Index	Explanation
First byte	Fixed value: PTR_S_RECEIPT
Second byte	Brightness: 0 ~ 100
Third byte	Whether to use a compression algorithm 0x00: no compression, 0x01: RLE, 2 : LZMA
Fourth byte	Whether to use dither 0x00: not used, 0x01 : FloydSteinberg

• Uri uri: Enter the PDF file path in URI format.
• int width: Specifies the image width.
• int alignment: Choose image alignment. (“5-5 Alignment” Reference)
• int startPage: Specifies the page number from which to start printing.
• int endPage: Specifies the page number to end printing.
• int thresholdValue: Specifies the threshold to apply to each pixel. Only pixel values that are less then the threshold are printed. ($0 \leq$ thresholdValue ≤ 255)

[Example]

```
import com.bxl.config.editor.BXLConfigLoader;
import jpos.config.JposEntry;
import jpos.POSPrinter;
import jpos.POSPrinterConst;
import com.bxl.BXLConst;

try
{
	POSPrinter posPrinter = new POSPrinter(this);
	posPrinter.open("SPP-R200III");
	posPrinter.claim(5000);
	posPrinter.setDeviceEnabled(true);

	ByteBuffer buffer = ByteBuffer.allocate(4);
	buffer.put((byte) POSPrinterConst.PTR_S_RECEIPT);
	buffer.put((byte) 80); // brightness
	buffer.put((byte) 0x01); // compress
	buffer.put((byte) 0x00);


	Uri uri = Uri.parse("/document/primary:Download/PDF_For_Print.pdf");
	posPrinter.printPDFFile(buffer.getInt(0),
					uri,
					posPrinter.getRecLineWidth(),
					POSPrinterConst.PTR_BM_LEFT,
					1,
					3,
					200);
}
catch(JposException e)
{
	e.printStackTrace();
}
```

6-2-37 Set Alarm()

Set an alarm on the printer.

[Syntax]
void setAlarm (int count) throws JposException

[Parameters]
• int count: Set the number of alarms.

[Example]

```
import com.bxl.config.editor.BXLConfigLoader;
import jpos.config.JposEntry;
import jpos.POSPrinter;
import jpos.POSPrinterConst;
import com.bxl.BXLConst;

try
{
	POSPrinter posPrinter = new POSPrinter(this);
	posPrinter.open("SPP-R200III");
	posPrinter.claim(5000);
	posPrinter.setDeviceEnabled(true);

	posPrinter.printNormal(POSPrinterConst.PTR_S_RECEIPT, "Print Data\n");
	posPrinter.cutPaper(100);
	posPrinter.setAlarm(3);

}
catch(JposException e)
{
	e.printStackTrace();
}
```

6-2-38 Search USB Peripheral Devices ()

Loading the list of peripheral devices connected to the printer.

Only available on models with USB peripheral controls.

[Syntax]
byte[] searchUsbPeripheralDevices (boolean enable) throws JposException

[Parameters]
• boolean enable: Enable plug-and-play.

[Example]

```
import com.bxl.config.editor.BXLConfigLoader;
import jpos.config.JposEntry;
import jpos.POSPrinter;
import jpos.POSPrinterConst;
import com.bxl.BXLConst;

try
{
	POSPrinter posPrinter = new POSPrinter(this);
	posPrinter.open("SRP-Q300II");
	posPrinter.claim(5000);
	posPrinter.setDeviceEnabled(true);

	devices = posPrinter.searchUsbPeripheralDevices(true);

}
catch(JposException e)
{
	e.printStackTrace();
}
```

6-2-39 Send USB Peripheral Data()

Send data to the peripheral connected to the printer.

Only available on models with USB peripheral controls.

[Syntax]
void sendUsbPeripheralData(int deviceID, byte[] data, int dataLen)
throws JposException

[Parameters]
• int deviceID: Peripheral ID
• byte[] data: Data to send
• int dataLen: Length of data to send

[Example]

```
import com.bxl.config.editor.BXLConfigLoader;
import jpos.config.JposEntry;
import jpos.POSPrinter;
import jpos.POSPrinterConst;
import com.bxl.BXLConst;

try
{
	POSPrinter posPrinter = new POSPrinter(this);
	posPrinter.open("SRP-Q300II");
	posPrinter.claim(5000);
	posPrinter.setDeviceEnabled(true);

	posPrinter.sendUsbPeripheralData(deviceID, data, data.length);

}
catch(JposException e)
{
	e.printStackTrace();
}
```

6-2-40 Read USB Peripheral Data()

Receive data from the peripheral connected to the printer.

Only available on models with USB peripheral controls.

[Syntax]
byte[] readUsbPeripheralData(int deviceID, boolean enable) throws JposException

[Parameters]
• int deviceID: Peripheral ID
• boolean enable: Receive/Don't receive data

[Example]

```
import com.bxl.config.editor.BXLConfigLoader;
import jpos.config.JposEntry;
import jpos.POSPrinter;
import jpos.POSPrinterConst;
import com.bxl.BXLConst;

try
{
	POSPrinter posPrinter = new POSPrinter(this);
	posPrinter.open("SRP-Q300II");
	posPrinter.claim(5000);
	posPrinter.setDeviceEnabled(true);

	posPrinter.readUsbPeripheralData(deviceID, true);

}
catch(JposException e)
{
	e.printStackTrace();
}
```

6-2-41 Set Serial Setting Data()

Change the serial settings of the peripheral connected to the printer.

Only available on models with USB peripheral controls.

[Syntax]
void setSerialSettingData(int deviceID, int baudRate, int dataBits, int stopBits, int parityBits, int flowControl) throws JposException

[Parameters]
• int deviceID: Peripheral ID
• int baudRate: Peripheral Baud Rate
• int dataBits: Peripheral Data Bits
• int stopBits: Peripheral Stop Bits
• int parityBits: Peripheral Parity Bits
• int flowControl: Peripheral Flow Control

[Example]

```
import com.bxl.config.editor.BXLConfigLoader;
import jpos.config.JposEntry;
import jpos.POSPrinter;
import jpos.POSPrinterConst;
import com.bxl.BXLConst;

try
{
	POSPrinter posPrinter = new POSPrinter(this);
	posPrinter.open("SRP-Q300II");
	posPrinter.claim(5000);
	posPrinter.setDeviceEnabled(true);

	posPrinter.setSerialSettingData(deviceID, baudRate, dataBits, stopBits, parityBits, flowControl);

}
catch(JposException e)
{
	e.printStackTrace();
}
```

6-2-42 Get Serial Setting Data()

Load the serial settings of the peripheral connected to the printer.

Only available on models with USB peripheral controls.

[Syntax]
void getSerialSettingData(int deviceID) throws JposException

[Parameters]
• int deviceID: Peripheral ID

[Example]

```
import com.bxl.config.editor.BXLConfigLoader;
import jpos.config.JposEntry;
import jpos.POSPrinter;
import jpos.POSPrinterConst;
import com.bxl.BXLConst;

try
{
	POSPrinter posPrinter = new POSPrinter(this);
	posPrinter.open("SRP-Q300II");
	posPrinter.claim(5000);
	posPrinter.setDeviceEnabled(true);

	data = posPrinter.getSerialSettingData(deviceID);

}
catch(JposException e)
{
	e.printStackTrace();
}
```

6-2-43 Get Vid Pid()

Load the Vendor ID and Product ID of the peripheral connected to the printer.

Only available on models with USB peripheral controls.

[Syntax]
void getVidPid(int deviceID) throws JposException

[Parameters]
• int deviceID: Peripheral ID

[Example]

```
import com.bxl.config.editor.BXLConfigLoader;
import jpos.config.JposEntry;
import jpos.POSPrinter;
import jpos.POSPrinterConst;
import com.bxl.BXLConst;

try
{
	POSPrinter posPrinter = new POSPrinter(this);
	posPrinter.open("SRP-Q300II");
	posPrinter.claim(5000);
	posPrinter.setDeviceEnabled(true);

	data = posPrinter.getSerialSettingData(deviceID);

}
catch(JposException e)
{
	e.printStackTrace();
}
```

6-2-44 Get Serial Number()

Load the serial number of the peripheral connected to the printer.

Only available on models with USB peripheral controls.

[Syntax]
void getSerialNumber (int deviceID) throws JposException

[Parameters]
• int deviceID: Peripheral ID

[Example]

```
import com.bxl.config.editor.BXLConfigLoader;
import jpos.config.JposEntry;
import jpos.POSPrinter;
import jpos.POSPrinterConst;
import com.bxl.BXLConst;

try
{
	POSPrinter posPrinter = new POSPrinter(this);
	posPrinter.open("SRP-Q300II");
	posPrinter.claim(5000);
	posPrinter.setDeviceEnabled(true);

	data = posPrinter. getSerialNumber(deviceID);

}
catch(JposException e)
{
	e.printStackTrace();
}
```

6-2-45 Get Custom USB Lists()

Load a list of user-defined USB devices among the peripherals connected to the printer.

Only available on models with USB peripheral controls.

[Syntax]
void getCustomUsbLists (int deviceID) throws JposException

[Parameters]
• int deviceID: Peripheral ID

[Example]

```
import com.bxl.config.editor.BXLConfigLoader;
import jpos.config.JposEntry;
import jpos.POSPrinter;
import jpos.POSPrinterConst;
import com.bxl.BXLConst;

try
{
	POSPrinter posPrinter = new POSPrinter(this);
	posPrinter.open("SRP-Q300II");
	posPrinter.claim(5000);
	posPrinter.setDeviceEnabled(true);

	data = posPrinter.getCustomUsbLists (deviceID);

}
catch(JposException e)
{
	e.printStackTrace();
}
```

6-2-46 Add Custom USB Device()

Set a device as a user-defined type among the peripherals connected to the printer.

Only available on models with USB peripheral controls.

[Syntax]
void addCustomUsbDevice(int deviceType, byte vidL, byte vidH, byte pidL, byte pidH) throws JposException

[Parameters]
• int deviceType: Peripheral Type
• int vidL: VID Low 4-bit Value
• int vidH: VID High 4-bit Value
• int pidL: PID Low 4-bit Value
• int pidH: PID High 4-bit Value

[Example]

```
import com.bxl.config.editor.BXLConfigLoader;
import jpos.config.JposEntry;
import jpos.POSPrinter;
import jpos.POSPrinterConst;
import com.bxl.BXLConst;

try
{
	POSPrinter posPrinter = new POSPrinter(this);
	posPrinter.open("SRP-Q300II");
	posPrinter.claim(5000);
	posPrinter.setDeviceEnabled(true);

	posPrinter.addCustomUsbDevice(deviceType, vidL, vidH, pidL, pidH);

}
catch(JposException e)
{
	e.printStackTrace();
}
```

6-2-47 Del Custom USB Device()

Delete the user-specified type of device from the peripherals connected to the printer.

Only available on models with USB peripheral controls.

[Syntax]
void delCustomUsbDevice(int deviceType, byte vidL, byte vidH, byte pidL, byte pidH) throws JposException

[Parameters]
• int deviceType: Peripheral Type
• int vidL: VID Low 4-bit Value
• int vidH: VID High 4-bit Value
• int pidL: PID Low 4-bit Value
• int pidH: PID High 4-bit Value

[Example]

```
import com.bxl.config.editor.BXLConfigLoader;
import jpos.config.JposEntry;
import jpos.POSPrinter;
import jpos.POSPrinterConst;
import com.bxl.BXLConst;

try
{
	POSPrinter posPrinter = new POSPrinter(this);
	posPrinter.open("SRP-Q300II");
	posPrinter.claim(5000);
	posPrinter.setDeviceEnabled(true);

	posPrinter.delCustomUsbDevice(deviceType, vidL, vidH, pidL, pidH);
}
catch(JposException e)
{
	e.printStackTrace();
}
```

6-2-48 Reset Custom USB Device()

Reset the user-specified type of peripherals connected to the printer.

Only available on models with USB peripheral controls.

[Syntax]
void resetCustomUsbDevice (int deviceType) throws JposException

[Parameters]
• int deviceType: Peripheral Type

[Example]

```
import com.bxl.config.editor.BXLConfigLoader;
import jpos.config.JposEntry;
import jpos.POSPrinter;
import jpos.POSPrinterConst;
import com.bxl.BXLConst;

try
{
	POSPrinter posPrinter = new POSPrinter(this);
	posPrinter.open("SRP-Q300II");
	posPrinter.claim(5000);
	posPrinter.setDeviceEnabled(true);

	posPrinter.resetCustomUsbDevice(deviceType);;
}
catch(JposException e)
{
	e.printStackTrace();
}
```

6-2-49 Retract Paper()

Reclaim paper.

Available on BK3R-3, BK5R-3 Presenter Accessory models only.

[Syntax]
void retractPaper() throws JposException

[Example]

```
import com.bxl.config.editor.BXLConfigLoader;
import jpos.config.JposEntry;
import jpos.POSPrinter;
import jpos.POSPrinterConst;
import com.bxl.BXLConst;

try
{
	POSPrinter posPrinter = new POSPrinter(this);
	posPrinter.open("BK3-3");
	posPrinter.claim(5000);
	posPrinter.setDeviceEnabled(true);

	posPrinter. retractPaper();
}
catch(JposException e)
{
	e.printStackTrace();
}
```

6-2-50 Eject And Retract Paper()

Eject the paper and reclaim it after the set time has elapsed.

Available on BK3R-3, BK5R-3 Presenter Accessory models only

[Syntax]
void ejectAndRetractPaper(int timeoutSeconds) throws JposException

[Parameters]
• int timeoutSeconds: Wait time (in seconds) between paper ejection and reclaim.

[Example]

```
import com.bxl.config.editor.BXLConfigLoader;
import jpos.config.JposEntry;
import jpos.POSPrinter;
import jpos.POSPrinterConst;
import com.bxl.BXLConst;

try
{
	POSPrinter posPrinter = new POSPrinter(this);
	posPrinter.open("BK3-3");
	posPrinter.claim(5000);
	posPrinter.setDeviceEnabled(true);

	posPrinter. ejectAndRetractPaper(5);
}
catch(JposException e)
{
	e.printStackTrace();
}
```

6-3 MSR Class

• This is a class for MSR(Magnetic Stripe Reader) control in the printer. The Card Track information from the MSR can be obtained using this class. It generates a JposException when an error occurs while performing a specific function. (Refer to “5-1 JposException”)

6-3-1 Open()

It initiates use of MSR class and includes initialization operations such as memory allocation. It must be first performed to call a Method above Claim.
Devices not saved via the BXLConfigLoader Class will not be opened.

[Syntax]
void open(String logicalDeviceName) throws JposException

[Parameters]
• String logicalDeviceName: device name to be opened(MSR)

[Example]

```
import jpos.JposConst;
import jpos.JposException;
import jpos.MSR;
import jpos.MSRConst;
import jpos.events.DataEvent;
import jpos.events.DataListener;

try
{
	MSR msr = new MSR();
	msr.open(“MSR”);
}
catch(JposException e)
{
	e.printStackTrace();
}
```

6-3-2 Claim()

It attempts to actually open the port included in the device information, and includes some initialization operations such as memory allocation initialization.
It must be first performed to enable the use of device.

[Syntax]
void claim(int timeout) throws JposException

[Parameters]
• int timeout: Attempt to open the port for the time specified in this parameter.

[Example]

```
import jpos.JposConst;
import jpos.JposException;
import jpos.MSR;
import jpos.MSRConst;
import jpos.events.DataEvent;
import jpos.events.DataListener;

try
{
	MSR msr = new MSR();
	msr.open("MSR");
	msr.claim(3000);
}
catch(JposException e)
{
	e.printStackTrace();
}
```

6-3-3 Set Device Enabled()

It includes whether the device will be used or not.
If the value of DeviceEnabled is disabled, the function may not be available.

[Syntax]
void setDeviceEnabled(boolean deviceEnabled) throws JposException

[Parameters]
• boolean deviceEnabled: Enter the device activation status.

Value	Description
true	Enabled
false	Disabled

[Example]

```
import jpos.JposConst;
import jpos.JposException;
import jpos.MSR;
import jpos.MSRConst;
import jpos.events.DataEvent;
import jpos.events.DataListener;

try
{
	MSR msr = new MSR();
	msr.open("MSR");
	msr.claim(3000);
	msr.setDeviceEnabled(true);
}
catch(JposException e)
{
	e.printStackTrace();
}
```

6-3-4 Release()

Physically terminates the use of the port of the claimed Device.
Operations such as freeing up memory can be performed.

[Syntax]
void release() throws JposException

[Example]

```
import jpos.JposConst;
import jpos.JposException;
import jpos.MSR;
import jpos.MSRConst;
import jpos.events.DataEvent;
import jpos.events.DataListener;

try
{
    msr.release();
}
catch(JposException e)
{
    e.printStackTrace();
}
```

6-3-5 Close()

Terminates use of the opened device.
Some operations such as freeing up memory may be performed.

[Syntax]
void close() throws JposException

[Example]

```
import jpos.JposConst;
import jpos.JposException;
import jpos.MSR;
import jpos.MSRConst;
import jpos.events.DataEvent;
import jpos.events.DataListener;

try
{
    msr.close();
    msr.setDeviceEnabled(false);
}
catch(JposException e)
{
    e.printStackTrace();
}
```

6-3-6 Set Auto Disable()

Disables MSR device automatically after MSR reading.
If AutoDisable is true, it changes DeviceEnabled to false after receiving MSR data.

[Syntax]
void setAutoDisable(boolean autoDisable) throws JposException

[Parameters]
• boolean autoDisable: Specify whether Auto Disable is enabled or not.

Value	Description
true	Enabled
false	Disabled

[Example]

```
import jpos.JposConst;
import jpos.JposException;
import jpos.MSR;
import jpos.MSRConst;
import jpos.events.DataEvent;
import jpos.events.DataListener;

try
{
	MSR msr = new MSR();
	msr.open("MSR");
	msr.claim(3000);
	msr.setDeviceEnabled(true);

	msr.setAutoDisable(true);
	// msr.setAutoDisable(false);
}
catch(JposException e)
{
	e.printStackTrace();
}
```

6-3-7 Set Data Event Enabled()

Specifies whether to receive track information via Data Event after MSR reading.

[Syntax]
void setDataEventEnabled(boolean dataEventEnabled) throws JposException

[Parameters]
• boolean dataEventEnabled: Specify whether or not to use Data Event.

Value	Description
true	Event Use (Refer to “5-2-4 Data Event”)
false	Event Not use

[Example]

```
import jpos.JposConst;
import jpos.JposException;
import jpos.MSR;
import jpos.MSRConst;
import jpos.events.DataEvent;
import jpos.events.DataListener;

try
{
	MSR msr = new MSR();
	msr.open(“MSR”);
	msr.claim(3000);
	msr.setDeviceEnabled(true);

	msr.addDataListener(this);
	msr.setDataEventEnabled(true);
}
catch(JposException e)
{
	e.printStackTrace();
}
```

6-3-8 Set Data Encryption Algorithm()

Specifies whether or not to encrypt MSR Track data.

[Syntax]
void setDataEncryptionAlgorithm(int encryptAlgorithm) throws JposException

[Parameters]
• int encryptAlgorithm: Specify whether Track Encryption is enabled or not.
(Refer to "MSR Encryption")

[Example]

```
import jpos.JposConst;
import jpos.JposException;
import jpos.MSR;
import jpos.MSRConst;
import jpos.events.DataEvent;
import jpos.events.DataListener;

try
{
      MSR msr = new MSR();
      msr.open("MSR");
      msr.claim(3000);
      msr.setDeviceEnabled(true);

      msr. setDataEncryptionAlgorithm(MSRConst.MSR_DE_NONE);
}
catch(JposException e)
{
      e.printStackTrace();
}
```

6-3-9 Get Track 1Data()

Obtains the Track1 data of the most recently read MSR Card.

[Syntax]
byte[] getTrack1Data() throws JposException

[Return Values]
※ Track1 Data of MSR Card

[Example]

```
import jpos.JposConst;
import jpos.JposException;
import jpos.MSR;
import jpos.MSRConst;
import jpos.events.DataEvent;
import jpos.events.DataListener;

try
{
	MSR msr = new MSR();
	msr.open("MSR");
	msr.claim(3000);
	msr.setDeviceEnabled(true);

	byte[] track1 = msr. getTrack1Data();
}
catch(JposException e)
{
	e.printStackTrace();
}
```

6-3-10 Get Track 2Data()

Obtains the Track2 data of the most recently read MSR Card.

[Syntax]
byte[] getTrack2Data() throws JposException

[Return Values]
※ Track2 Data of MSR Card

[Example]

```
import jpos.JposConst;
import jpos.JposException;
import jpos.MSR;
import jpos.MSRConst;
import jpos.events.DataEvent;
import jpos.events.DataListener;

try
{
	MSR msr = new MSR();
	msr.open("MSR");
	msr.claim(3000);
	msr.setDeviceEnabled(true);

	byte[] track2 = msr. getTrack2Data();
}
catch(JposException e)
{
	e.printStackTrace();
}
```

6-3-11 Get Track 3Data()

Obtains the Track3 data of the most recently read MSR Card.

[Syntax]
byte[] getTrack3Data() throws JposException

[Return Values]
※ Track3 Data of MSR Card

[Example]

```
import jpos.JposConst;
import jpos.JposException;
import jpos.MSR;
import jpos.MSRConst;
import jpos.events.DataEvent;
import jpos.events.DataListener;

try
{
	MSR msr = new MSR();
	msr.open("MSR");
	msr.claim(3000);
	msr.setDeviceEnabled(true);

	byte[] track3 = msr. getTrack3Data();
}
catch(JposException e)
{
	e.printStackTrace();
}
```

6-4 SmartCardRW Class

• This is a class for SCR(Smart Card Reader) control in the printer. It generates a JposException when an error occurs while performing a specific function.
("5-1 JposException" Refer to)

6-4-1 Open()

It initiates use of SmartCardRW class and includes initialization operations such as memory allocation. It must be first performed to call a Method above Claim.
Devices not saved via the BXLConfigLoader Class will not be opened.

[Syntax]
void open(String logicalDeviceName) throws JposException

[Parameters]
• String logicalDeviceName: device name to be opened(SmartCardRW)

[Example]

```
import jpos.JposConst;
import jpos.JposException;
import jpos.SmartCardRW;
import jpos.SmartCardRWConst;

try
{
    SmartCardRW smartCardRW = new SmartCardRW();
    smartCardRW.open("SmartCardRW");
}
catch(JposException e)
{
    e.printStackTrace();
}
```

6-4-2 Claim()

It attempts to actually open the port included in the device information, and includes some initialization operations such as memory allocation initialization.
It must be first performed to enable the use of device.

[Syntax]
void claim(int timeout) throws JposException

[Parameters]
• int timeout: Attempt to open the port for the time specified in this parameter.

[Example]

```
import jpos.JposConst;
import jpos.JposException;
import jpos.SmartCardRW;
import jpos.SmartCardRWConst;

try
{
    SmartCardRW smartCardRW = new SmartCardRW();
    smartCardRW.open("SmartCardRW");
    smartCardRW.claim(3000);
}
catch(JposException e)
{
    e.printStackTrace();
}
```

6-4-3 Set Device Enabled()

It includes whether the device will be used or not.
If the value of DeviceEnabled is disabled, the function may not be available.

[Syntax]
void setDeviceEnabled(boolean deviceEnabled) throws JposException

[Parameters]
• boolean deviceEnabled: Enter the device activation status.

Value	Description
true	Enabled
false	Disabled

[Example]

```
import jpos.JposConst;
import jpos.JposException;
import jpos.SmartCardRW;
import jpos.SmartCardRWConst;

try
{
    SmartCardRW smartCardRW = new SmartCardRW();
    smartCardRW.open("SmartCardRW");
    smartCardRW.claim(3000);
    SmartCardRW.setDeviceEnabled(true);
}
catch(JposException e)
{
    e.printStackTrace();
}
```

6-4-4 Release()

Physically terminates use of the port of the claimed Device.
Operations such as freeing up memory can be performed.

[Syntax]
void release() throws JposException

[Example]

```
import jpos.JposConst;
import jpos.JposException;
import jpos.SmartCardRW;
import jpos.SmartCardRWConst;

try
{
    smartCardRW.release();
}
catch(JposException e)
{
    e.printStackTrace();
}
```

6-4-5 Close()

Terminates use of the opened device.
Operations such as freeing up memory can be performed.

[Syntax]
void close() throws JposException

[Example]

```
import jpos.JposConst;
import jpos.JposException;
import jpos.SmartCardRW;
import jpos.SmartCardRWConst;

try
{
    smartCardRW.close();
    smartCardRW.setDeviceEnabled(false);
}
catch(JposException e)
{
    e.printStackTrace();
}
```

6-4-6 Set SCSlot()

Selects the card slot for communication.

[Syntax]
void setSCSlot() throws JposException

[Parameters]
• int scSlot: Select the card slot.

Value	Description
0x1000	Smart Card
0x0100	SAM1
0x0010	SAM2

[Example]

```
import jpos.JposConst;
import jpos.JposException;
import jpos.SmartCardRW;
import jpos.SmartCardRWConst;

try
{
    SmartCardRW smartCardRW = new SmartCardRW();
    smartCardRW.open("SmartCardRW");
    smartCardRW.claim(3000);
    SmartCardRW.setDeviceEnabled(true);

    smartCardRW.setSCSlot(0x01 << (Integer.SIZE - 1)); // Smart Card
    smartCardRW.setSCSlot(0x01 << (Integer.SIZE - 2)); // SAM1
    smartCardRW.setSCSlot(0x01 << (Integer.SIZE - 3)); // SAM2
}
catch(JposException e)
{
    e.printStackTrace();
}
```

6-4-7 Set Iso Emv Mode()

Selects the ISO/EMV mode.

[Syntax]
void setIsoEmvMode(int isoEmvMode) throws JposException

[Parameters]
• int isoEmvMode: Select the mode.(Refer to “5-11 SCR Mode”)

[Example]

```
import jpos.JposConst;
import jpos.JposException;
import jpos.SmartCardRW;
import jpos.SmartCardRWConst;

try
{
    SmartCardRW smartCardRW = new SmartCardRW();
    smartCardRW.open(“SmartCardRW”);
    smartCardRW.claim(3000);
    SmartCardRW.setDeviceEnabled(true);

    smartCardRW.setIsoEmvMode(SmartCardRWConst.SC_CMODE_EMV); // EMV Mode
    smartCardRW.setIsoEmvMode(SmartCardRWConst.SC_CMODE_ISO); // ISO Mode
}
catch(JposException e)
{
    e.printStackTrace();
}
```

6-4-8 Begin Insertion()

Checks if the Smart Card is inserted for the specified time.
Be sure to call the endInsertion () function after calling the function.

[Syntax]
void beginInsertion(int timeout) throws JposException

[Parameters]
• int timeout: Sets card insertion check time

[Example]

```
import jpos.JposConst;
import jpos.JposException;
import jpos.SmartCardRW;
import jpos.SmartCardRWConst;

try
{
    SmartCardRW smartCardRW = new SmartCardRW();
    smartCardRW.open("SmartCardRW");
    smartCardRW.claim(5000);
    SmartCardRW.setDeviceEnabled(true);

    smartCardRW. beginInsertion(5000);
    smartCardRW.endInsertion();
}
catch(JposException e)
{
    e.printStackTrace();
}
```

6-4-9 End Insertion()

Applies power to the inserted Smart Card chip.
Be sure to call the beginInsertion function before calling the function.

[Syntax]
void endInsertion() throws JposException

[Example]

```
import jpos.JposConst;
import jpos.JposException;
import jpos.SmartCardRW;
import jpos.SmartCardRWConst;

try
{
    SmartCardRW smartCardRW = new SmartCardRW();
    smartCardRW.open("SmartCardRW");
    smartCardRW.claim(5000);
    SmartCardRW.setDeviceEnabled(true);

    smartCardRW. beginInsertion(5000);
    smartCardRW.endInsertion();}
catch(JposException e)
{
    e.printStackTrace();
}
```

6-4-10 Begin Removal()

Terminates the power to the Smart Card chip.
Be sure to call the endRemoval function after calling the function.

[Syntax]
void beginRemoval(int timeout) throws JposException

[Parameters]
• int timeout: Sets power-off time

[Example]

```
import jpos.JposConst;
import jpos.JposException;
import jpos.SmartCardRW;
import jpos.SmartCardRWConst;

try
{
    SmartCardRW smartCardRW = new SmartCardRW();
    smartCardRW.open("SmartCardRW");
    smartCardRW.claim(5000);
    SmartCardRW.setDeviceEnabled(true);

    smartCardRW.beginRemoval(5000);
    smartCardRW.endRemoval();
}
catch(JposException e)
{
    e.printStackTrace();
}
```

6-4-11 End Removal()

Checks if the power to the Smart Card chip has been terminated normally.
Be sure to call the beginRemoval function before calling the function.

[Syntax]
void endRemoval() throws JposException

[Example]

```
import jpos.JposConst;
import jpos.JposException;
import jpos.SmartCardRW;
import jpos.SmartCardRWConst;

try
{
    SmartCardRW smartCardRW = new SmartCardRW();
    smartCardRW.open("SmartCardRW");
    smartCardRW.claim(5000);
    SmartCardRW.setDeviceEnabled(true);

    smartCardRW.beginRemoval(5000);
    smartCardRW.endRemoval();
}
catch(JposException e)
{
    e.printStackTrace();
}
```

6-4-12 Read Data()

Reads and writes to Smart Card chip.

[Syntax]
void readData(int action, int[] count, String[] data) throws JposException

[Parameters]
• int action: fixed Value SmartCardRWConst.SC_READ_DATA
• int[] count: Response data size
• String[] data: R/W buffer (“cp437” encoding)

[Example]

```
import jpos.JposConst;
import jpos.JposException;
import jpos.SmartCardRW;
import jpos.SmartCardRWConst;

try
{
	SmartCardRW smartCardRW = new SmartCardRW();
	smartCardRW.open(“SmartCardRW”);
	smartCardRW.claim(3000);
	smartCardRW.setDeviceEnabled(true);

	// Power up
	smartCardRW.beingInsertion(5000);
	smartCardRW.endInsertion();

	String[] data = new String[]{
		new String(new byte[]{
			0x00, (byte) 0xA4, 0x04, 0x00, 0x07, (byte) 0xD4, 0x10, 0x65,
			0x09, (byte) 0x90, 0x00, 0x10}, "CP437")
	int[] count = new int[1];

	smartCardRW. readData(SmartCardRWConst.SC_READ_DATA, count, data);
	int rspSize = count[0];
	byte[] rspData = data[0].getBytes("CP437");

	// Power down
	smartCardRW.beginRemoval(5000);
	smartCardRW.endRemoval();
}
catch(JposException e)
{
	e.printStackTrace();
}
```

6-5 Cash Drawer Class

• This is a class for CashDrawer control. It generates a JposException when an error occurs while performing a specific function. (Refer to “5-1 JposException”)

6-5-1 Open()

It initiates use of CashDrawer class and includes initialization operations such as memory allocation. It must be first performed to call a Method above Claim. Devices not saved via the BXLConfigLoader Class will not be opened.

[Syntax]
void open(String logicalDeviceName) throws JposException

[Parameters]
• String logicalDeviceName: device name to be opened(CashDrawer)

[Example]

```
import jpos.JposConst;
import jpos.JposException;
import jpos.CashDrawer;
import jpos.CashDrawerConst;

try
{
    CashDrawer cashDrawer = new CashDrawer();
    cashDrawer.open(“CashDrawer”);
}
catch(JposException e)
{
    e.printStackTrace();
}
```

6-5-2 Claim()

It attempts to actually open the port included in the device information, and includes some initialization operations such as memory allocation initialization.
It must be first performed to enable the use of device.

[Syntax]
void claim(int timeout) throws JposException

[Parameters]
• int timeout: Attempt to open the port for the time specified in this parameter.

[Example]

```
import jpos.JposConst;
import jpos.JposException;
import jpos.CashDrawer;
import jpos.CashDrawerConst;

try
{
	CashDrawer cashDrawer = new CashDrawer();
	cashDrawer.open("CashDrawer");
	cashDrawer.claim(3000);
}
catch(JposException e)
{
	e.printStackTrace();
}
```

6-5-3 Set Device Enabled()

It includes whether the device will be used or not.
If the value of DeviceEnabled is disabled, the function may not be available.

[Syntax]
void setDeviceEnabled(boolean deviceEnabled) throws JposException

[Parameters]
• boolean deviceEnabled: Enter the device activation status.

Value	Description
true	Enabled
false	Disabled

[Example]

```
import jpos.JposConst;
import jpos.JposException;
import jpos.CashDrawer;
import jpos.CashDrawerConst;

try
{
    CashDrawer cashDrawer = new CashDrawer();
    cashDrawer.open("CashDrawer");
    cashDrawer.claim(3000);
    cashDrawer.setDeviceEnabled(true);
}
catch(JposException e)
{
    e.printStackTrace();
}
```

6-5-4 Release()

Physically terminates the use of the port of the claimed Device.
Operations such as freeing up memory can be performed.

[Syntax]
void release() throws JposException

[Example]

```
import jpos.JposConst;
import jpos.JposException;
import jpos.CashDrawer;
import jpos.CashDrawerConst;

try
{
    cashDrawer.release();
}
catch(JposException e)
{
    e.printStackTrace();
}
```

6-5-5 Close()

Terminates use of the opened device.
Operations such as freeing up memory can be performed.

[Syntax]
void close() throws JposException

[Example]

```
import jpos.JposConst;
import jpos.JposException;
import jpos.CashDrawer;
import jpos.CashDrawerConst;

try
{
    cashDrawer.close();
    cashDrawer.setDeviceEnabled(false);
}
catch(JposException e)
{
    e.printStackTrace();
}
```

6-5-6 Open Drawer()

Opens Cash Drawer.

[Syntax]
void openDrawer() throws JposException

[Example]

```
import jpos.JposConst;
import jpos.JposException;
import jpos.CashDrawer;
import jpos.CashDrawerConst;

try
{
    CashDrawer cashDrawer = new CashDrawer();
    cashDrawer.open("CashDrawer");
    cashDrawer.claim(3000);
    cashDrawer.setDeviceEnabled(true);

    cashDrawer.openDrawer();
}
catch(JposException e)
{
    e.printStackTrace();
}
```

6-5-7 Get Drawer Opened()

Returns the open/close status of Cash Drawer.

[Syntax]
boolean getDrawerOpened() throws JposException

[Return Values]

Value	Description
true	Cash drawer is opened.
false	Cash drawer is closed.

[Example]

```
import jpos.JposConst;
import jpos.JposException;
import jpos.CashDrawer;
import jpos.CashDrawerConst;

try
{
	CashDrawer cashDrawer = new CashDrawer();
	cashDrawer.open("CashDrawer");
	cashDrawer.claim(3000);
	cashDrawer.setDeviceEnabled(true);

	if(cashDrawer.getDrawerOpened())
	{
		// opened
	}
	else
	{
		// closed
	}
}
catch(JposException e)
{
	e.printStackTrace();
}
```

7. Samples for Test

7-1 Text print

```
import com.bxl.config.editor.BXLConfigLoader;
import jpos.config.JposEntry;
import jpos.POSPrinter;
import jpos.POSPrinterConst;

try
{
	// BXLConfigLoader creating / setting file open
	BXLConfigLoader bxlConfigLoader = new BXLConfigLoader(this);
	bxlConfigLoader.openFile();
}
catch(JposException e)
{
	e.printStackTrace();
	bxlConfigLoader.newFile();
}

try
{
	// Adding device information
	bxlConfigLoader.addEntry("SPP-R200III",
					BXLConfigLoader.DEVICE_CATEGORY_POS_PRINTER,
					BXLConfigLoader.PRODUCT_NAME_SPP_R200III,
					BXLConfigLoader.DEVICE_BUS_BLUETOOTH,
					"74:F0:7D:E4:11:AF");
	// Saving setting file
	bxlConfigLoader.saveFile();

	// printer Open/Claim/DeviceEnabled
	POSPrinter posPrinter = new POSPrinter(this);
	posPrinter.open("SPP-R200III");
	posPrinter.claim(5000);
	posPrinter.setDeviceEnabled(true);

	// text print
	posPrinter.printNormal(POSPrinterConst.PTR_S_RECEIPT, "Print Data\n");
	posPrinter.printNormal(POSPrinterConst.PTR_S_RECEIPT,
				EscapeSequence.getString(7) + "www.bixolon.com\n");

	posPrinter.close();
}
catch(JposException e)
{
```

```
        e.printStackTrace();
}
```

7-2 Image print

```
import com.bxl.config.editor.BXLConfigLoader;
import jpos.config.JposEntry;
import jpos.POSPrinter;
import jpos.POSPrinterConst;

try
{
    // BXLConfigLoader creating / setting file open
    BXLConfigLoader bxlConfigLoader = new BXLConfigLoader(this);
    bxlConfigLoader.openFile();
}
catch(JposException e)
{
    e.printStackTrace();
    bxlConfigLoader.newFile();
}

try
{
    // Adding device information
    bxlConfigLoader.addEntry("SPP-R200III",
                             BXLConfigLoader.DEVICE_CATEGORY_POS_PRINTER,
                             BXLConfigLoader.PRODUCT_NAME_SPP_R200III,
                             BXLConfigLoader.DEVICE_BUS_BLUETOOTH,
                             "74:F0:7D:E4:11:AF");
    // Saving setting file
    bxlConfigLoader.saveFile();

    // print Open/Claim/DeviceEnabled
    POSPrinter posPrinter = new POSPrinter(this);
    posPrinter.open("SPP-R200III");
    posPrinter.claim(5000);
    posPrinter.setDeviceEnabled(true);

    // image print
    ByteBuffer buffer = ByteBuffer.allocate(4);
    buffer.put((byte) POSPrinterConst.PTR_S_RECEIPT);
    buffer.put((byte) 80); // brightness
    buffer.put((byte) 0x01); // compress
    buffer.put((byte) 0x00);

    posPrinter.printBitmap(buffer.getInt(0),
                           "/storage/emulated/0/kicc/sample_image.png",
                           384,
                           POSPrinterConst.PTR_BM_LEFT);
```

```
        posPrinter.close();
}
catch(JposException e)
{
        e.printStackTrace();
}
```

7-3 Page mode print

```
import com.bxl.config.editor.BXLConfigLoader;
import jpos.config.JposEntry;
import jpos.POSPrinter;
import jpos.POSPrinterConst;

try
{
	// BXLConfigLoader creating / setting file open
	BXLConfigLoader bxlConfigLoader = new BXLConfigLoader(this);
	bxlConfigLoader.openFile();
}
catch(JposException e)
{
	e.printStackTrace();
	bxlConfigLoader.newFile();
}

try
{
	// Adding device information
	bxlConfigLoader.addEntry("SPP-R200III",
					BXLConfigLoader.DEVICE_CATEGORY_POS_PRINTER,
					BXLConfigLoader.PRODUCT_NAME_SPP_R200III,
					BXLConfigLoader.DEVICE_BUS_BLUETOOTH,
					"74:F0:7D:E4:11:AF");
	// Saving setting file
	bxlConfigLoader.saveFile();

	// printer Open/Claim/DeviceEnabled
	POSPrinter posPrinter = new POSPrinter(this);
	posPrinter.open("SPP-R200III");
	posPrinter.claim(5000);
	posPrinter.setDeviceEnabled(true);

	ByteBuffer buffer = ByteBuffer.allocate(4);
	buffer.put((byte) POSPrinterConst.PTR_S_RECEIPT);
	buffer.put((byte) 80); // brightness
	buffer.put((byte) 0x01); // compress
	buffer.put((byte) 0x00);

	// Page mode print area setting
	posPrinter.setPageModePrintArea("0,0,384,1200");
	// Page mode print direction setting
	posPrinter.setPageModePrintDirection(POSPrinterConst.PTR_PD_LEFT_TO_RIGHT);
	// Page mode change
	posPrinter.pageModePrint(POSPrinterConst.PTR_PM_PAGE_MODE);
```

```
    // print location setting : width/height
    posPrinter.setPageModeHorizontalPosition(0);
    posPrinter.setPageModeVerticalPosition(0);
    posPrinter.printBitmap(buffer.getInt(0),
                           “/storage/emulated/0/kicc/sample_image.png”,
                           384,
                           POSPrinterConst.PTR_BM_LEFT);

    posPrinter.setPageModeHorizontalPosition(130);
    posPrinter.setPageModeVerticalPosition(200);
    posPrinter.printNormal(POSPrinterConst.PTR_S_RECEIPT,
                           EscapeSequence.getString(0) + // Normal
                           EscapeSequence.getString(7) + // Bold
                           EscapeSequence.getString(18) + //Scale_2_time_horizontally
                           EscapeSequence.getString(26) + // Scale_2_time_vertically
                           "www.bixolon.com\n");

    posPrinter.setPageModeHorizontalPosition(340);
    posPrinter.setPageModeVerticalPosition(550);
    String data = "www.bixolon.com\n";
    posPrinter.printBarCode(POSPrinterConst.PTR_S_RECEIPT,
                            data,
                            POSPrinterConst.PTR_BCS_QRCODE,
                            0, 6/*1to8*/,
                            POSPrinterConst.PTR_BC_LEFT,
                            POSPrinterConst.PTR_BC_TEXT_NONE);

    // Start of page mode print
    posPrinter.pageModePrint(POSPrinterConst.PTR_PM_NORMAL);

    posPrinter.close();
}
catch(JposException e)
{
    e.printStackTrace();
}
```

Revision history

Ver.	Date	Description
2.00	21-08-2017	New
2.01	22-11-2017	SRP-QE300/SRP-QE302 model added. Add missing content.
2.02	09-01-2018	SRP-E300/SRP-E302 models added. API descriptions for controlling BCD-3000 added. Battery status event and others added
2.03	28-05-2018	Sample application changed and reference class added API description on SmartCard control added Missing contents added
2.04	03-08-2018	SRP-383 model added, Add missing content.
2.05	12-09-2018	BK3-3 model added. descriptions for FeedCut added.
2.06	08-03-2019	SRP-Q200 model added. API for NV image download added. Edit example code typo
2.07	26-09-2019	API description on printSvg added. API description on updateFirmware added. printPdf not supported.
2.08	29-11-2019	BK3-2 model added. API description on BK3-3 Accessory(Presenter) added.
2.09	24-12-2019	library file modified. Support barcode definition modified.
2.10	15-09-2020	SRP-S320 model added.
2.11	30-10-2020	SRP-S3000 model added.
2.12	24-01-2021	LZMA option added.
2.13	-	SRP-B300 model added.
2.14	-	SRP-S200 model added.
2.15	19-10-2021	SPP-C200 model added.
2.16	17-01-2022	Add PDF print API
2.17	21-03-2022	SPP-C300 model added.
2.18	10-06-2022	Add new model : SRP-350V / SRP-352V : SRP-350plusV / SRP-352plusV : SRP-330III / SRP-332III
2.19	28-06-2022	Delete PDF license.
2.20	17-08-2022	SRP-380/2II model added.
2.21	16-01-2023	BK5-3 model added.
	30-03-2023	G30 model added.
2.22	17-07-2023	Support OS update.

2.23	24-10-2023	SRP-Q300II model added. Added USB peripheral API.
2.24	27-08-2024	SRP-S300II model added
2.25	26-12-2024	SRP-380II/SRP-382II Renaming models : SRP-380plus/SRP-382plus
2.26	14-01-2025	BK3R-3, BK5R-3 model added Add a paper reclaim API