

# CITIZEN

## **Android Label Print SDK Programming Manual**

**Version 1.05**

**CITIZEN SYSTEMS JAPAN CO., LTD.**

## Revision History

Date	Version	Description
Sep 28, 2016	1.00	First issue.
Mar 15, 2017	1.01	- Renamed the namespace to "com.citizen.sdk.labelprint". - Added an explanation of ParallelErrorOutput to printerCheck method. - Added the measurementUnit as a parameter of drawTextLocalFont method. - Added the resolution and measurementUnit as a parameter of drawBitmap Method.
Sep 29, 2017		- Added CL-E300/CL-E303 to support model.
Jan 24, 2018		- Added CL-E321/CL-E331 to support model.
Sep 13, 2018	1.02	- Added Bluetooth as a interface of CL-E720/730 model. - Modified the printer model name. - Added a locale for Chinese model and Korean model to drawTextPtrFont. - Added the Chinese model and the Korean model to the definition of the locale.
Apr 4, 2019	1.03	- Added binary data specification to drawQRCode method.
Jul 31, 2019	1.04	- Modified the setting contents of resolution and measurementUnit parameter of drawTextLocalFont method. - Modified the setting contents of measurementUnit parameter of drawBitmap method.
Nov 12, 2019		- Added CL-S700II/703II, CL-S621II/631II and CL-S521II/531II to support model.
Jun 2, 2020		- Added CL-E300EX/303EX and CL-E321EX/331EX to support model.
Dec 22, 2023		- Added CL-S700III/703III to support model.
Oct 22, 2025	1.05	- Modified explanation and usage example of PrinterCheck method. (Page 21) - Added PrinterCheckEx and ClearOutputEx meshods. (Page 32,35) - Modified the description of the log function. (Page 37,102) - Added PrintHeadLowTemp, PrintHeadFailure, PrintHeadOverheat, MechanismOpen, AutoCutterError, FanMotorError and MiscError properties to the LabelPrinter class. (Page 64-70) - Supported UTF-8 for DrawTextPtrFont and DrawTextDLFont methods. (Page 72,74)

## Permission Notice

1. Unauthorized use of all or any part of this document is prohibited.
2. The information in this document is subject to change without prior notice.
3. This document has been created with full attention. If, however, you find an error or question, please contact us.
4. We shall not be liable for any effect resulting from operation regardless of the above item 3.
5. If you do not agree with the above terms, you are not permitted to use this SDK.

## Trademark

Android is registered trademarks of Google Inc. in the United States and/or other countries.

Java is a registered trademark of Oracle in the United States and/or other countries.

Company names and product names appearing on this document are trademarks and/or registered trademarks of respective companies.

CITIZEN is a registered trademark of Citizen Watch Co., Ltd.

# Table of Contents

<b>1. Introduction</b>	<b>6</b>
1.1. Who should read this document	6
1.2. System summary	6
1.3. Adding library to your .net project	8
1.4. Program structure	10
1.5. Class summary	12
<b>2. API specification</b>	<b>15</b>
2.1. Return value	15
2.2. LabelPrinter class	16
2.2.1 Constructor	16
2.2.2 setContext method	17
2.2.3 connect method	18
2.2.4 disconnect method	20
2.2.5 printerCheck method	21
2.2.6 print method	23
2.2.7 storeNVBitmap method	24
2.2.8 clearOutPut method	25
2.2.9 sendData method	26
2.2.10 searchCitizenPrinter method	27
2.2.11 searchLabelPrinter method	29
2.2.12 printerCheckEx method	30
2.2.13 clearOutputEx method	33
2.2.14 setLog method	35
2.2.15 HorizontalMagnification property	36
2.2.16 VerticalMagnification property	37
2.2.17 FormatAttribute property	38
2.2.18 ContinuousMediaLength property	39
2.2.19 MeasurementUnit property	40
2.2.20 PrintSpeed property	41
2.2.21 FeedSpeed property	42
2.2.22 SlewSpeed property	43
2.2.23 BackupSpeed property	44
2.2.24 PrintDarkness property	45
2.2.25 DoubleHeat property	46
2.2.26 VerticalOffset property	47
2.2.27 HorizontalOffset property	48
2.2.28 MediaHandling property	49
2.2.29 StartOffset property	50
2.2.30 StopOffset property	51
2.2.31 LabelSensor property	52
2.2.32 PrintMethod property	53
2.2.33 SensorLocation property	54
2.2.34 CommandInterpreterInAction property	55
2.2.35 PaperError property	56
2.2.36 RibbonEnd property	57
2.2.37 BatchProcessing property	58
2.2.38 Printing property	59
2.2.39 Pause property	60

2.2.40	<i>WaitingForPeeling</i> property .....	61
2.2.41	<i>PrintHeadLowTemp</i> property .....	62
2.2.42	<i>PrintHeadFailure</i> property .....	63
2.2.43	<i>PrintHeadOverheat</i> property .....	64
2.2.44	<i>MechanismOpen</i> property .....	65
2.2.45	<i>AutoCutterError</i> property .....	66
2.2.46	<i>FanMotorError</i> property .....	67
2.2.47	<i>MiscError</i> property .....	68
2.3.	<b>LabelDesign</b> class .....	69
2.3.1	<i>Constructor</i> .....	69
2.3.2	<i>drawTextPtrFont</i> method .....	70
2.3.3	<i>drawTextDLFont</i> method .....	72
2.3.4	<i>drawTextLocalFont</i> method .....	74
2.3.5	<i>drawNVBitmap</i> method .....	76
2.3.6	<i>drawBitmap</i> method .....	77
2.3.7	<i>drawBarCode</i> method .....	79
2.3.8	<i>drawMaxiCode</i> method .....	84
2.3.9	<i>drawPDF417</i> method .....	85
2.3.10	<i>drawDataMatrix</i> method .....	86
2.3.11	<i>drawQRCode</i> method .....	87
2.3.12	<i>drawAztec</i> method .....	88
2.3.13	<i>drawGS1DataBar</i> method .....	89
2.3.14	<i>drawLine</i> method .....	91
2.3.15	<i>drawRect</i> method .....	92
2.3.16	<i>fillRect</i> method .....	93
2.3.17	<i>drawCircle</i> method .....	94
2.3.18	<i>fillCircle</i> method .....	95
2.3.19	<i>drawPolygon</i> method .....	96
2.3.20	<i>fillPolygon</i> method .....	97
2.3.21	<i>embedRawDesignCommand</i> method .....	98
3.	<b>Appendix</b> .....	99
3.1.	Specifying object position .....	99
3.2.	Logging function .....	100
3.3.	Predefined Constants .....	102

# 1. Introduction

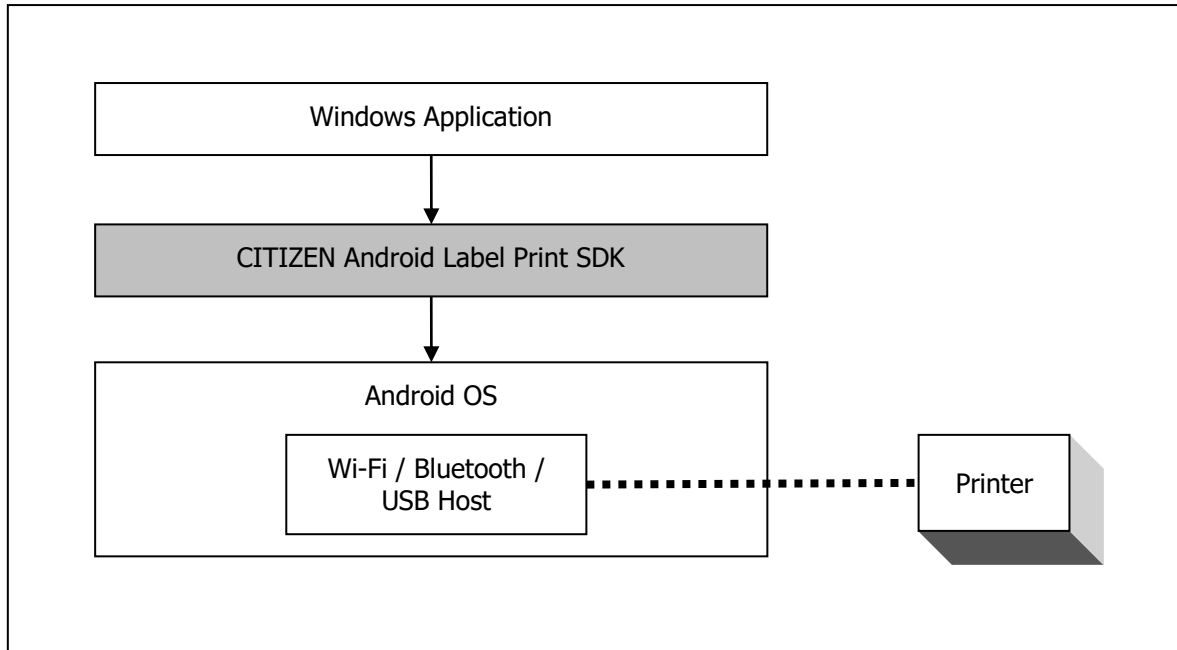
This document is a programming manual for the CITIZEN Android Label Print SDK.

## 1.1. Who should read this document

This document is intended for developers who integrate their Android application programs with Citizen label printers.

## 1.2. System summary

This SDK is referred by Android application program to control Citizen Label printers.



**System diagram of the SDK**

### Supported terminals

Below is a list of supported terminals for the SDK.

Android OS version:	Supported interfaces:
Android 2.3.3 (API Level 10) or newer	Wi-Fi, Bluetooth
Android 3.1 (API Level 12) or newer	Wi-Fi, Bluetooth, USB Host

### Supported printer models

Object Model	Interface	Print Method
CL-S700/703	Wired/Wireless LAN, USB	Direct thermal/Thermal transfer
CL-S700II/703II	Wired/Wireless LAN, USB	Direct thermal/Thermal transfer
CL-S700III/703III	Wired/Wireless LAN, USB	Direct thermal/Thermal transfer
CL-E720/730	Wired/Wireless LAN, USB, Bluetooth	Direct thermal/Thermal transfer
CL-S621/631	Wired/Wireless LAN, USB	Direct thermal/Thermal transfer
CL-S621II/631II	Wired/Wireless LAN, USB	Direct thermal/Thermal transfer
CL-S521/531	Wired/Wireless LAN, USB	Direct thermal
CL-S521II/531II	Wired/Wireless LAN, USB	Direct thermal
CL-S400DT	Wired/Wireless LAN, USB, Bluetooth	Direct thermal

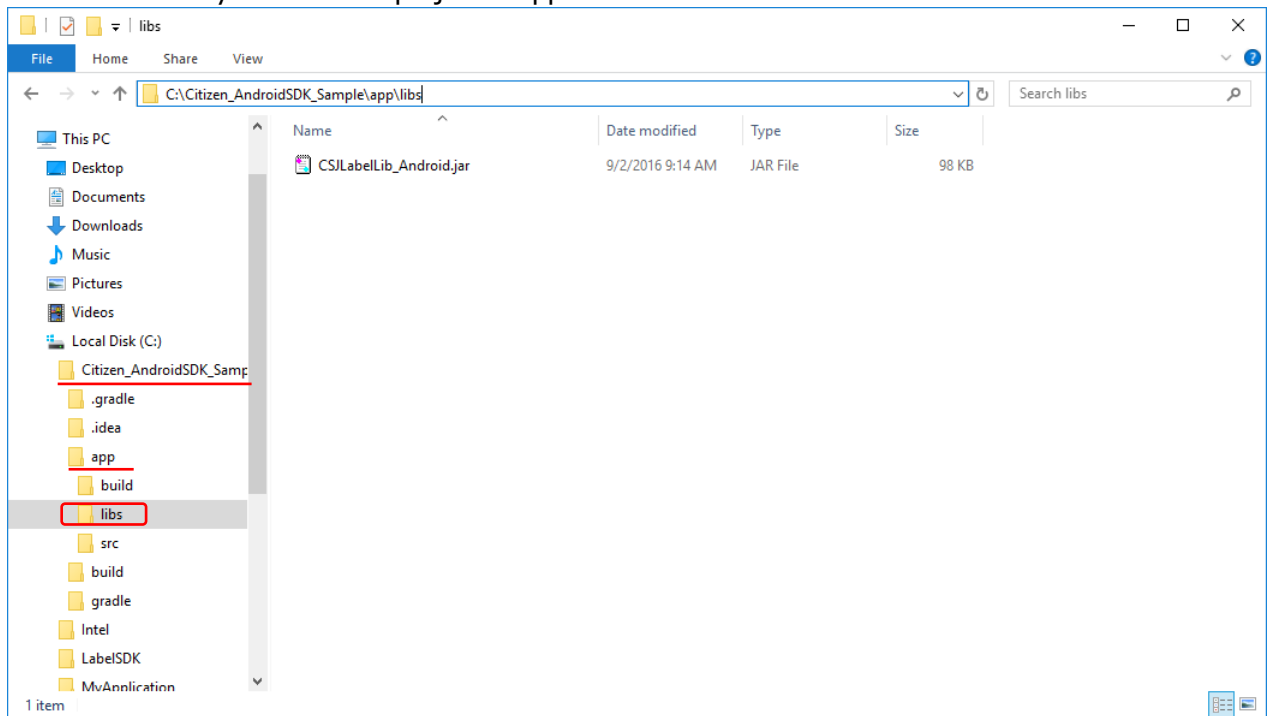
CL-E300/303	Wired LAN, USB	Direct thermal
CL-E300EX/303EX	Wired/Wireless LAN, USB, Bluetooth	Direct thermal
CL-E321/331	Wired LAN, USB	Direct thermal/Thermal transfer
CL-E321EX/331EX	Wired/Wireless LAN, USB, Bluetooth	Direct thermal/Thermal transfer

Refer to each user's manual for more details.

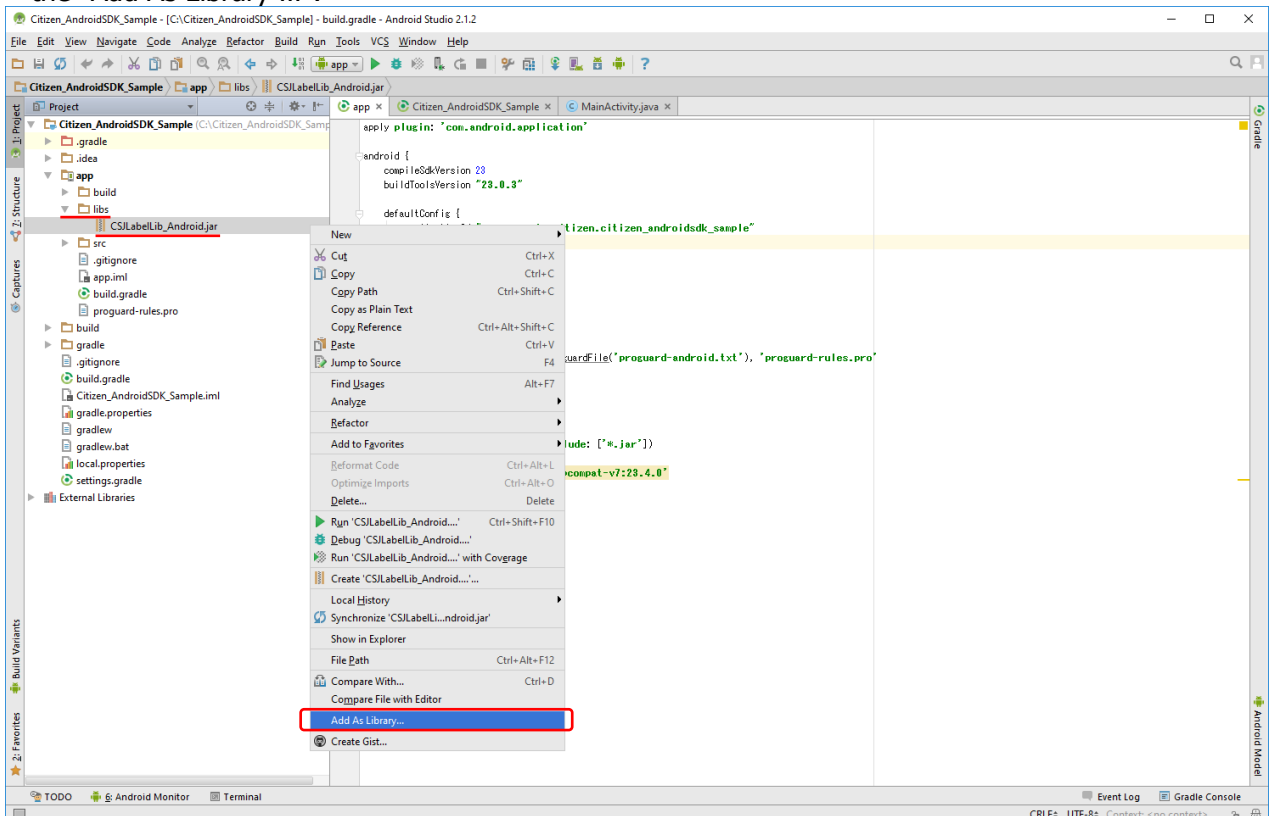
## 1.3.Adding library to your .net project

### Adding Library

Place the library file in the "<project>\app\libs" folder as below.



Start up the AndroidStudio, Right-click the CSJLabelLib\_Android.jar of libs in the folder, please select the "Add As Library ...".





**Declaration of Import**

Add the library file that is provided, if you use this SDK, please import the following three classes.

```
import com.citizen.sdk.labelprint.LabelConst;  
import com.citizen.sdk.labelprint.LabelPrinter;  
import com.citizen.sdk.labelprint.LabelDesign;
```

## 1.4. Program structure

Here is an example program in Java which uses the SDK.

```
// Create an instance( LabelDesign class )
LabelDesign design = new LabelDesign();

// Text
design.drawTextPtrFont("Sample Print",
    LabelConst.CLS_LOCALE_JP, LabelConst.CLS_PRT_FNT_TRIUMVIRATE_B,
    LabelConst.CLS_RT_NORMAL, 1, 1, LabelConst.CLS_PRT_FNT_SIZE_24, 20,
    300);

// QRCode
design.drawQRCode("DrawQRCode",
    LabelConst.CLS_ENC_CDPG_US_ASCII, LabelConst.CLS_RT_NORMAL, 4,
    LabelConst.CLS_QRCODE_EC_LEVEL_H, 20, 220);

// Rect(fill)
design.fillRect(20, 150, 350, 40, LabelConst.CLS_SHADED_PTN_11);

// BarCode
design.drawBarCode("0123456789",
    LabelConst.CLS_BCS_CODE128, LabelConst.CLS_RT_NORMAL,
    3, 3, 30, 20, 70, LabelConst.CLS_BCS_TEXT_SHOW);

// Create an instance( LabelPrinter class )
LabelPrinter printer = new LabelPrinter();

// Connect printer
int result = printer.connect(LabelConst.CLS_PORT_WiFi,
    "192.168.128.212")
if (LabelConst.CLS_SUCCESS == result)
{
    // Get Properties
    int printDarkness = printer.getPrintDarkness();
    if (LabelConst.CLS_PROPERTY_DEFAULT == printDarkness)
    {
        // Set Properties
        printer.setPrintDarkness(10);
    }

    // Print Label
    result = printer.print(design, 0001);

    // Disconnect
    printer.disconnect();

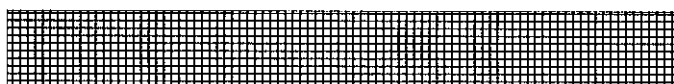
    if (LabelConst.CLS_SUCCESS != result)
    {
        // Print Error
        Toast.makeText(MainActivity.this, "Print Error :" +
            Integer.toString(result), Toast.LENGTH_LONG ).show();
    }
}
else
{
    // Connect Error
    Toast.makeText(MainActivity.this, "Connect or Printer Error :" +
        Integer.toString(result), Toast.LENGTH_LONG ).show();
}
}
```

Design of the  
label

Print of the  
label  
(Connect,  
Print,  
Disconnect)

**Printing result**

# Sample Print



0123456789

## 1.5. Class summary

### Methods (LabelPrinter class)

No	Method	Description
1	LabelPrinter	Constructor.
2	setContext	Set the context.
3	connect	Connects to a printer.
4	disconnect	Disconnects with a printer.
5	printerCheck	Gets status of a printer.
6	print	Prints labels.
7	storeNVBitmap	Stores a bitmap image into the flash memory.
8	clearOutput	Clears all data in the printer buffer.
9	sendData	Sends binary data to a printer.
10	searchCitizenPrinter	Searches available printers and gets printer information.
11	searchLabelPrinter	Searches available printers and gets addresses.
12	PrinterCheckEx	Connect and gets status of a printer.
13	ClearOutputEx	Connect and clears all data in the printer buffer.
14	setLog	Sets the log function.

### Properties (LabelPrinter class)

No	Property	Attribute	Description
1	HorizontalMagnification	R/W	Horizontal magnification of the label.
2	VerticalMagnification	R/W	Vertical magnification of the label.
3	FormatAttribute	R/W	Specifies how to print overlapped objects.
4	ContinuousMediaLength	R/W	Label length for continuous paper.
5	MeasurementUnit	R/W	Measurement unit. Inches or millimeters.
6	PrintSpeed	R/W	Print speed.
7	FeedSpeed	R/W	Feed speed.
8	SlewSpeed	R/W	Slew speed.
9	BackupSpeed	R/W	Backup speed.
10	PrintDarkness	R/W	Print darkness.
11	DoubleHeat	R/W	Double heat.
12	VerticalOffset	R/W	Vertical offset of the printing position.
13	HorizontalOffset	R/W	Horizontal offset of the printing position.
14	MediaHandling	R/W	Media handling after print. (Pause, Cut or Peel)
15	StartOffset	R/W	Vertical offset of the paper position when start printing.
16	StopOffset	R/W	Vertical offset of the paper position when stop printing.
17	LabelSensor	R/W	Label sensor type. (See through, Reflect or None)
18	PrintMethod	R/W	Thermal transfer or Direct thermal.
19	SensorLocation	R/W	Front or Rear. (for the models which have multiple sensors)
20	CommandInterpreterInAction	R	Status: CommandInterpreterInAction
21	PaperError	R	Status: PaperError
22	RibbonEnd	R	Status: RibbonEnd
23	BatchProcessing	R	Status: BatchProcessing
24	Printing	R	Status: Printing
25	Pause	R	Status: Pause
26	WaitingForPeeling	R	Status: WaitingForPeeling
27	PrintHeadLowTemp	R	Status: PrintHeadLowTemp
28	PrintHeadFailure	R	Status: PrintHeadFailure

29	PrintHeadOverheat	R	Status: PrintHeadOverheat
30	MechanismOpen	R	Status: MechanismOpen
31	AutoCutterError	R	Status: AutoCutterError
32	FanMotorError	R	Status: FanMotorError
33	MiscError	R	Status: MiscError

**Methods (LabelDesign class)**

No	Method	Description
1	LabelDesign	Constructor.
2	drawTextPtrFont	Draws text by using a printer device font.
3	drawTextDLFont	Draws text by using a downloaded font.
4	drawTextLocalFont	Draws text by using a font installed in the terminal.
5	drawNVBitmap	Draws a bitmap image stored in the flash memory of the printer.
6	drawBitmap	Draws a bitmap file in the computer.
7	drawBarCode	Draws a barcode.
8	drawMaxiCode	
9	drawPDF417	
10	drawDataMatrix	
11	drawQRCode	
12	drawAztec	
13	drawGS1DataBar	
14	drawLine	Draws a line.
15	drawRect	Draws a rectangular.
16	fillRect	Draws a rectangular filled with a specified pattern.
17	drawCircle	Draws a circle.
18	fillCircle	Draws a circle filled with a specified pattern.
19	drawPolygon	Draws a polygon.
20	fillPolygon	Draws a polygon filled with a specified pattern.
21	embedRawDesignCommand	Embeds raw printer commands.

## 2. API specification

### 2.1. Return value

Return value	Description
CLS_SUCCESS (0)	The operation has been done successfully.
CLS_E_CONNECTED (1001)	The printer is already connected.
CLS_E_DISCONNECT (1002)	The printer is not connected.
CLS_E_NOTCONNECT (1003)	Failed to connect to the printer.
CLS_E_CONNECT_NOTFOUND (1004)	Failed to confirm the model name after connecting to the printer.
CLS_E_NOCONTEXT (1006)	No context is specified.
CLS_E_BT_DISABLE (1007)	The Bluetooth device setting is invalid.
CLS_E_BT_NODEVICE (1008)	The Bluetooth device is not found.
CLS_E_ILLEGAL (1101)	The operation is not supported by the printer or an invalid parameter was used.
CLS_E_OFFLINE (1102)	The printer is offline.
CLS_E_NOEXIST (1103)	The file name does not exist.
CLS_E_FAILURE (1104)	The service could not perform the requested procedure.
CLS_E_TIMEOUT (1105)	The service has been timed out waiting for a response from the printer.
CLS_E_NO_LIST (1106)	A printer could not be found for the printer search.
CLS_EPTR_BADFORMAT (1203)	The file format is not supported.

## 2.2. LabelPrinter class

### 2.2.1 Constructor

**Syntax**

LabelPrinter ()

**Parameters**

none

**Description**

Creates an instance of the LabelPrinter class.

**Return value**

none

**Example**

```
LabelPrinter printer = new LabelPrinter();
```



## 2.2.2 setContext method

### Syntax

```
int setContext( Context context )
```

### Parameters

値	[IN/OUT]	意味	設定可能範囲
context	[IN]	Context (android.content.Context)	Context :

### Description

This method is used to the USB connection and the search printer. Specify the context. Must be executed this method before the [connect method](#), the [searchCitizenPrinter method](#), the [searchLabelPrinter method](#), the [printerCheckEx method](#), and the [clearOutputEx method](#) with the USB connection.

If you want to connect with other interfaces, you do not need this method.

### Return value

Returns CLS\_SUCCESS(0) on success, an error code otherwise. See "[2.1 Return value](#)" for the error codes.

### Example

```
printer.setContext (this);
printer.setContext (getApplicationContext());
```

## 2.2.3 connect method

### Syntax

- 1) int connect (int connectType, String address)
- 2) int connect (int connectType, String address, int port)
- 3) int connect (int connectType, String address, int port, int timeout)
- 4) int connect (int connectType, UsbDevice device)

### Parameters

Parameter	[IN/OUT]	Description	Setting range
connectType	[IN]	Interface type	CLS_PORT_WiFi CLS_PORT_Bluetooth CLS_PORT_Bluetooth_Insecure CLS_PORT_USB
address	[IN]	IP address to connect or Bluetooth device address or Bluetooth device name	WiFi: 0.0.0.0~255.255.255.255 Bluetooth: 00:00:00:00:00:00~ FF:FF:FF:FF:FF:FF Device name (Automatic detection)
device	[IN]	UsbDevice class (android.hardware.usb.UsbDevice)	null: Automatic detection UsbDevice: Any USB device
port	[IN]	Port number	
timeout	[IN]	Timeout (msec)	

### Description

This method is to connect to a printer. Specify an interface type and address.

The Bluetooth device address must be written in upper case.

When the Bluetooth device name is specified, it looks for a paired device which matches up.

When the Bluetooth device name is not specified, it looks for a paired/supported device automatically.

In case of using the Insecure Bluetooth connection after Android 2.3.3, use

CLS\_PORT\_Bluetooth\_Insecure as connectType.

In case of USB device, the [setContext method](#) must be called beforehand.

The port number is effective only when CLS\_PORT\_WiFi is specified as connectType. The default port number is 9100.

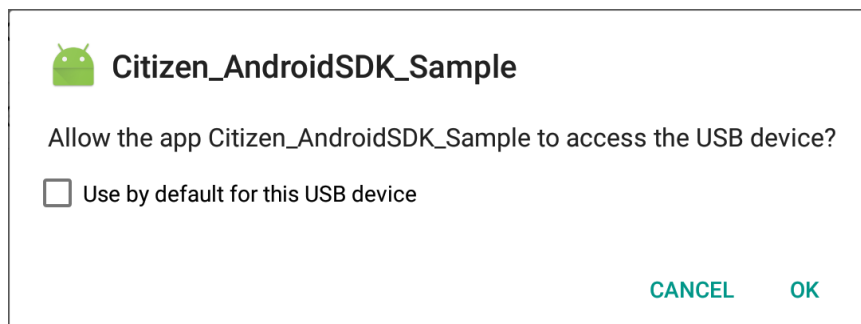
The timeout is to specify the maximum amount of time (in millisecond) to wait while attempting to connect to the printer. This is not effective when CLS\_PORT\_USB is specified as connectType. The default values are 4000 ms when WiFi, 8000 ms when Bluetooth.

When connecting, this method checks if the printer model is supported.

When the connection is no longer used, the disconnect method must be called. The next connection will be failed otherwise.

### Note

When connecting via USB for the first time, a dialog message for user permission pops up. Press OK to continue.



A dialog message for user permission (example).

**Return value**

Returns CLS\_SUCCESS (0) on success, an error code otherwise. See below for the error codes. "2.1 Return value" describes all error codes.

Error code	Description
CLS_E_CONNECTED (1001)	The printer is already connected.
CLS_E_NOTCONNECT (1003)	Failed to connect to the printer. (1) The printer is not connected. (2) The printer is not turned on. (3) The port handle could not be obtained.
CLS_E_CONNECT_NOTFOUND (1004)	Failed to confirm if the printer is a supported model. (1) The printer model is not supported.
CLS_E_NOCONTEXT (1006)	No context is specified.
CLS_E_BT_DISABLE (1007)	The Bluetooth device setting is invalid.
CLS_E_BT_NODEVICE (1008)	The Bluetooth device is not found.

**Example**

```
printer.connect(LabelConst.CLS_PORT_WiFi, "192.168.129.130");

printer.connect(LabelConst.CLS_PORT_Bluetooth, "");

UsbDevice usbDevice = null;
printer.connect(LabelConst.CLS_PORT_USB, usbDevice);
```

## 2.2.4 disconnect method

**Syntax**

```
int disconnect ()
```

**Parameters**

none

**Description**

Disconnects a printer.

This must be called when a connection is no longer needed or when an error occurs.

**Return value**

Returns CLS\_SUCCESS(0) on success, an error code otherwise. See "[2.1 Return value](#)" for the error codes.

**Example**

```
printer.disconnect();
```

## 2.2.5 printerCheck method

### Syntax

```
int printerCheck ()
```

### Parameters

none

### Description

Gets the status of the printer and stores information into the properties, [CommandInterpreterInAction](#), [PaperError](#), [RibbonEnd](#), [BatchProcessing](#), [Printing](#), [Pause](#), [WaitingForPeeling](#), [PrintHeadLowTemp](#), [PrintHeadFailure](#), [PrintHeadOverheat](#), [MechanismOpen](#), [AutoCutterError](#), [FanMotorError](#) and [MiscError](#). When this method returns an error, a communication failure or a device error might occur. The connection should be reestablished by using [disconnect/connect](#) methods in this case.

This method should be used to check the printer status before printing especially after a long idle.

This method can also be used to keep a network connection by calling it periodically.

In case of network connection with the CL - S5xx / 6xx / 70x, set "Parallel Error Output" to OFF beforehand. In case of parallel connection with all models, set "Parallel Error Output" to OFF. This can be done in the "Advanced" tab of the Label Printer Utility.

### Return value

Returns CLS\_SUCCESS (0) on success. See ["2.1 Return value"](#) for the error codes.

### Example

```
// PrinterCheck
if (LabelConst.CLS_SUCCESS == printer.printerCheck())
{
    // CommandInterpreterInAction Property
    if ( printer.getCommandInterpreterInAction() == 1 ) {
        // Command interpreter in action
    }
    // PaperError Property
    if ( printer.getPaperError() == 1 ) {
        // Paper error
    }
    // RibbonEnd Property
    if ( printer.getRibbonEnd() == 1 ) {
        // Ribbon end
    }
    // BatchProcessing Property
    if ( printer.getBatchProcessing() == 1 ) {
        // Batch processing
    }
    // Printing Property
    if ( printer.getPrinting() == 1 ) {
        // Printing
    }
    // Pause Property
    if ( printer.getPause() == 1 ) {
        // Pause
    }
    // WaitingForPeeling Property
    if ( printer.getWaitingForPeeling() == 1 ) {
        // Waitiong for peeling
    }
}
```

```
}

// PrintHeadLowTemp Property
if ( printer.getPrintHeadLowTemp() == 1 ) {
    // Print head low temp
}
// PrintHeadFailure Property
if ( printer.getPrintHeadFailure() == 1 ) {
    // Print head failure
}
// PrintHeadOverheat Property
if ( printer.getPrintHeadOverheat() == 1 ) {
    // Print head overheat
}
// MechanismOpen Property
if ( printer.getMechanismOpen() == 1 ) {
    // Mechanism open
}
// AutoCutterError Property
if ( printer.getAutoCutterError() == 1 ) {
    // Auto cutter error
}
// FanMotorError Property
if ( printer.getFanMotorError() == 1 ) {
    // Fan motor error
}
// MiscError Property
if ( printer.getMiscError() == 1 ) {
    // Misc error
}
}
else
{
    // Fail
}
```

## 2.2.6 print method

### Syntax

int print (LabelDesign design, int quantity)

### Parameters

Parameter	[IN/OUT]	Description	Setting range
design	[IN]	Instance of LabelDesignclass	
quantity	[IN]	Number of labels to print	1 - 9999

### Description

Prints labels by sending a label design created in the LabelDesign class, followed by printer configuration commands if any of properties are set.

### Return value

Returns CLS\_SUCCESS (0) on success. See "[2.1 Return value](#)" for the error codes.

### Example

```
design.fillCircle(50, 50, 50, LabelConst.CLS_SHADED_PTN_11);  
printer.print(design, 0001);
```

## 2.2.7 storeNVBitmap method

### Syntax

int storeNVBitmap (String filePath, String name, int rotation, int width, int height)

### Parameters

Parameter	[IN/OUT]	Description	Setting range
filePath	[IN]	Path to a bitmap file to store	
name	[IN]	File name to store	ASCII characters except below: -Underscore cannot be the first character. -The following symbols. ¥ / : * ? " < >
rotation	[IN]	Direction of rotation	CLS_RT_NORMAL : No rotate CLS_RT_RIGHT90 : Rotate right 90 CLS_RT_ROTATE180 : Rotate 180 CLS_RT_LEFT90 : Rotate left 90
width	[IN]	Horizontal size (pixels)	
height	[IN]	Vertical size (pixels)	

### Description

Stores a bitmap image into the flash memory with specified settings such as file name, rotation, width and height. The stored bitmap image can be used by the [drawNVBitmap method](#). Supported graphic file formats are BMP/GIF/EXIF/JPG/PNG/TIFF.

The image is resized by the specified width and height with keeping the original aspect ratio.

Example : The image size will be 200 x 50 pixels under the following conditions.

The original image size : 400 x 100 pixels

Width : 200

Height : 200

If 0 is set to either width or height, the image size will be calculated by another parameter.

Example : The image size will be 800 x 200" pixels under the following conditions.

The original image size : 400 x 100 pixels

Width : 0

Height : 200

### Return value

Returns CLS\_SUCCESS (0) on success. See ["2.1 Return value"](#) for the error codes.

### Example

```
// Store bitmap
printer.storeNVBitmap("/sdcard/temp/citizen/Images/Sample.bmp", "Sample",
    LabelConst.CLS_RT_NORMAL, 0, 0);

// Draw bitmap
design.drawNVBitmap("Sample", 1, 1, 0, 0);

// Print
printer.print(design, 0001);
```



## 2.2.8 clearOutPut method

### Syntax

int clearOutput ()

### Parameters

none

### Description

Clears data in the print buffer. This triggers the initialization process which is equivalent to the boot sequence.

### Return value

Returns CLS\_SUCCESS (0) on success. See "[2.1 Return value](#)" for the error codes.

### Example

```
printer.clearOutput();
```

## 2.2.9 sendData method

### Syntax

```
int sendData (byte[] data)
```

### Parameters

Parameter	[IN/OUT]	Description	Setting range
data	[IN]	Send data	

### Description

Sends binary data to a printer.

This can be used only when sending raw printer commands to the printer.

### Return value

Returns CLS\_SUCCESS (0) on success. See "[2.1 Return value](#)" for the error codes.

### Example

```
byte[] data = new byte[] {0x01, 0x23}; // [01]# (Reset)

printer.sendData(data);
```

## 2.2.10 searchCitizenPrinter method

### Syntax

CitizenPrinterInfo[] searchCitizenPrinter (int ifType, int searchTime, int[] result)

### Parameters

Parameter	[IN/OUT]	Description	Setting range
ifType	[IN]	Connection type	CLS_PORT_WiFi CLS_PORT_Bluetooth CLS_PORT_Bluetooth_Insecure
searchTime	[IN]	Time out (sec)	0: Get paired address in the case of CLS_PORT_Bluetooth or CLS_PORT_Bluetooth_Insecure. 1 - 30: Search for a specified time.
result	[OUT]	Error code	

The output parameter, result receives CLS\_SUCCESS (0) on success, an error code otherwise. See below for the error codes. "2.1 Return value" describes all error codes.

Error code	Description
CLS_E_NOCONTEXT (1006)	No context is specified.
CLS_E_ILLEGAL (1101)	Invalid parameter. (1) The connect type is unsupported. (2) The time out value is out of range.
CLS_E_NO_LIST (1106)	No printer found.

### Description

Searches available Citizen printers and obtains a list of printer information. The connection type and time out value must be specified. [setContext\(\)](#) must be called prior to searchCitizenPrinter(). This method doesn't work on the simulator.

After the specified time of searching process, it returns an array of printer information found, receives a result code to the output parameter, result.

In case of CLS\_PORT\_WiFi, the CL-S400/521/531/621/631/700/703 series and the CL-E720/730/300/303/321/331 series can be found. It may fail to find when the time out value is less than 3 seconds depending on the network environment.

In case of CLS\_PORT\_Bluetooth or CLS\_PORT\_Bluetooth\_Insecure as ifType, it returns addresses which are already paired when 0 is set to searchTime, it returns addresses which are available to connect when 1 to 30 is set to searchTime. 10 or above are recommended. It may fail to find if it is less than 10 depending on the Bluetooth condition.

### Return value

Returns a list of printer information when successful, returns empty otherwise.

The information will be stored in the CitizenPrinterInfo class type. Available information inside of the class is depending on the connection type.

Connection Type	Property	Available information
CLS_PORT_WiFi	ipAddress	IP Address
	macAddress	MAC Address
	deviceName	(Empty character)
	bdAddress	(Empty character)
CLS_PORT_Bluetooth	ipAddress	(Empty character)
	macAddress	(Empty character)
	deviceName	Bluetooth device name

	bdAddress	Bluetooth device address
--	-----------	--------------------------

**Example**

```
int[] errCode = new int[1];

CitizenPrinterInfo[] list = printer.searchCitizenPrinter(
    LabelConst.CLS_PORT_WiFi, 3, errCode);

for (int i = 0; i < list.length; i++)
{
    Log.e("Search", "IP Address: " + list[i].ipAddress);
    Log.e("Search", "MAC Address: " + list[i].macAddress);
}
```

## 2.2.11 searchLabelPrinter method

### Syntax

String[] searchLabelPrinter (int ifType, int searchTime, int[] result)

### Parameters

Parameter	[IN/OUT]	Description	Setting range
ifType	[IN]	Connection type	CLS_PORT_WiFi CLS_PORT_Bluetooth CLS_PORT_Bluetooth_Insecure
searchTime	[IN]	Time out (sec)	0: Get paired address in the case of CLS_PORT_Bluetooth or CLS_PORT_Bluetooth_Insecure. 1 - 30: Search for a specified time.
result	[OUT]	Error code	

The output parameter, result receives CLS\_SUCCESS (0) on success, an error code otherwise. See below for the error codes. "2.1 Return value" describes all error codes.

Error code	Description
CLS_E_NOCONTEXT (1006)	No context is specified.
CLS_E_ILLEGAL (1101)	Invalid parameter. (1) The connect type is unsupported. (2) The time out value is out of range.
CLS_E_NO_LIST (1106)	No printer found.

### Description

Searches available Citizen printers and obtains a list of printer addresses. The connection type and time out value must be specified. [setContext\(\)](#) must be called prior to searchLabelPrinter(). This method doesn't work on the simulator.

After the specified time of searching process, it returns an array of printer addresses found, receives a result code to the output parameter, result.

In case of CLS\_PORT\_WiFi, the CL-S400/521/531/621/631/700/703 series and the CL-E720/730/300/303/321/331 series can be found. It may fail to find when the time out value is less than 3 seconds depending on the network environment.

In case of CLS\_PORT\_Bluetooth or CLS\_PORT\_Bluetooth\_Insecure as ifType, it returns addresses which are already paired when 0 is set to searchTime, it returns addresses which are available to connect when 1 to 30 is set to searchTime. 10 or above are recommended. It may fail to find if it is less than 10 depending on the Bluetooth condition.

### Return value

Returns a list of printer addresses when successful, returns empty otherwise.

### Example

```
int[] errCode = new int[1];

String[] list = printer.searchLabelPrinter(
    LabelConst.CLS_PORT_WiFi, 3, errCode);
```

## 2.2.12 printerCheckEx method

### Syntax

- 1) int printerCheckEx (int connectType, String address)
- 2) int printerCheckEx (int connectType, String address, int port)
- 3) int printerCheckEx (int connectType, String address, int port, int timeout)
- 4) int printerCheckEx (int connectType, UsbDevice device)

### Parameters

Parameter	[IN/OUT]	Description	Setting range
connectType	[IN]	Interface type	CLS_PORT_WiFi CLS_PORT_Bluetooth CLS_PORT_Bluetooth_Insecure CLS_PORT_USB
address	[IN]	IP address to connect or Bluetooth device address or Bluetooth device name	WiFi: 0.0.0.0~255.255.255.255 Bluetooth: 00:00:00:00:00:00~ FF:FF:FF:FF:FF:FF Device name (Automatic detection)
device	[IN]	UsbDevice class (android.hardware.usb.UsbDevice)	null: Automatic detection UsbDevice: Any USB device
port	[IN]	Port number	
timeout	[IN]	Timeout (msec)	

### Description

After connecting to a printer, gets the status of the printer and stores information into the properties, [CommandInterpreterInAction](#), [PaperError](#), [RibbonEnd](#), [BatchProcessing](#), [Printing](#), [Pause](#), [WaitingForPeeling](#), [PrintHeadLowTemp](#), [PrintHeadFailure](#), [PrintHeadOverheat](#), [MechanismOpen](#), [AutoCutterError](#), [FanMotorError](#) and [MiscError](#). Specify an interface type and address/UsbDevice. Disconnect the connection after the process is complete.

This method can connect and process regardless of the printer's status.

The Bluetooth device address must be written in upper case.

When the Bluetooth device name is specified, it looks for a paired device which matches up.

When the Bluetooth device name is not specified, it looks for a paired/supported device automatically.

In case of using the Insecure Bluetooth connection after Android 2.3.3, use

CLS\_PORT\_Bluetooth\_Insecure as connectType.

In case of USB device, the [setContext method](#) must be called beforehand.

The port number is effective only when CLS\_PORT\_WiFi is specified as connectType. The default port number is 9100.

The timeout is to specify the maximum amount of time (in millisecond) to wait while attempting to connect to the printer. This is not effective when CLS\_PORT\_USB is specified as connectType. The default values are 4000 ms when WiFi, 8000 ms when Bluetooth.

When connecting, this method checks if the printer model is supported.

When the connection is no longer used, the disconnect method must be called. The next connection will be failed otherwise.

### Return value

Returns CLS\_SUCCESS (0) on success, an error code otherwise. See below for the error codes.

"[2.1 Return value](#)" describes all error codes.

Error code	Description
CLS_E_CONNECTED (1001)	The printer is already connected.
CLS_E_NOTCONNECT (1003)	Failed to connect to the printer. (1) The printer is not connected.

	(2) The printer is not turned on. (3) The port handle could not be obtained.
CLS_E_CONNECT_NOTFOUND (1004)	Failed to confirm if the printer is a supported model. (1) The printer model is not supported.
CLS_E_NOCONTEXT (1006)	No context is specified.
CLS_E_BT_DISABLE (1007)	The Bluetooth device setting is invalid.
CLS_E_BT_NODEVICE (1008)	The Bluetooth device is not found.

**Example**

```
// PrinterCheckEx
if (LabelConst.CLS_SUCCESS ==
    printer.printerCheckEx(LabelConst.CLS_PORT_WiFi, "192.168.129.130"))
{
    // CommandInterpreterInAction Property
    if ( printer.getCommandInterpreterInAction() == 1 ) {
        // Command interpreter in action
    }
    // PaperError Property
    if ( printer.getPaperError() == 1 ) {
        // Paper error
    }
    // RibbonEnd Property
    if ( printer.getRibbonEnd() == 1 ) {
        // Ribbon end
    }
    // BatchProcessing Property
    if ( printer.getBatchProcessing() == 1 ) {
        // Batch processing
    }
    // Printing Property
    if ( printer.getPrinting() == 1 ) {
        // Printing
    }
    // Pause Property
    if ( printer.getPause() == 1 ) {
        // Pause
    }
    // WaitingForPeeling Property
    if ( printer.getWaitingForPeeling() == 1 ) {
        // Waitiong for peeling
    }

    // PrintHeadLowTemp Property
    if ( printer.getPrintHeadLowTemp() == 1 ) {
        // Print head low temp
    }
    // PrintHeadFailure Property
    if ( printer.getPrintHeadFailure() == 1 ) {
        // Print head failure
    }
    // PrintHeadOverheat Property
    if ( printer.getPrintHeadOverheat() == 1 ) {
        // Print head overheat
    }
    // MechanismOpen Property
    if ( printer.getMechanismOpen() == 1 ) {
        // Mechanism open
    }
}
```

```
// AutoCutterError Property
if ( printer.getAutoCutterError() == 1 ) {
    // Auto cutter error
}
// FanMotorError Property
if ( printer.getFanMotorError() == 1 ) {
    // Fan motor error
}
// MiscError Property
if ( printer.getMiscError() == 1 ) {
    // Misc error
}
}
else
{
    // Fail
}
```



## 2.2.13 clearOutputEx method

### Syntax

- 1) int clearOutputEx (int connectType, String address)
- 2) int clearOutputEx (int connectType, String address, int port)
- 3) int clearOutputEx (int connectType, String address, int port, int timeout)
- 4) int clearOutputEx (int connectType, UsbDevice device)

### Parameters

Parameter	[IN/OUT]	Description	Setting range
connectType	[IN]	Interface type	CLS_PORT_WiFi CLS_PORT_Bluetooth CLS_PORT_Bluetooth_Insecure CLS_PORT_USB
address	[IN]	IP address to connect or Bluetooth device address or Bluetooth device name	WiFi: 0.0.0.0~255.255.255.255 Bluetooth: 00:00:00:00:00:00~ FF:FF:FF:FF:FF:FF Device name (Automatic detection)
device	[IN]	UsbDevice class (android.hardware.usb.UsbDevice)	null: Automatic detection UsbDevice: Any USB device
port	[IN]	Port number	
timeout	[IN]	Timeout (msec)	

### Description

After connecting to a printer, reset the printer. This triggers the initialization process which is equivalent to the boot sequence. Specify an interface type and address/UsbDevice. Disconnect the connection after the process is complete.

This method can connect and process regardless of the printer's status.

The Bluetooth device address must be written in upper case.

When the Bluetooth device name is specified, it looks for a paired device which matches up.

When the Bluetooth device name is not specified, it looks for a paired/supported device automatically.

In case of using the Insecure Bluetooth connection after Android 2.3.3, use

CLS\_PORT\_Bluetooth\_Insecure as connectType.

In case of USB device, the [setContext method](#) must be called beforehand.

The port number is effective only when CLS\_PORT\_WiFi is specified as connectType. The default port number is 9100.

The timeout is to specify the maximum amount of time (in millisecond) to wait while attempting to connect to the printer. This is not effective when CLS\_PORT\_USB is specified as connectType. The default values are 4000 ms when WiFi, 8000 ms when Bluetooth.

When connecting, this method checks if the printer model is supported.

When the connection is no longer used, the disconnect method must be called. The next connection will be failed otherwise.

### Return value

Returns CLS\_SUCCESS (0) on success, an error code otherwise. See below for the error codes.

"2.1 Return value" describes all error codes.

Error code	Description
CLS_E_CONNECTED (1001)	The printer is already connected.
CLS_E_NOTCONNECT (1003)	Failed to connect to the printer. (1) The printer is not connected. (2) The printer is not turned on. (3) The port handle could not be obtained.

CLS_E_CONNECT_NOTFOUND (1004)	Failed to confirm if the printer is a supported model. (1) The printer model is not supported.
CLS_E_NOCONTEXT (1006)	No context is specified.
CLS_E_BT_DISABLE (1007)	The Bluetooth device setting is invalid.
CLS_E_BT_NODEVICE (1008)	The Bluetooth device is not found.

**Example**

```
printer.clearOutputEx(LabelConst.CLS_PORT_WiFi, "192.168.129.130");
```

## 2.2.14 setLog method

### Syntax

```
void setLog (int mode, String path, int maxSize)
```

### Parameters

Parameter	[IN/OUT]	Description	Setting range
mode	[IN]	Logging mode	0: None 1: Access logs 2: Error logs
path	[IN]	File path to store	The folder of SD card
maxSize	[IN]	Maximum Log Size	0: Unlimited 1 - : Maximum size (MB)

### Description

Sets the logging function. See "[3.2 Logging function](#)" for more details.

For the output folder, specify the path to a folder that the application that outputs the log file can write to. If a full path other than the one beginning with '/storage/' is specified, it will be specified as the path from the top of the external storage path.

### Return value

none

### Example

```
// Specifying the folder for the application's specific area.
printer.setLog(1, this.getExternalFilesDir(null).toString() + "/Log", 10);

// Specifying the public folder (Download).
printer.setLog(1, Environment.getExternalStoragePublicDirectory(
    Environment.DIRECTORY_DOWNLOADS) + "/Log", 10);
```

## 2.2.15 HorizontalMagnification property

### Syntax

int HorizontalMagnification

### Attribute

Read/Write

### Description

Horizontal dot size. "1" or "2".

### Setter method

int setHorizontalMagnification (int horizontalMagnification)

"1" or "2" must be set as a parameter.

Returns CLS\_SUCCESS (0) on success. See ["2.1 Return value"](#) for the error codes.

### Getter method

int getHorizontalMagnification ()

### Example

```
int pixelSize;  
  
pixelSize = printer.getHorizontalMagnification();  
printer.setHorizontalMagnification(2);
```

## 2.2.16 VerticalMagnification property

### Syntax

int VerticalMagnification

### Attribute

Read/Write

### Description

Vertical dot size. "1", "2" or "3".

### Setter method

int setVerticalMagnification (int verticalMagnification)

"1", "2" or "3" must be set as a parameter.

Returns CLS\_SUCCESS (0) on success. See ["2.1 Return value"](#) for the error codes.

### Getter method

int getVerticalMagnification ()

### Example

```
int pixelSize;

printer.setVerticalMagnification(3);
pixelSize = printer.getVerticalMagnification();
```

## 2.2.17 FormatAttribute property

### Syntax

int FormatAttribute

### Attribute

Read/Write

### Description

Specifies how to print the area where multiple objects are overlapped.

0: XOR mode. Overlapped area will not be printed.

1: OR mode. Overlapped area will be printed.

### Setter method

int setFormatAttribute (int formatAttribute)

Returns CLS\_SUCCESS (0) on success. See "[2.1 Return value](#)" for the error codes.

### Getter method

int getFormatAttribute ()

Returns the set value. If nothing has been set, it returns CLS\_PROPERTY\_DEFAULT(999999).

### Example

```
int attr;

printer.setFormatAttribute(1);
attr = printer.getFormatAttribute();
```

### Printing Result



## 2.2.18 ContinuousMediaLength property

### Syntax

int ContinuousMediaLength

### Attribute

Read/Write

### Description

Sets a paper length for continuous paper.

Inch system	0001 – 9999 (0.01 - 99.99 inches)
Metric system	0001 – 9999 (0.1 - 999.9 mm)

### Setter method

int setContinuousMediaLength (int continuousMediaLength)

The default value in the printer is "0000."

Returns CLS\_SUCCESS (0) on success. See "[2.1 Return value](#)" for the error codes.

### Getter method

int getContinuousMediaLength ()

Returns the set value. If nothing has been set, it returns CLS\_PROPERTY\_DEFAULT(999999).

### Example

```
int mediaLength;  
  
printer.setContinuousMediaLength(2000);  
mediaLength = printer.getContinuousMediaLength();
```

## 2.2.19 MeasurementUnit property

### Syntax

int MeasurementUnit

### Attribute

Read/Write

### Description

Sets the measurement unit.

Value	Description
CLS_UNIT_MILLI (0)	Metric system
CLS_UNIT_INCH (1)	Inch system

### Setter method

int setMeasurementUnit (int measurementUnit)

The default value in the printer is "1."

Returns CLS\_SUCCESS (0) on success. See ["2.1 Return value"](#) for the error codes.

### Getter method

int getMeasurementUnit ()

Returns the set value. If nothing has been set, it returns CLS\_PROPERTY\_DEFAULT(999999).

### Example

```
int unit;
```

```
printer.setMeasurementUnit (LabelConst.CLS_UNIT_MILLI);  
unit = printer.getMeasurementUnit();
```



## 2.2.20 PrintSpeed property

### Syntax

int PrintSpeed

### Attribute

Read/Write

### Description

Sets the print speed.

See "[3.3.Predefined Constants](#)", No.21 for available value.

\*Initial and maximum value vary depending on the printer model.

### Setter method

int setPrintSpeed (int printSpeed)

Returns CLS\_SUCCESS (0) on success. See "[2.1 Return value](#)" for the error codes.

### Getter method

int getPrintSpeed ()

Returns the set value. If nothing has been set, it returns CLS\_PROPERTY\_DEFAULT(999999).

### Example

```
int printSpeed;  
  
printer.setPrintSpeed(LabelConst.CLS_SPEEDSETTING_X);  
printSpeed = printer.getPrintSpeed();
```

## 2.2.21 FeedSpeed property

### Syntax

int FeedSpeed

### Attribute

Read/Write

### Description

Sets the feed speed.

See "3.3.Predefined Constants", No.21 for available value.

\*Initial and maximum value vary depending on the printer model.

### Setter method

int setFeedSpeed (int feedSpeed)

Returns CLS\_SUCCESS (0) on success. See "2.1 Return value" for the error codes.

### Getter method

int getFeedSpeed ()

Returns the set value. If nothing has been set, it returns CLS\_PROPERTY\_DEFAULT(999999).

### Example

```
int feedSpeed;  
  
printer.setFeedSpeed(LabelConst.CLS_SPEEDSETTING_W);  
feedSpeed = printer.getFeedSpeed();
```

## 2.2.22 SlewSpeed property

### Syntax

int SlewSpeed

### Attribute

Read/Write

### Description

Sets the slew speed.

See "3.3.Predefined Constants", No.21 for available value.

\*Initial and maximum value vary depending on the printer model.

### Setter method

int setSlewSpeed (int slewSpeed)

Returns CLS\_SUCCESS (0) on success. See "2.1 Return value" for the error codes.

### Getter method

int getSlewSpeed ()

Returns the set value. If nothing has been set, it returns CLS\_PROPERTY\_DEFAULT(999999).

### Example

```
int slewSpeed;  
  
printer.setSlewSpeed(LabelConst.CLS_SPEEDSETTING_V);  
slewSpeed = printer.getSlewSpeed();
```

## 2.2.23 BackupSpeed property

### Syntax

int BackupSpeed

### Attribute

Read/Write

### Description

Sets the backup speed. See below for the acceptable values.

Value	Description
CLS_SPEEDSETTING_A or CLS_SPEEDSETTING_B	1.0 inch( 25.4mm) / sec
CLS_SPEEDSETTING_C or CLS_SPEEDSETTING_D	2.0 inch( 50.8mm) / sec
CLS_SPEEDSETTING_E or CLS_SPEEDSETTING_F	3.0 inch( 76.2mm) / sec
CLS_SPEEDSETTING_G or CLS_SPEEDSETTING_H	4.0 inch(101.6mm) / sec
CLS_SPEEDSETTING_I or CLS_SPEEDSETTING_J	5.0 inch(127.0mm) / sec
CLS_SPEEDSETTING_K or CLS_SPEEDSETTING_L	6.0 inch(152.4mm) / sec
CLS_SPEEDSETTING_M or CLS_SPEEDSETTING_N	7.0 inch(177.8mm) / sec
CLS_SPEEDSETTING_O	8.0 inch(203.2mm) / sec
CLS_SPEEDSETTING_1 - CLS_SPEEDSETTING_8	1.0 inch( 25.4mm) / sec – 8.0 inch(203.2mm) / sec

\*Initial and maximum value vary depending on the printer model.

### Setter method

int setBackupSpeed (int backupSpeed)

Returns CLS\_SUCCESS (0) on success. See ["2.1 Return value"](#) for the error codes.

### Getter method

int getBackupSpeed ()

Returns the set value. If nothing has been set, it returns CLS\_PROPERTY\_DEFAULT(999999).

### Example

```
int backupSpeed;

printer.setBackupSpeed(LabelConst.CLS_SPEEDSETTING_O);
backupSpeed = printer.getBackupSpeed();
```

## 2.2.24 PrintDarkness property

### Syntax

int PrintDarkness

### Attribute

Read/Write

### Description

Sets the print density. The acceptable values are below:

Setting range	0 - 30
Default value	10

### Setter method

int setPrintDarkness (int printDarkness)

Returns CLS\_SUCCESS (0) on success. See "[2.1 Return value](#)" for the error codes.

### Getter method

int getPrintDarkness ()

Returns the set value. If nothing has been set, it returns CLS\_PROPERTY\_DEFAULT(999999).

### Example

```
int printDarkness;

printer.setPrintDarkness(30);
printDarkness = printer.getPrintDarkness();
```

## 2.2.25 DoubleHeat property

### Syntax

int DoubleHeat

### Attribute

Read/Write

### Description

Sets the double heat option. The acceptable values are below:

0: OFF (Default)

1: ON

### Setter method

int setDoubleHeat (int doubleHeat)

Returns CLS\_SUCCESS (0) on success. See "[2.1 Return value](#)" for the error codes.

### Getter method

int getDoubleHeat ()

Returns the set value. If nothing has been set, it returns CLS\_PROPERTY\_DEFAULT(999999).

### Example

```
int doubleHeat;  
  
printer.setDoubleHeat(1);  
doubleHeat = printer.getDoubleHeat();
```

## 2.2.26 VerticalOffset property

### Syntax

int VerticalOffset

### Attribute

Read/Write

### Description

Sets the vertical offset to adjust the vertical printing position on the paper.

Inch system	0000 – 9999 (0.00 inch - 99.99 inches)
Metric system	0000 – 9999 (0.0 mm - 999.9 mm)
Default value	0000

### Setter method

int setVerticalOffset (int verticalOffset)

Returns CLS\_SUCCESS (0) on success. See "[2.1 Return value](#)" for the error codes.

### Getter method

int getVerticalOffset ()

Returns the set value. If nothing has been set, it returns CLS\_PROPERTY\_DEFAULT(999999).

### Example

```
int verticalOffset;  
  
printer.setVerticalOffset(10);  
verticalOffset = printer.getVerticalOffset();
```

## 2.2.27 HorizontalOffset property

### Syntax

int HorizontalOffset

### Attribute

Read/Write

### Description

Sets the horizontal offset to adjust the horizontal printing position on the paper.

Inch system	0000 – 9999 (0.00 inch - 99.99 inches)
Metric system	0000 – 9999 (0.0 mm - 999.9 mm)
Default value	0000

### Setter method

int setHorizontalOffset (int horizontalOffset)

Returns CLS\_SUCCESS (0) on success. See "[2.1 Return value](#)" for the error codes.

### Getter method

int getHorizontalOffset ()

Returns the set value. If nothing has been set, it returns CLS\_PROPERTY\_DEFAULT(999999).

### Example

```
int horizontalOffset;  
  
printer.setHorizontalOffset(20);  
horizontalOffset = printer.getHorizontalOffset();
```



## 2.2.28 MediaHandling property

### Syntax

int MediaHandling

### Attribute

Read/Write

### Description

Sets the way how to handle media after printing.

Value	Description
CLS_MEDIAHANDLING_NONE (0)	To print without any special actions.
CLS_MEDIAHANDLING_TEAROFF (1)	To feed labels to the tear bar.
CLS_MEDIAHANDLING_DISPENSES (2)	To feed labels to the tear bar, and wait for removal.
CLS_MEDIAHANDLING_PAUSE (3)	To pause after printing.
CLS_MEDIAHANDLING_CUT (4)	To cut after printing.
CLS_MEDIAHANDLING_CUTANDPAUSE (5)	To cut and pause after printing.
CLS_MEDIAHANDLING_PEELOFF (6)	To peel labels off the backing and wait for removal.
CLS_MEDIAHANDLING_REWIND (7)	To use a rewinder.

### Setter method

int setMediaHandling (int mediaHandling)

Returns CLS\_SUCCESS (0) on success. See ["2.1 Return value"](#) for the error codes.

### Getter method

int getMediaHandling ()

Returns the set value. If nothing has been set, it returns CLS\_PROPERTY\_DEFAULT(999999).

### Example

```
int mediaHandling;

printer.setMediaHandling(LabelConst.CLS_MEDIAHANDLING_PAUSE);
mediaHandling = printer.getMediaHandling();
```

## 2.2.29 StartOffset property

### Syntax

int StartOffset

### Attribute

Read/Write

### Description

Specifies the distance between paper sensor and print head to change the print starting position.

Inch system			Metric system		
Initial value	Minimum value	Max value	Initial value	Minimum value	Max value
0220	0120	0320	0559	0305	0813

\*0.01 inches or 0.1 mm

### Setter method

int setStartOffset(int startOffset)

Make sure to set a valid value since the consistency with the selected measurement unit will never be checked.

Returns CLS\_SUCCESS (0) on success. See "[2.1 Return value](#)" for the error codes.

### Getter method

int getStartOffset ()

Returns the set value. If nothing has been set, it returns CLS\_PROPERTY\_DEFAULT(999999).

### Example

```
int startOffset;

printer.setStartOffset(220);
startOffset = printer.getStartOffset();
```

## 2.2.30 StopOffset property

### Syntax

int StopOffset

### Attribute

Read/Write

### Description

Specifies the distance between paper sensor and cutter or peeler to change the print stop position.

Note that the initial values are calculated to stop paper at an appropriate position. An insufficient value truncates the printed label, an exceeded value truncates the next label.

Media handling	Inch system			Metric system		
	Initial value	Minimum value	Max value	Initial value	Minimum value	Max value
None	0000	0000	9999	0000	0000	9999
Cutter	0100			0254		
Peel Off	0050			0127		
Tear Off	0070			0178		

\*0.01 inches or 0.1 mm

### Setter method

int setStopOffset (int stopOffset)

Returns CLS\_SUCCESS (0) on success. See ["2.1 Return value"](#) for the error codes.

### Getter method

int getStopOffset ()

Returns the set value. If nothing has been set, it returns CLS\_PROPERTY\_DEFAULT(999999).

### Example

```
int stopOffset;

printer.setStopOffset(240);
stopOffset = printer.getStopOffset();
```

## 2.2.31 LabelSensor property

### Syntax

int LabelSensor

### Attribute

Read/Write

### Description

Sets the label sensor type.

Value	Description
CLS_SELSENSOR_NONE (0)	None. This assumes that a continuous paper roll is used. Therefore a valid length must be set in the <a href="#">ContinuousMediaLength property</a> , returns CLS_E_ILLEGAL (1101) otherwise.
CLS_SELSENSOR_SEETHROUGH (1)	Gap sensor. To detect a gap between labels or a notch on tags.
CLS_SELSENSOR_REFLECT (2)	Reflect sensor. Detect a black mark on the back side of the label.

### Setter method

int setLabelSensor (int labelSensor)

Returns CLS\_SUCCESS (0) on success. See "[2.1 Return value](#)" for the error codes.

### Getter method

int getLabelSensor ()

Returns the set value. If nothing has been set, it returns CLS\_PROPERTY\_DEFAULT(999999).

### Example

```
// Other than "CLS_SELSENSOR_NONE"
int labelSensor;

printer.setLabelSensor(LabelConst.CLS_SELSENSOR_SEETHROUGH);
labelSensor = printer.getLabelSensor();

// "CLS_SELSENSOR_NONE"
int labelSensor;

printer.setContinuousMediaLength(100);
printer.setLabelSensor(LabelConst.CLS_SELSENSOR_NONE);
labelSensor = printer.getLabelSensor();
```

## 2.2.32 PrintMethod property

### Syntax

int PrintMethod

### Attribute

Read/Write

### Description

Sets the print method to either thermal transfer or direct thermal.

Value	Description
CLS_PRTMETHOD_TT (0)	Thermal transfer
CLS_PRTMETHOD_DT (1)	Direct thermal

### Setter method

int setPrintMethod (int printMethod)

Returns CLS\_SUCCESS (0) on success. See "[2.1 Return value](#)" for the error codes.

### Getter method

int getPrintMethod ()

Returns the set value. If nothing has been set, it returns CLS\_PROPERTY\_DEFAULT(999999).

### Example

```
int printMethod;  
  
printer.setPrintMethod(LabelConst.CLS_PRTMETHOD_DT);  
printMethod = printer.getPrintMethod();
```

## 2.2.33 SensorLocation property

### Syntax

int SensorLocation

### Attribute

Read/Write

### Description

Selects the sensor to use for the models which have multiple paper sensors, front and rear. The choice will be stored in the non-volatile flash memory and will be kept until overwritten.

Value	Description
CLS_SENS_LOCATION_FRONT (0)	Front sensor
CLS_SENS_LOCATION_ADJUSTABLE (1)	Rear sensor

### Setter method

int setSensorLocation (int sensorLocation)

Returns CLS\_SUCCESS (0) on success. See ["2.1 Return value"](#) for the error codes.

### Getter method

int getSensorLocation ()

Returns the set value. If nothing has been set, it returns CLS\_PROPERTY\_DEFAULT(999999).

### Example

```
int location;

printer.setSensorLocation(LabelConst.CLS_SENS_LOCATION_ADJUSTABLE);
location = printer.getSensorLocation();
```

## 2.2.34 CommandInterpreterInAction property

### Syntax

int CommandInterpreterInAction

### Attribute

Read

### Description

Shows the printer status if the interpreter is processing commands.

Note that the status properties won't be updated automatically.

[printerCheck method](#) checks the status and updates the value, must be called before referring to this property.

### Setter method

none

### Getter method

int getCommandInterpreterInAction ()

Returns one of the following values.

Value	Description
0	No
1	Yes
CLS_PROPERTY_DEFAULT(999999)	Nothing has been set.

### Example

See [printerCheck method](#).

## 2.2.35 PaperError property

### Syntax

int PaperError

### Attribute

Read

### Description

Shows the printer status if there is a paper error.

Note that the status properties won't be updated automatically.

[printerCheck method](#) checks the status and updates the value, must be called before referring to this property.

### Setter method

none

### Getter method

int getPaperError ()

Returns one of the following values.

Value	Description
0	No
1	Yes
CLS_PROPERTY_DEFAULT(999999)	Nothing has been set.

### Example

See [printerCheck method](#).



## 2.2.36 RibbonEnd property

### Syntax

int RibbonEnd

### Attribute

Read

### Description

Shows the printer status if there is a ribbon end error.

Note that the status properties won't be updated automatically.

[printerCheck method](#) checks the status and updates the value, must be called before referring to this property.

### Setter method

none

### Getter method

int getRibbonEnd ()

Returns one of the following values.

Value	Description
0	No
1	Yes
CLS_PROPERTY_DEFAULT(999999)	Nothing has been set.

### Example

See [printerCheck method](#).

## 2.2.37 BatchProcessing property

### Syntax

int BatchProcessing

### Attribute

Read

### Description

Shows the status if the printer is processing batch print jobs.

Note that the status properties won't be updated automatically.

[printerCheck method](#) checks the status and updates the value, must be called before referring to this property.

### Setter method

none

### Getter method

int getBatchProcessing ()

Returns one of the following values.

Value	Description
0	No
1	Yes
CLS_PROPERTY_DEFAULT(999999)	Nothing has been set.

### Example

See [printerCheck method](#).

## 2.2.38 Printing property

### Syntax

int Printing

### Attribute

Read

### Description

Shows the status if the printer is currently printing.

Note that the status properties won't be updated automatically.

[printerCheck method](#) checks the status and updates the value, must be called before referring to this property.

### Setter method

none

### Getter method

int getPrinting ()

Returns one of the following values.

Value	Description
0	No
1	Yes
CLS_PROPERTY_DEFAULT(999999)	Nothing has been set.

### Example

See [printerCheck method](#).

## 2.2.39 Pause property

### Syntax

int Pause

### Attribute

Read

### Description

Shows the status if the printer is in the pause mode.

Note that the status properties won't be updated automatically.

[printerCheck method](#) checks the status and updates the value, must be called before referring to this property.

### Setter method

none

### Getter method

int getPause ()

Returns one of the following values.

Value	Description
0	No
1	Yes
CLS_PROPERTY_DEFAULT(999999)	Nothing has been set.

### Example

See [printerCheck method](#).

## 2.2.40 WaitingForPeeling property

### Syntax

int WaitingForPeeling

### Attribute

Read

### Description

Shows the status if the printer is in the wait for peeling status.

Note that the status properties won't be updated automatically.

[printerCheck method](#) checks the status and updates the value, must be called before referring to this property.

### Setter method

none

### Getter method

int getWaitingForPeeling ()

Returns one of the following values.

Value	Description
0	No
1	Yes
CLS_PROPERTY_DEFAULT(999999)	Nothing has been set.

### Example

See [printerCheck method](#).

## 2.2.41 PrintHeadLowTemp property

### Syntax

int PrintHeadLowTemp

### Attribute

Read

### Description

Shows the printer status if there is a print head low temp error.

Note that the status properties won't be updated automatically.

[printerCheck method](#) checks the status and updates the value, must be called before referring to this property.

### Setter method

none

### Getter method

int GetPrintHeadLowTemp ()

Returns one of the following values.

Value	Description
0	No
1	Yes
CLS_PROPERTY_DEFAULT(999999)	Nothing has been set.

### Example

See [printerCheck method](#).

## 2.2.42 PrintHeadFailure property

### Syntax

int PrintHeadFailure

### Attribute

Read

### Description

Shows the printer status if there is a print head failure.

Note that the status properties won't be updated automatically.

[PrinterCheck method](#) checks the status and updates the value, must be called before referring to this property.

### Setter method

none

### Getter method

int GetPrintHeadFailure ()

Returns one of the following values.

Value	Description
0	No
1	Yes
CLS_PROPERTY_DEFAULT(999999)	Nothing has been set.

### Example

See [PrinterCheck method](#).

## 2.2.43 PrintHeadOverheat property

### Syntax

int PrintHeadOverheat

### Attribute

Read

### Description

Shows the printer status if there is a print head over heat error.

Note that the status properties won't be updated automatically.

[PrinterCheck method](#) checks the status and updates the value, must be called before referring to this property.

### Setter method

none

### Getter method

int GetPrintHeadOverheat ()

Returns one of the following values.

Value	Description
0	No
1	Yes
CLS_PROPERTY_DEFAULT(999999)	Nothing has been set.

### Example

See [PrinterCheck method](#).



## 2.2.44 MechanismOpen property

### Syntax

int MechanismOpen

### Attribute

Read

### Description

Shows the printer status if there is a mechanism open error.

Note that the status properties won't be updated automatically.

[PrinterCheck method](#) checks the status and updates the value, must be called before referring to this property.

### Setter method

none

### Getter method

int GetMechanismOpen ()

Returns one of the following values.

Value	Description
0	No
1	Yes
CLS_PROPERTY_DEFAULT(999999)	Nothing has been set.

### Example

See [PrinterCheck method](#).

## 2.2.45 AutoCutterError property

### Syntax

int AutoCutterError

### Attribute

Read

### Description

Shows the printer status if there is an auto cutter error.

Note that the status properties won't be updated automatically.

[PrinterCheck method](#) checks the status and updates the value, must be called before referring to this property.

### Setter method

none

### Getter method

int GetAutoCutterError ()

Returns one of the following values.

Value	Description
0	No
1	Yes
CLS_PROPERTY_DEFAULT(999999)	Nothing has been set.

### Example

See [PrinterCheck method](#).

## 2.2.46 FanMotorError property

### Syntax

int FanMotorError

### Attribute

Read

### Description

Shows the printer status if there is a fan motor error.

Note that the status properties won't be updated automatically.

[PrinterCheck method](#) checks the status and updates the value, must be called before referring to this property.

### Setter method

none

### Getter method

int GetFanMotorError ()

Returns one of the following values.

Value	Description
0	No
1	Yes
CLS_PROPERTY_DEFAULT(999999)	Nothing has been set.

### Example

See [PrinterCheck method](#).

## 2.2.47 MiscError property

### Syntax

int MiscError

### Attribute

Read

### Description

Shows the printer status if there is a misc error.

Note that the status properties won't be updated automatically.

[PrinterCheck method](#) checks the status and updates the value, must be called before referring to this property.

### Setter method

none

### Getter method

int GetMiscError ()

Returns one of the following values.

Value	Description
0	No
1	Yes
CLS_PROPERTY_DEFAULT(999999)	Nothing has been set.

### Example

See [PrinterCheck method](#).

## 2.3. LabelDesign class

### 2.3.1 Constructor

**Syntax**

LabelDesign ()

**Parameters**

none

**Description**

Creates an instance of the LabelDesign class.

**Return value**

none

**Example**

```
LabelDesign design = new LabelDesign();
```

## 2.3.2 drawTextPtrFont method

### Syntax

- 1) int drawTextPtrFont (String data, int locale,  
int font, int rotation, int hexp, int vexp, int size, int x, int y)
- 2) int drawTextPtrFont (String data, int locale,  
int font, int rotation, int hexp, int vexp, int size, int x, int y, int encoding)

### Parameters

Parameter	[IN/OUT]	Description	Setting range
data	[IN]	String data	* Unsupported characters will be replaced with blank text.
locale	[IN]	Locale	CLS_LOCALE_JP: Japan CLS_LOCALE_OTHER: Other CLS_LOCALE_CN: Chinese CLS_LOCALE_KR: Korean
font	[IN]	Font typeface	CLS_PRT_FNT_0: SystemFont 0 CLS_PRT_FNT_1: SystemFont 1 CLS_PRT_FNT_2: SystemFont 2 CLS_PRT_FNT_3: SystemFont 3 CLS_PRT_FNT_4: SystemFont 4 CLS_PRT_FNT_5: SystemFont 5 CLS_PRT_FNT_6: SystemFont 6 CLS_PRT_FNT_7: SystemFont 7 CLS_PRT_FNT_8: SystemFont 8 CLS_PRT_FNT_TRIUMVIRATE: Smooth font CLS_PRT_FNT_TRIUMVIRATE_B: Smooth font (bold) CLS_PRT_FNT_KANJI: Kanji (left to right) CLS_PRT_FNT_KANJIT: Kanji (top to bottom)
rotation	[IN]	Direction of rotation	CLS_RT_NORMAL: No rotation CLS_RT_RIGHT90: Rotate CW 90 CLS_RT_ROTATE180: Rotate CW 180 CLS_RT_LEFT90: Rotate CCW 90
hexp	[IN]	Horizontal magnification	1 – 24
vexp	[IN]	Vertical magnification	1 – 24
size	[IN]	Font size	CLS_PRT_FNT_SIZE_4: 4pt CLS_PRT_FNT_SIZE_5: 5pt CLS_PRT_FNT_SIZE_6: 6pt CLS_PRT_FNT_SIZE_8: 8pt CLS_PRT_FNT_SIZE_10: 10pt CLS_PRT_FNT_SIZE_12: 12pt CLS_PRT_FNT_SIZE_14: 14pt CLS_PRT_FNT_SIZE_18: 18pt CLS_PRT_FNT_SIZE_24: 24pt CLS_PRT_FNT_SIZE_30: 30pt CLS_PRT_FNT_SIZE_36: 36pt CLS_PRT_FNT_SIZE_48: 48pt CLS_PRT_FNT_KANJI_SIZE_16: Kanji 16dot CLS_PRT_FNT_KANJI_SIZE_24: Kanji 24dot CLS_PRT_FNT_KANJI_SIZE_32: Kanji 32dot * CLS_PRT_FNT_KANJI_SIZE_48: Kanji 48dot * * Not supported in locale for Chinese model.
x	[IN]	X-coordinate	0000 - 9999
y	[IN]	Y-coordinate	* The origin is at bottom-left.(0, 0)

encoding	[IN]	Character encoding	See " <a href="#">3.3.Predefined Constants.</a> "
----------	------	--------------------	---

**Description**

Draws characters by using a printer device font with specifying options such as rotation, magnification, size and coordinates.

Available font sizes depend on typeface.

Kanji: 16, 24, 32, 48 \*dots

Smooth font: 4, 5, 6, 8, 10, 12, 14, 18, 24, 30, 36, 48

Other: Fixed sizes depend on typeface.

If encoding is omitted, the encoding according to the locale will be automatically set. The model that can specify UTF-8 (CLS\_ENC\_CDPG\_UTF\_8) for encoding is only CL-S700III. (As of April 2025)

**Return value**

Returns CLS\_SUCCESS(0) on success, an error code otherwise. See "[2.1 Return value](#)" for the error codes.

**Example**

```
// Locale:OTHER
design.drawTextPtrFont("drawTextPtrFont",
    LabelConst.CLS_LOCALE_OTHER, LabelConst.CLS_PRT_FNT_TRIUMVIRATE,
    LabelConst.CLS_RT_NORMAL, 1, 1, LabelConst.CLS_PRT_FNT_SIZE_10,
    100, 100);

// Locale:JP
design.drawTextPtrFont("テキスト印刷(プリンタフォント)",
    LabelConst.CLS_LOCALE_JP, LabelConst.CLS_PRT_FNT_KANJI,
    LabelConst.CLS_RT_NORMAL, 1, 1,
    LabelConst.CLS_PRT_FNT_KANJI_SIZE_16, 100, 300);

// Locale:CN
design.drawTextPtrFont("测试打印", LabelConst.CLS_LOCALE_CN,
    LabelConst.CLS_PRT_FNT_KANJI, LabelConst.CLS_RT_NORMAL, 1, 1,
    LabelConst.CLS_PRT_FNT_KANJI_SIZE_16, 100, 300);

// Locale:KR
design.drawTextPtrFont("테스트 인쇄", LabelConst.CLS_LOCALE_KR,
    LabelConst.CLS_PRT_FNT_KANJI, LabelConst.CLS_RT_NORMAL, 1, 1,
    LabelConst.CLS_PRT_FNT_KANJI_SIZE_16, 100, 300);
```

### 2.3.3 drawTextDLFont method

#### Syntax

```
int drawTextDLFont (String data, int encoding,  
                    String fontID, int rotation, int hexp, int vexp, int point, int x, int y)
```

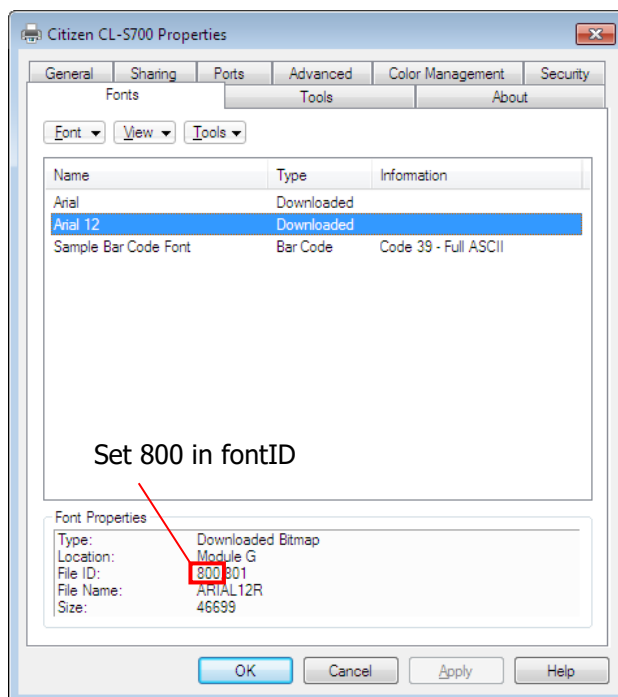
#### Parameters

Parameter	[IN/OUT]	Description	Setting range
data	[IN]	String data	* Unsupported characters will be replaced with blank text.
encoding	[IN]	Character encoding	See " <a href="#">3.3.Predefined Constants.</a> "
fontID	[IN]	Font ID stored in the printer	Numeric characters - Bitmap font Alphanumeric characters - TrueType font
rotation	[IN]	Direction of rotation	CLS_RT_NORMAL: No rotation CLS_RT_RIGHT90: Rotate CW 90 CLS_RT_ROTATE180: Rotate CW 180 CLS_RT_LEFT90: Rotate CCW 90
hexp	[IN]	Horizontal magnification	1 - 24
vexp	[IN]	Vertical magnification	1 - 24
point	[IN]	Font size	001 - 999
x	[IN]	X-coordinate	0000 - 9999
y	[IN]	Y-coordinate	* The origin is at bottom-left.(0, 0)

#### Description

Draws characters by using a downloaded font with specifying options such as rotation, magnification, size and coordinates.

In case of bitmap font, the driver shows two IDs. Use the smaller number of ID.



The model that can specify UTF-8 (CLS\_ENC\_CDPG\_UTF\_8) for encoding is only CL-S700III. (As of April 2025)



**Return value**

Returns CLS\_SUCCESS(0) on success, an error code otherwise. See "[2.1 Return value](#)" for the error codes.

**Example**

```
// TrueType font
design.drawTextDLFont("TrueType",
    LabelConst.CLS_ENC_CDPG_IBM850, "S50",
    LabelConst.CLS_RT_NORMAL, 1, 1, 12, 100, 100);

// Bitmap font
design.drawTextDLFont("Bitmap",
    LabelConst.CLS_ENC_CDPG_IBM850, "800",
    LabelConst.CLS_RT_NORMAL, 1, 1, 12, 100, 200);
```

## 2.3.4 drawTextLocalFont method

### Syntax

- 3) int drawTextLocalFont (String data, Typeface fontType, int rotation, int hRatio, int vRatio, int point, int style, int x, int y)
- 4) int drawTextLocalFont (String data, Typeface fontType, int rotation, int hRatio, int vRatio, int point, int style, int x, int y, int resolution)
- 5) int drawTextLocalFont (String data, Typeface fontType, int rotation, int hRatio, int vRatio, int point, int style, int x, int y, int resolution, int measurementUnit)

### Parameters

Parameter	[IN/OUT]	Description	Setting range
data	[IN]	String data	
fontType	[IN]	Font type	Font type
rotation	[IN]	Direction of rotation	CLS_RT_NORMAL: No rotation CLS_RT_RIGHT90: Rotate CW 90 CLS_RT_ROTATE180: Rotate CW 180 CLS_RT_LEFT90: Rotate CCW 90
hRatio	[IN]	Horizontal ratio [%]	1 - 1000
vRatio	[IN]	Vertical ratio [%]	1 - 1000
point	[IN]	Font size	
style	[IN]	Font style	CLS_FNT_DEFAULT : None CLS_FNT_BOLD : Bold CLS_FNT_REVERSE : Reverse CLS_FNT_UNDERLINE : Underline CLS_FNT_ITALIC : Italic CLS_FNT_STRIKEOUT : Strikethrough * Use " " to specify multiple options.
x	[IN]	X-coordinate	0000 – 9999 * The origin is at bottom-left.(0, 0)
y	[IN]	Y-coordinate	
resolution	[IN]	Resolution [dpi]	CLS_PRT_RES_203(203dpi) CLS_PRT_RES_300(300dpi) * 203 dpi by default.
measurementUnit	[IN]	Printer Metric / Inch setting	CLS_UNIT_MILLI CLS_UNIT_INCH * CLS_UNIT_INCH by default.

### Description

Draws characters by using a font installed in the computer, with specifying options such as rotation, magnification, size and coordinates. What this method does internally is to generate a graphic image based on the given parameters, to store the graphic image into the printer and to print the stored graphic image. This doesn't check the file availability or validity. To print with the specified font size, specify the resolution as same as the actual printer. When specifying image rotation, it is necessary to specify the resolution and unit selection settings according to the printer for reference position correction.

### Return value

Returns CLS\_SUCCESS(0) on success, an error code otherwise. See ["2.1 Return value"](#) for the error codes.

**Example**

```
// 203 dpi(default), inch(default)
design.drawTextLocalFont("drawTextLocalFont", Typeface.SERIF,
    LabelConst.CLS_RT_NORMAL, 100, 100, 10,
    (LabelConst.CLS_FNT_BOLD | LabelConst.CLS_FNT_ITALIC), 10, 0);

// 300 dpi, inch(default)
design.drawTextLocalFont("drawTextLocalFont", Typeface.SERIF,
    LabelConst.CLS_RT_NORMAL, 100, 100, 10,
    LabelConst.CLS_FNT_DEFAULT, 10, 50,
    LabelConst.CLS_PRT_RES_300);

// 300 dpi, mm
design.drawTextLocalFont("drawTextLocalFont", Typeface.SERIF,
    LabelConst.CLS_RT_NORMAL, 100, 100, 10,
    LabelConst.CLS_FNT_DEFAULT, 10, 50,
    LabelConst.CLS_PRT_RES_300, LabelConst.CLS_UNIT_MILLI);
```

## 2.3.5 drawNVBitmap method

### Syntax

int drawNVBitmap (String name, int hexp, int vexp, int x, int y)

### Parameters

Parameter	[IN/OUT]	Description	Setting range
name	[IN]	Graphic file name	ASCII characters except below: -Underscore cannot be the first character. -The following symbols. ¥ / : * ? " < >
hexp	[IN]	Horizontal magnification	1 - 24
vexp	[IN]	Vertical magnification	1 - 24
x	[IN]	X-coordinate	0000 - 9999
y	[IN]	Y-coordinate	* The origin is at bottom-left.(0, 0)

### Description

Draws a graphic by using a graphic image stored in the printer, with specifying options such as magnification and coordinates. This doesn't check the file availability or validity.

### Return value

Returns CLS\_SUCCESS(0) on success, an error code otherwise. See "[2.1 Return value](#)" for the error codes.

### Example

See [storeNVBitmap method](#).

## 2.3.6 drawBitmap method

### Syntax

- 1) int drawBitmap (String filePath, int rotation, int width, int height, int x, int y)
- 2) int drawBitmap (String filePath, int rotation, int width, int height, int x, int y, int resolution, int measurementUnit)

### Parameters

Parameter	[IN/OUT]	Description	Setting range
filePath	[IN]	Path to a local graphic file	
rotation	[IN]	Direction of rotation	CLS_RT_NORMAL: No rotation CLS_RT_RIGHT90: Rotate CW 90 CLS_RT_ROTATE180: Rotate CW 180 CLS_RT_LEFT90: Rotate CCW 90
width	[IN]	Drawing width [Pixel]	
height	[IN]	Drawing height [Pixel]	
x	[IN]	X-coordinate	0000-9999
y	[IN]	Y-coordinate	* The origin is at bottom-left.(0, 0)
resolution	[IN]	Resolution [dpi]	CLS_PRT_RES_203(203dpi) CLS_PRT_RES_300(300dpi) * 203 dpi by default.
measurementUnit	[IN]	Printer Metric / Inch setting	CLS_UNIT_MILLI CLS_UNIT_INCH * CLS_UNIT_INCH by default.

### Description

Draws a graphic image stored in the local computer, with specifying options such as rotation, size and coordinates.

What this method does internally is to store a graphic image into the printer and to print the stored graphic image. This doesn't check the availability or validity of the stored file. When specifying image rotation, it is necessary to specify the resolution and unit selection settings according to the printer for reference position correction.

Supported graphic file formats are BMP/GIF/EXIF/JPG/PNG/TIFF.

The graphic will be resized based on the width/height parameters with keeping the aspect ratio and taking the maximum available size to fit.

Example: The sizes will be "200x50" in the case below.

Original sizes: 400x100 pixels

Width parameter: 200

Height parameter: 200

When 0 is set to either width or height, the sizes will be calculated based on another (non-zero) parameter. When both parameters are zero, the original sizes will be used.

Example : The sizes will be "800x200" in the case below.

Original sizes: 400x100 pixels

Width parameter: 0

Height parameter: 200

### Return value

Returns CLS\_SUCCESS(0) on success, an error code otherwise. See "[2.1 Return value](#)" for the error codes.

**Example**

```
// 203 dpi(default), inch(default)
design.drawBitmap("/sdcard/temp/citizen/Images/Sample.bmp",
    LabelConst.CLS_RT_NORMAL, 0, 0, 10, 10);

// 300 dpi, mm
design.drawBitmap("/sdcard/temp/citizen/Images/Sample.bmp",
    LabelConst.CLS_RT_NORMAL, 0, 0, 10, 10
    LabelConst.CLS_PRT_RES_300, LabelConst.CLS_UNIT_MILLI);
```

## 2.3.7 drawBarCode method

### Syntax

```
int drawBarCode (String data, int symbology,
                 int rotation, int thick, int narrow, int height, int x, int y, int showText)
```

### Parameters

Parameter	[IN/OUT]	Description	Setting range
data	[IN]	String data	ASCII code character string.
symbology	[IN]	Barcode type	CLS_BCS_CODE39 : Code 3 of 9 CLS_BCS_UPCA : UPC-A CLS_BCS_UPCE : UPC-E CLS_BCS_INTERLEAVED25 : Interleaved 2 of 5 CLS_BCS_CODE128 : Code 128 CLS_BCS_EAN13 : EAN-13(JAN-13) CLS_BCS_EAN8 : EAN-8(JAN-8) CLS_BCS_HIBC : HIBC CLS_BCS_CODABAR : CODABAR(NW-7) CLS_BCS_INT25 : Int 2 of 5 CLS_BCS_PLESSEY : Plessey CLS_BCS_CASECODE : CASE CODE CLS_BCS_UPC2DIG : UPC 2DIG ADD CLS_BCS_UPC5DIG : UPC 5DIG ADD CLS_BCS_CODE93 : Code93 CLS_BCS_ITF14 : ITF-14 CLS_BCS_ZIP : ZIP CLS_BCS_ITF16 : IFT-16 CLS_BCS_UCCEAN128 : UCC/EAN-128 CLS_BCS_INDUSTRIAL25 : Industrial 2 of 5 CLS_BCS_UCCEAN128KMART : UCC/EAN-128(for K-MART) CLS_BCS_COOP25 : COOP 2 of 5 CLS_BCS_UCCEAN128RANDOMWEIGHT : UCC/EAN-128 Random Weight CLS_BCS_TELEPEN : Telepen
rotation	[IN]	Direction of rotation	CLS_RT_NORMAL: No rotation CLS_RT_RIGHT90: Rotate CW 90 CLS_RT_ROTATE180: Rotate CW 180 CLS_RT_LEFT90: Rotate CCW 90
thick	[IN]	Thick bar width	1-24
narrow	[IN]	Narrow bar width	1-24
height	[IN]	Height of barcode	001-999
x	[IN]	X-coordinate	0000-9999 * The origin is at bottom-left.(0, 0)
y	[IN]	Y-coordinate	
showText	[IN]	Human readable text	CLS_BCS_TEXT_HIDE : Hide CLS_BCS_TEXT_SHOW : Show

**Description**

Draws a 1D (one-dimensional) barcode with specifying options such as barcode type, rotation, thick/narrow bar width, coordinates and human readable text.

\* Supported thick/narrow bar ratio or data type depends on symbology. Refer to the command reference for details.

**Return value**

Returns CLS\_SUCCESS(0) on success, an error code otherwise. See "[2.1 Return value](#)" for the error codes.

**Example**

```
// Code3of9
String code39 = "ABC123456789";
design.drawBarCode(code39, LabelConst.CLS_BCS_CODE39,
    LabelConst.CLS_RT_NORMAL, 6, 2, 50, 10, 10,
    LabelConst.CLS_BCS_TEXT_SHOW);

// UPC-A
String upca = "01234567890";
design.drawBarCode(upca, LabelConst.CLS_BCS_UPCA,
    LabelConst.CLS_RT_NORMAL, 2, 2, 50, 10, 100,
    LabelConst.CLS_BCS_TEXT_SHOW);

// UPC-E
String upce = "123456";
design.drawBarCode(upce, LabelConst.CLS_BCS_UPCE,
    LabelConst.CLS_RT_NORMAL, 2, 2, 50, 10, 100,
    LabelConst.CLS_BCS_TEXT_SHOW);

// Interleaved2of5
String interleaved25 = "1234567890";
design.drawBarCode(interleaved25, LabelConst.CLS_BCS_INTERLEAVED25,
    LabelConst.CLS_RT_NORMAL, 5, 2, 50, 10, 100,
    LabelConst.CLS_BCS_TEXT_SHOW);

// Code128
String code128 = "1234567890";
design.drawBarCode(code128, LabelConst.CLS_BCS_CODE128,
    LabelConst.CLS_RT_NORMAL, 2, 4, 50, 10, 10,
    LabelConst.CLS_BCS_TEXT_SHOW);

// EAN-13
String ean13 = "123456789012";
design.drawBarCode(ean13, LabelConst.CLS_BCS_EAN13,
    LabelConst.CLS_RT_NORMAL, 3, 3, 50, 10, 100,
    LabelConst.CLS_BCS_TEXT_SHOW);

// EAN-8
String ean8 = "1234567";
design.drawBarCode(ean8, LabelConst.CLS_BCS_EAN8,
    LabelConst.CLS_RT_NORMAL, 4, 4, 50, 10, 10,
    LabelConst.CLS_BCS_TEXT_SHOW);
```



```
// HIBC
String hibc = "1234567890";
design.drawBarCode(hibc, LabelConst.CLS_BCS_HIBC,
    LabelConst.CLS_RT_NORMAL, 6, 2, 50, 10, 100,
    LabelConst.CLS_BCS_TEXT_SHOW);

// CODABAR
String codabar = "a1234567890b";
design.drawBarCode(codabar, LabelConst.CLS_BCS_CODABAR,
    LabelConst.CLS_RT_NORMAL, 6, 2, 40, 10, 10,
    LabelConst.CLS_BCS_TEXT_SHOW);

// Interleaved2of5 W/BARS
String int25 = "1234567890";
design.drawBarCode(int25, LabelConst.CLS_BCS_INT25,
    LabelConst.CLS_RT_NORMAL, 5, 2, 50, 10, 100,
    LabelConst.CLS_BCS_TEXT_SHOW);

// PLESSEY
String plessey = "1234567890";
design.drawBarCode(plessey, LabelConst.CLS_BCS_PLESSEY,
    LabelConst.CLS_RT_NORMAL, 4, 2, 50, 10, 10,
    LabelConst.CLS_BCS_TEXT_SHOW);

// CASE CODE
String cascode = "1234567890123";
design.drawBarCode(cascode, LabelConst.CLS_BCS_CASECODE,
    LabelConst.CLS_RT_NORMAL, 5, 2, 50, 10, 100,
    LabelConst.CLS_BCS_TEXT_SHOW);

// UPC 2DIG ADD
String upca = "01234567890";
String upc2dig = "12";

design.drawBarCode(upca, LabelConst.CLS_BCS_UPCA,
    LabelConst.CLS_RT_NORMAL, 2, 2, 50, 10, 100,
    LabelConst.CLS_BCS_TEXT_SHOW);

design.drawBarCode(upc2dig, LabelConst.CLS_BCS_UPC2DIG,
    LabelConst.CLS_RT_NORMAL, 4, 2, 50, 130, 100,
    LabelConst.CLS_BCS_TEXT_SHOW);

// UPC 5DIG ADD
String upca = "01234567890";
String upc5dig = "12345";

design.drawBarCode(upca, LabelConst.CLS_BCS_UPCA,
    LabelConst.CLS_RT_NORMAL, 2, 2, 50, 10, 100,
    LabelConst.CLS_BCS_TEXT_SHOW);

design.drawBarCode(upc5dig, LabelConst.CLS_BCS_UPC5DIG,
    LabelConst.CLS_RT_NORMAL, 4, 2, 50, 130, 100,
    LabelConst.CLS_BCS_TEXT_SHOW);
```

```
// Code93
String code93 = "1234ABCD";
design.drawBarCode(code93, LabelConst.CLS_BCS_CODE93,
    LabelConst.CLS_RT_NORMAL, 6, 6, 50, 10, 10,
    LabelConst.CLS_BCS_TEXT_SHOW);

// ITF-14
String itf14 = "1234567890123";
design.drawBarCode(itf14, LabelConst.CLS_BCS_ITF14,
    LabelConst.CLS_RT_NORMAL, 5, 2, 50, 10, 100,
    LabelConst.CLS_BCS_TEXT_SHOW);

// ZIP
String zip = "123456789";
design.drawBarCode(zip, LabelConst.CLS_BCS_ZIP,
    LabelConst.CLS_RT_NORMAL, 1, 1, 10, 10, 10,
    LabelConst.CLS_BCS_TEXT_SHOW);

// ITF-16
String itf16 = "123456789012345";
design.drawBarCode(itf16, LabelConst.CLS_BCS_ITF16,
    LabelConst.CLS_RT_NORMAL, 5, 2, 50, 10, 100,
    LabelConst.CLS_BCS_TEXT_SHOW);

// UCC/EAN-128
String uccean128 = "1234567890123456789";
design.drawBarCode(uccean128, LabelConst.CLS_BCS_UCCEAN128,
    LabelConst.CLS_RT_NORMAL, 4, 4, 50, 10, 10,
    LabelConst.CLS_BCS_TEXT_SHOW);

// Industrial2of5
String industrial25 = "1234567890";
design.drawBarCode(industrial25, LabelConst.CLS_BCS_INDUSTRIAL25,
    LabelConst.CLS_RT_NORMAL, 5, 2, 50, 10, 100,
    LabelConst.CLS_BCS_TEXT_SHOW);

// UCC/EAN-128(for K-MART)
String uccean128kmart = "123456789012345678";
design.drawBarCode(uccean128kmart, LabelConst.CLS_BCS_UCCEAN128KMART,
    LabelConst.CLS_RT_NORMAL, 3, 3, 50, 10, 10,
    LabelConst.CLS_BCS_TEXT_SHOW);

// COOP 2 of 5
String coop25 = "0123456789";
design.drawBarCode(coop25, LabelConst.CLS_BCS_COOP25,
    LabelConst.CLS_RT_NORMAL, 10, 4, 50, 10, 100,
    LabelConst.CLS_BCS_TEXT_SHOW);
```

```
// UCC/EAN-128 Random Weight
String uccean128randomweight = "1234567890123456789012345678909999";
design.drawBarcode(uccean128randomweight,
    LabelConst.CLS_BCS_UCCEAN128RANDOMWEIGHT,
    LabelConst.CLS_RT_NORMAL, 2, 2, 50, 10, 10,
    LabelConst.CLS_BCS_TEXT_SHOW);

// Telepen
String telepen = "1234567890";
design.drawBarcode(telepen, LabelConst.CLS_BCS_TELEPEN,
    LabelConst.CLS_RT_NORMAL, 2, 2, 50, 10, 100,
    LabelConst.CLS_BCS_TEXT_SHOW);
```

## 2.3.8 drawMaxiCode method

### Syntax

```
int drawMaxiCode (String[] data, int rotation, int x, int y)
```

### Parameters

Parameter	[IN/OUT]	Description	Setting range
data	[IN]	String data	ASCII code character string array
rotation	[IN]	Direction of rotation	CLS_RT_NORMAL: No rotation CLS_RT_RIGHT90: Rotate CW 90 CLS_RT_ROTATE180: Rotate CW 180 CLS_RT_LEFT90: Rotate CCW 90
x	[IN]	X-coordinate	0000-9999 * The origin is at bottom-left.(0, 0)
y	[IN]	Y-coordinate	

### Description

Draws the UPS MaxiCode barcode with specifying options such as rotation and coordinates.

The following information are required in the data parameter in order as below:

- 1) 5-digit - Zip code
- 2) 4-digit ++4 Zip code
- 3) 3-digit – Country Code
- 4) 3-digit - Class of Service
- 5) Up to 84 digits other information

### Return value

Returns CLS\_SUCCESS(0) on success, an error code otherwise. See ["2.1 Return value"](#) for the error codes.

### Example

```
String[] data = new String[] {"90501", "6282", "840", "001", "1Z00004951"};
design.drawMaxiCode(data, LabelConst.CLS_RT_NORMAL, 100, 0);
```

## 2.3.9 drawPDF417 method

### Syntax

int drawPDF417 (String data, int encoding, int rotation, int exp, int ECLevel, int x, int y)

### Parameters

Parameter	[IN/OUT]	Description	Setting range
data	[IN]	String data	* The characters available with the specified encoding.
encoding	[IN]	Character encoding	See " <a href="#">3.3.Predefined Constants</a> ."
rotation	[IN]	Direction of rotation	CLS_RT_NORMAL: No rotation CLS_RT_RIGHT90: Rotate CW 90 CLS_RT_ROTATE180: Rotate CW 180 CLS_RT_LEFT90: Rotate CCW 90
exp	[IN]	Magnification	1-5
ECLevel	[IN]	Error correction level	CLS_PDF417_EC_LEVEL_0 : Level 0 CLS_PDF417_EC_LEVEL_1 : Level 1 CLS_PDF417_EC_LEVEL_2 : Level 2 CLS_PDF417_EC_LEVEL_3 : Level 3 CLS_PDF417_EC_LEVEL_4 : Level 4 CLS_PDF417_EC_LEVEL_5 : Level 5 CLS_PDF417_EC_LEVEL_6 : Level 6 CLS_PDF417_EC_LEVEL_7 : Level 7 CLS_PDF417_EC_LEVEL_8 : Level 8
x	[IN]	X-coordinate	0000-9999 * The origin is at bottom-left.(0, 0)
y	[IN]	Y-coordinate	

### Description

Draws the PDF417 barcode with specifying options such as character encoding, rotation, magnification, EC level and coordinates.

### Return value

Returns CLS\_SUCCESS(0) on success, an error code otherwise. See "[2.1 Return value](#)" for the error codes.

### Example

```
design.drawPDF417("0123456789",
    LabelConst.CLS_ENC_CDPG_US_ASCII,
    LabelConst.CLS_RT_NORMAL, 3,
    LabelConst.CLS_PDF417_EC_LEVEL_0, 0, 0);
```

## 2.3.10 drawDataMatrix method

### Syntax

int drawDataMatrix (String data, int encoding, int rotation, int exp, int ECLevel, int x, int y)

### Parameters

Parameter	[IN/OUT]	Description	Setting range
data	[IN]	String data	-Numeric: Up to 3,116 characters -Alphanumeric: Up to 2,335 characters * The characters available with the specified encoding.
encoding	[IN]	Character encoding	See #9 in " <a href="#">3.3.Predefined Constants</a> ."
rotation	[IN]	Direction of rotation	CLS_RT_NORMAL: No rotation CLS_RT_RIGHT90: Rotate CW 90 CLS_RT_ROTATE180: Rotate CW 180 CLS_RT_LEFT90: Rotate CCW 90
exp	[IN]	Magnification	1-15
ECLevel	[IN]	Error correction level	CLS_DATAMATRIX_EC_LEVEL_200 : 200
x	[IN]	X-coordinate	0000-9999 * The origin is at bottom-left.(0, 0)
y	[IN]	Y-coordinate	

### Description

Draws the DataMatrix barcode with specifying options such as character encoding, rotation, magnification, EC level and coordinates.

### Return value

Returns CLS\_SUCCESS(0) on success, an error code otherwise. See "[2.1 Return value](#)" for the error codes.

### Example

```
design.drawDataMatrix("0123456789",
    LabelConst.CLS_ENC_CDPG_US_ASCII,
    LabelConst.CLS_RT_NORMAL, 15,
    LabelConst.CLS_DATAMATRIX_EC_LEVEL_200, 0, 0);
```

### 2.3.11 drawQRCode method

#### Syntax

int drawQRCode (String data, int encoding, int rotation, int exp, int ECLevel, int x, int y)

int drawQRCode (byte[] bytes, int rotation, int exp, int ECLevel, int x, int y)

#### Parameters

Parameter	[IN/OUT]	Description	Setting range
data	[IN]	String data	* The characters available with the specified encoding.
encoding	[IN]	Character encoding	See #9 in " <a href="#">3.3.Predefined Constants</a> ."
bytes	[IN]	Binary data	Byte array data
rotation	[IN]	Direction of rotation	CLS_RT_NORMAL: No rotation CLS_RT_RIGHT90: Rotate CW 90 CLS_RT_ROTATE180: Rotate CW 180 CLS_RT_LEFT90: Rotate CCW 90
exp	[IN]	Magnification	1-15
ECLevel	[IN]	Error correction level	CLS_QRCODE_EC_LEVEL_L: Level L (7%) CLS_QRCODE_EC_LEVEL_M: Level M (15%) CLS_QRCODE_EC_LEVEL_Q: Level Q (25%) CLS_QRCODE_EC_LEVEL_H: Level H (30%)
x	[IN]	X-coordinate	0000-9999 * The origin is at bottom-left.(0, 0)
y	[IN]	Y-coordinate	

#### Description

Draws the QR code with specifying options such as character encoding, rotation, magnification, EC level and coordinates.

#### Return value

Returns CLS\_SUCCESS(0) on success, an error code otherwise. See "[2.1 Return value](#)" for the error codes.

#### Example

```
// String data
design.drawQRCode("アイエオ12345",
    LabelConst.CLS_ENC_CDPG_SHIFT_JIS,
    LabelConst.CLS_RT_NORMAL, 10,
    LabelConst.CLS_QRCODE_EC_LEVEL_H, 100, 100);

// Binary data
byte[] bindata = new byte[] {0x01, 0x02, 0x03, 0x04, 0x05};
design.drawQRCode(bindata,
    LabelConst.CLS_RT_NORMAL, 10,
    LabelConst.CLS_QRCODE_EC_LEVEL_H, 150, 150);
```

### 2.3.12 drawAztec method

#### Syntax

int drawAztec (String data, int encoding, int rotation, int exp, int ECLevel, int x, int y)

#### Parameters

Parameter	[IN/OUT]	Description	Setting range
data	[IN]	String data	* The characters available with the specified encoding.
encoding	[IN]	Character encoding	See #9 in " <a href="#">3.3.Predefined Constants</a> ."
rotation	[IN]	Direction of rotation	CLS_RT_NORMAL: No rotation CLS_RT_RIGHT90: Rotate CW 90 CLS_RT_ROTATE180: Rotate CW 180 CLS_RT_LEFT90: Rotate CCW 90
exp	[IN]	Magnification	1-15
ECLevel	[IN]	Error correction level	CLS_AXTEC_EC_LEVEL_000: Level 0 (Error correction ratio 23%)
x	[IN]	X-coordinate	0000-9999
y	[IN]	Y-coordinate	* The origin is at bottom-left.(0, 0)

#### Description

Draws the Aztec barcode with specifying options such as character encoding, rotation, magnification, EC level and coordinates.

#### Return value

Returns CLS\_SUCCESS(0) on success, an error code otherwise. See "[2.1 Return value](#)" for the error codes.

#### Example

```
design.drawAztec("0123456789",
    LabelConst.CLS_ENC_CDPG_US_ASCII,
    LabelConst.CLS_RT_NORMAL, 10,
    LabelConst.CLS_AXTEC_EC_LEVEL_000, 0, 0);
```



### 2.3.13 drawGS1DataBar method

#### Syntax

int drawGS1DataBar (String[] data, int type, int rotation, int exp, int x, int y)

#### Parameters

Parameter	[IN/OUT]	Description	Setting range
data	[IN]	String data	ASCII code character string.
symbology	[IN]	Barcode type	CLS_GS1_DATABAR_OMNI_DIRECTIONAL: Omni-directional CLS_GS1_DATABAR_COMPOSITE: Composite CLS_GS1_DATABAR_TRUNCATION: Truncation CLS_GS1_DATABAR_STACKED: Stacked CLS_GS1_DATABAR_STACKED_OMNI_DIRECTIONAL: Stacked Omni-directional CLS_GS1_DATABAR_LIMITED: Limited CLS_GS1_DATABAR_EXPANDED: Expanded
rotation	[IN]	Direction of rotation	CLS_RT_NORMAL: No rotation CLS_RT_RIGHT90: Rotate CW 90 CLS_RT_ROTATE180: Rotate CW 180 CLS_RT_LEFT90: Rotate CCW 90
exp	[IN]	Magnification	1-15
x	[IN]	X-coordinate	0000-9999 * The origin is at bottom-left.(0, 0)
y	[IN]	Y-coordinate	

#### Description

Draws the GS1DataBar barcode with specifying options such as rotation, magnification and coordinates.

\* The available text types vary depending on the symbology. Refer to the "GS1 DataBar (RSS) in the command reference.

#### Return value

Returns CLS\_SUCCESS(0) on success, an error code otherwise. See "[2.1 Return value](#)" for the error codes.

#### Example

```
// GS1 DataBar Omni-Directional
String[] omnidirectional = new String[] { "1234567890123" };
design.drawGS1DataBar(omnidirectional,
    LabelConst.CLS_GS1_DATABAR_OMNI_DIRECTIONAL,
    LabelConst.CLS_RT_NORMAL, 3, 10, 10);

// GS1 DataBar Composite
String[] composit = new String[] {
    "1234567890123", "1234567890-07/07/07" };
design.drawGS1DataBar(composit,
    LabelConst.CLS_GS1_DATABAR_COMPOSITE,
    LabelConst.CLS_RT_NORMAL, 3, 10, 110);
```

```
// GS1 DataBar Truncation
String[] truncation = new String[] { "1234567890123" };
design.drawGS1DataBar(truncation,
    LabelConst.CLS_GS1_DATABAR_TRUNCATION,
    LabelConst.CLS_RT_NORMAL, 3, 10, 210);

// GS1 DataBar Stacked
String[] stacked = new String[] { "1234567890123" };
design.drawGS1DataBar(stacked,
    LabelConst.CLS_GS1_DATABAR_STACKED,
    LabelConst.CLS_RT_NORMAL, 3, 10, 310);

// GS1 DataBar Stacked-Omni-Directional
String[] stackedOD = new String[] { "1234567890123" };
design.drawGS1DataBar(stackedOD,
    LabelConst.CLS_GS1_DATABAR_STACKED_OMNI_DIRECTIONAL,
    LabelConst.CLS_RT_NORMAL, 3, 10, 410);

// GS1 DataBar Limited
String[] limited = new String[] { "1234567890123" };
design.drawGS1DataBar(limited,
    LabelConst.CLS_GS1_DATABAR_LIMITED,
    LabelConst.CLS_RT_NORMAL, 3, 10, 610);

// GS1 DataBar Expanded
String[] expanded = new String[] { "041234567890123" };
design.drawGS1DataBar(expanded,
    LabelConst.CLS_GS1_DATABAR_EXPANDED,
    LabelConst.CLS_RT_NORMAL, 3, 10, 710);
```

## 2.3.14 drawLine method

### Syntax

```
int drawLine (int x1, int y1, int x2, int y2, int thickness)
```

### Parameters

Parameter	[IN/OUT]	Description	Setting range
x1	[IN]	Start position (X-coordinate)	0000-9999 * The origin is at bottom-left.(0, 0)
y1	[IN]	Start position (Y-coordinate)	
x2	[IN]	End position (X-coordinate)	
y2	[IN]	End position (Y-coordinate)	
thickness	[IN]	Line width (Reference point is center)	0000-9999

### Description

Draws a line of the specified width.

Start and stop positions will be located at the center of the line. Each coordinate must be within the range of 0000-9999. Moreover the whole line image (rectangle) including the line thickness must be within the range of 0000-9999 as well. Returns CLS\_E\_ILLEGAL(1101) otherwise.

### Return value

Returns CLS\_SUCCESS(0) on success, an error code otherwise. See "[2.1 Return value](#)" for the error codes.

### Example

```
// FormatAttribute:OR
printer.setFormatAttribute(1);

// Bar
design.drawLine(20, 30, 20, 300, 10);

// Horizontal line
design.drawLine(16, 34, 200, 34, 10);
```

## 2.3.15 drawRect method

### Syntax

int drawRect (int x, int y, int width, int height, int thickness)

### Parameters

Parameter	[IN/OUT]	Description	Setting range
x	[IN]	Start position (X-coordinate)	0000 - 9999 * The origin is at bottom-left.(0, 0)
y	[IN]	Start position (Y-coordinate)	
width	[IN]	Box width	0000 - 9999
height	[IN]	Box height	
thickness	[IN]	Line width	0000 - 9999

### Description

Draws a box of the specified width, height and line width. Thicker line width expands the line inward and doesn't affect the outline size.

### Return value

Returns CLS\_SUCCESS(0) on success, an error code otherwise. See "[2.1 Return value](#)" for the error codes.

### Example

```
design.drawRect(20, 30, 180, 280, 10);
```

### 2.3.16 fillRect method

#### Syntax

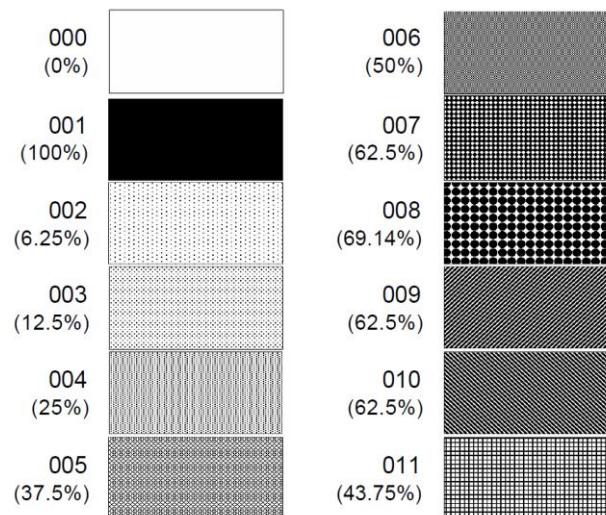
int fillRect (int x, int y, int width, int height, int pattern)

#### Parameters

Parameter	[IN/OUT]	Description	Setting range
x	[IN]	Start position (X-coordinate)	0000 - 9999 * The origin is at bottom-left.(0, 0)
y	[IN]	Start position (Y-coordinate)	
width	[IN]	Box width	0000 - 9999
height	[IN]	Box height	
pattern	[IN]	Shaded pattern	CLS_SHADED_PTN_0 : Shaded pattern 000 CLS_SHADED_PTN_1 : Shaded pattern 001 CLS_SHADED_PTN_2 : Shaded pattern 002 CLS_SHADED_PTN_3 : Shaded pattern 003 CLS_SHADED_PTN_4 : Shaded pattern 004 CLS_SHADED_PTN_5 : Shaded pattern 005 CLS_SHADED_PTN_6 : Shaded pattern 006 CLS_SHADED_PTN_7 : Shaded pattern 007 CLS_SHADED_PTN_8 : Shaded pattern 008 CLS_SHADED_PTN_9 : Shaded pattern 009 CLS_SHADED_PTN_10 : Shaded pattern 010 CLS_SHADED_PTN_11 : Shaded pattern 011

#### Description

Draws a box of the specified width, height and shaded pattern.



#### Return value

Returns CLS\_SUCCESS(0) on success, an error code otherwise. See "[2.1 Return value](#)" for the error codes.

#### Example

```
design.fillRect(20, 30, 180, 280, LabelConst.CLS_SHADED_PTN_10);
```

### 2.3.17 drawCircle method

**Syntax**

```
int drawCircle (int x, int y, int radius)
```

**Parameters**

Parameter	[IN/OUT]	Description	Setting range
x	[IN]	Start position (X-coordinate, center)	0000 - 9999 * The origin is at bottom-left.(0, 0)
y	[IN]	Start position (Y-coordinate, center)	
radius	[IN]	Radius	0000 - 0398

**Description**

Draws a circle of the specified radius.

**Return value**

Returns CLS\_SUCCESS(0) on success, an error code otherwise. See "[2.1 Return value](#)" for the error codes.

**Example**

```
design.drawCircle(50, 50, 15);
```

### 2.3.18 fillCircle method

#### Syntax

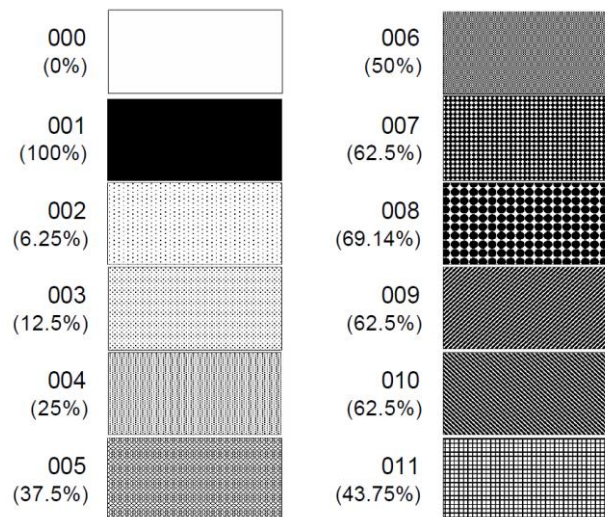
```
int fillCircle (int x, int y, int radius, int pattern)
```

#### Parameters

Parameter	[IN/OUT]	Description	Setting range
x	[IN]	Start position (X-coordinate, center)	0000 - 9999 * The origin is at bottom-left.(0, 0)
y	[IN]	Start position (Y-coordinate, center)	
radius	[IN]	Radius	0000 - 0398
pattern	[IN]	Shaded pattern	CLS_SHADED_PTN_0 : Shaded pattern 000 CLS_SHADED_PTN_1 : Shaded pattern 001 CLS_SHADED_PTN_2 : Shaded pattern 002 CLS_SHADED_PTN_3 : Shaded pattern 003 CLS_SHADED_PTN_4 : Shaded pattern 004 CLS_SHADED_PTN_5 : Shaded pattern 005 CLS_SHADED_PTN_6 : Shaded pattern 006 CLS_SHADED_PTN_7 : Shaded pattern 007 CLS_SHADED_PTN_8 : Shaded pattern 008 CLS_SHADED_PTN_9 : Shaded pattern 009 CLS_SHADED_PTN_10 : Shaded pattern 010 CLS_SHADED_PTN_11 : Shaded pattern 011

#### Description

Draws a circle of the specified radius and shaded pattern.



#### Return value

Returns CLS\_SUCCESS(0) on success, an error code otherwise. See "[2.1 Return value](#)" for the error codes.

#### Example

```
design.fillCircle(100, 100, 40, LabelConst.CLS_SHADED_PTN_2);
```

### 2.3.19 drawPolygon method

#### Syntax

```
int drawPolygon (int[] x, int[] y)
```

#### Parameters

Parameter	[IN/OUT]	Description	Setting range
x	[IN]	X points	0000 - 9999 * The origin is at bottom-left.(0, 0)
y	[IN]	Y points	

#### Description

Draws a closed polygon defined by arrays of x and y coordinates. Each pair of (x, y) coordinates defines a point. The number of elements must match. Returns CLS\_E\_ILLEGAL(1101) otherwise.

#### Return value

Returns CLS\_SUCCESS(0) on success, an error code otherwise. See "[2.1 Return value](#)" for the error codes.

#### Example

```
int[] x = new int[] {100, 200, 250, 150};
int[] y = new int[] {100, 100, 200, 200};

design.drawPolygon(x, y);           // Draw a parallelogram
```



## 2.3.20 fillPolygon method

### Syntax

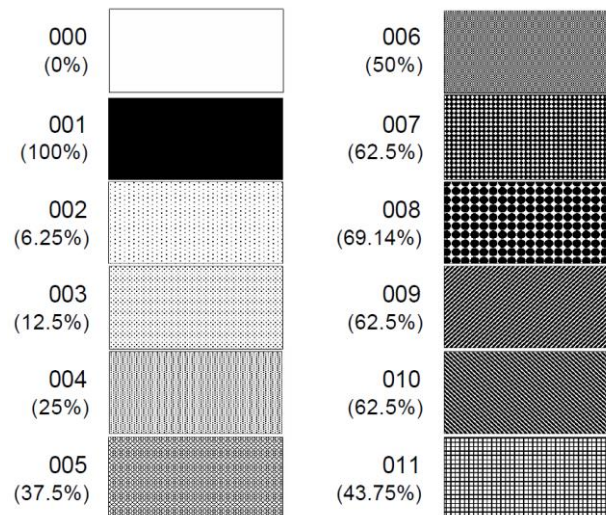
```
int fillPolygon (int[] x, int[] y, int pattern)
```

### Parameters

Parameter	[IN/OUT]	Description	Setting range
x	[IN]	X points	0000 - 9999 * The origin is at bottom-left.(0, 0)
y	[IN]	Y points	
pattern	[IN]	Shaded pattern	CLS_SHADED_PTN_0 : Shaded pattern 000 CLS_SHADED_PTN_1 : Shaded pattern 001 CLS_SHADED_PTN_2 : Shaded pattern 002 CLS_SHADED_PTN_3 : Shaded pattern 003 CLS_SHADED_PTN_4 : Shaded pattern 004 CLS_SHADED_PTN_5 : Shaded pattern 005 CLS_SHADED_PTN_6 : Shaded pattern 006 CLS_SHADED_PTN_7 : Shaded pattern 007 CLS_SHADED_PTN_8 : Shaded pattern 008 CLS_SHADED_PTN_9 : Shaded pattern 009 CLS_SHADED_PTN_10 : Shaded pattern 010 CLS_SHADED_PTN_11 : Shaded pattern 011

### Description

Draws a closed polygon defined by arrays of x and y coordinates, with the specified shaded pattern. Each pair of (x, y) coordinates defines a point. The number of elements must match. Returns CLS\_E\_ILLEGAL(1101) otherwise.



### Return value

Returns CLS\_SUCCESS(0) on success, an error code otherwise. See "[2.1 Return value](#)" for the error codes.

### Example

```
int[] x = new int[] {100, 200, 250, 150};
int[] y = new int[] {100, 100, 200, 200};

design.fillPolygon(x, y, LabelConst.CLS_SHADED_PTN_3);
// Draw a parallelogram
```

### 2.3.21 embedRawDesignCommand method

#### Syntax

```
int embedRawDesignCommand (byte[] data)
```

#### Parameters

Parameter	[IN/OUT]	Description	Setting range
data	[IN]	Design command	Data of binary.

#### Description

Embeds raw printer commands to the label design. This allows adding raw label format commands individually.

The SendData method in the LabelPrinter class requires complete label commands (both system level commands and label format commands).

#### Return value

Returns CLS\_SUCCESS(0) on success, an error code otherwise. See "[2.1 Return value](#)" for the error codes.

#### Example

To add a box command,  
1x1100002000100b0300025001000100:

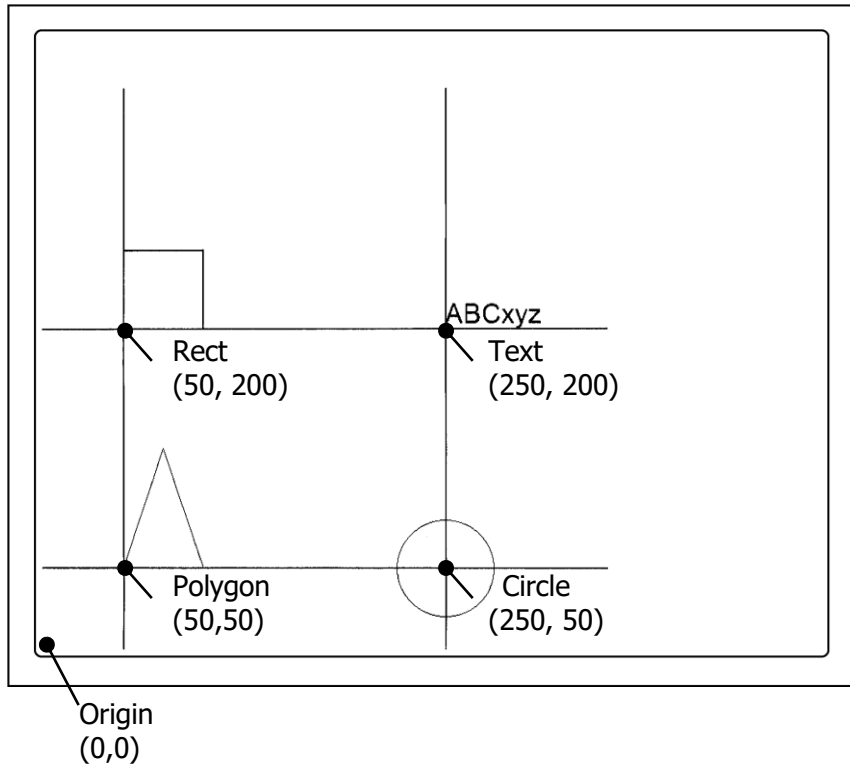
```
byte[] hexdata = new byte[] {
    0x31, 0x58, 0x31, 0x31, 0x30, 0x30, 0x30, 0x30, 0x32, 0x30,
    0x30, 0x30, 0x31, 0x30, 0x30, 0x62, 0x30, 0x33, 0x30, 0x30,
    0x30, 0x32, 0x35, 0x30, 0x30, 0x31, 0x30, 0x30, 0x30, 0x31,
    0x30, 0x30, 0x0D, 0x0A };

design.embedRawDesignCommand(hexdata);
```

## 3. Appendix

### 3.1. Specifying object position

The coordinate origin of the label design is at bottom-left. The measurement unit, either inches or millimeters, is set in the printer. The MeasurementUnit property in the LabelPrinter class allows setting the measurement unit.



#### Example

```
// Polygon
int[] x = new int[] {50, 75, 100};
int[] y = new int[] {50, 125, 50};
design.drawPolygon(x, y);

// Rect
design.drawRect(50, 200, 50, 50, 1);

// Circle
design.drawCircle(250, 50, 30);

// Text
design.drawTextLocalFont("ABCxyz", Typeface.SERIF,
    LabelConst.CLS_RT_NORMAL, 100, 100, 12,
    LabelConst.CLS_FNT_DEFAULT, 250, 200);

// Line
design.drawLine(0, 200, 350, 200, 1);      // Line-1
design.drawLine(50, 0, 50, 350, 1);        // Line-2
design.drawLine(0, 50, 350, 50, 1);        // Line-3
design.drawLine(250, 0, 250, 350, 1);      // Line-4
```

### 3.2. Logging function

This SDK has a logging function which keeps history of executing methods or reading/writing properties. This can be enabled by using the [setLog method](#), or placing a file "CSJLabelLib.cfg" in the /temp/citizen at the top of the external storage path.

On Android 11 or later, to access the above path, you need to grant the application "Access to all files (MANAGE\_EXTERNAL\_STORAGE) permission" and allow the permission in the device's app settings.

Please refer to the attached sample "AndroidManifest.xml" for how to define the application.

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

< Example of CSJLabelLib.cfg >

[LogSetting]	...Section name (Fixed)
LogMode=1	...Specifies the log mode.
LogPath=/storage/emulated/0/Download/Log	...Specifies the folder to output the log files.
LogMaxSize=10	...Specifies the maximum size of log file in MB.

The external storage path depends on the device. For many terminals, the path to place the file will be one of the following.

```
/storage/sdcard0/temp/citizen/Log/
or
/storage/emulated/0/temp/citizen/Log/
```

#### Setting items

##### - LogMode

Specifies a log mode:

- 0: None
- 1: Access log
- 2: Error log

##### - LogPath

Specifies a folder to store the log files.

If a full path starting with /storage/ is specified for the storage folder, the files will be stored in the specified path. Otherwise, it is specified as the path from the top of the external storage path. "/temp/citizen" of the external storage path by default.

##### - LogMaxSize

Specifies maximum size to log file in MB. "0" sets the size to "unlimited."

#### Log file name

Log files will be stored with a file name "CSJLabelLib\_" and a number which indicates the day of week(0 to 6. 0: Sunday, 1: Monday...), and a file extension ".log."

Example: CSJLabelLib\_1.log

When the same file name exists, the file will be overwritten if the file is one week older, new logs will be added to the existing file if the file is created on the same day.

#### Log format

The log file keeps the information of executed methods, accessed properties, timestamps and results.

```
--- Example 1, method (Connect) ---
2016/03/10 17:14:25.353 001 METHOD call    connect(0, "192.168.129.130")
2016/03/10 17:14:25.474 001 METHOD result connect() -> Success(0)
```

```

--- Example 2, method (PrintText) ---

2016/03/10 17:14:25.474 001 METHOD call  print([See below], 1)
-----Parameter Detail-----
drawTextPtrFont("Sample Print", 0, 10, 1, 1, 1, 8, 20, 300) -> 0
drawQRCode("DrawQRCode", 850, 1, 4, 3, 20, 220) -> 0
fillRect(20, 150, 350, 40, 11) -> 0
drawBarCode("0123456789", 104, 1, 3, 3, 30, 20, 70, 1) -> 0
-----
2016/03/10 17:14:25.490 001 METHOD result print() -> Success(0)

--- Example, set to a property ---

2016/03/10 17:14:25.474 001 PROPERTY get  printDarkness -> 999999

--- Example, get from a property ---

2016/03/10 17:14:25.474 001 PROPERTY set  printDarkness <- 10 : Success(0)

```

- \* The logging function could be a "bottleneck" of processing since it tries to keep the information of every single execution of a method or access to a property.
- \* Logging could fail without a notification for the reason below.
  - A write-protected device is selected as a storage location.
  - No enough space in the selected storage device.
  - The storage location contains a file with write-protection.
  - No access privilege to a file or folder.
  - Another program is using the log file.

### 3.3. Predefined Constants

No	Type	Name	Data Type	Value	Description
1	Result/Error	CLS_SUCCESS	int	0	Successfully completed
		CLS_E_CONNECTED	int	1001	Already connected
		CLS_E_DISCONNECT	int	1002	Not connected
		CLS_E_NOTCONNECT	int	1003	Failed to connect
		CLS_E_CONNECT_NOTFOUND	int	1004	Non supported model
		CLS_E_NOCONTEXT	int	1006	No context specified
		CLS_E_BT_DISABLE	int	1007	Bluetooth device setting invalid
		CLS_E_BT_NODEVICE	int	1008	Bluetooth device not found
		CLS_E_ILLEGAL	int	1101	Unsupported or invalid parameter
		CLS_E_OFFLINE	int	1102	Off-line
		CLS_E_NOEXIST	int	1103	File does not exist
		CLS_E_FAILURE	int	1104	Process failure
		CLS_E_TIMEOUT	int	1105	Timeout
		CLS_E_NO_LIST	int	1106	Printer cannot be found
		CLS_EPTR_BADFORMAT	int	1203	Unsupported file format
2	Property default value	CLS_PROPERTY_DEFAULT	int	999999	Default value of property
3	Status of printer	CLS_STS_NO	int	0	Status: NO
		CLS_STS_YES	int	1	Status: YES
4	Connection interface	CLS_PORT_WiFi	int	0	Network
		CLS_PORT_Bluetooth	int	1	Bluetooth
		CLS_PORT_Bluetooth_Insecure	int	2	Bluetooth Insecure
		CLS_PORT_USB	int	3	USB
5	Serial communication condition	CLS_COM_BAUDRATE_1200	int	1200	Baud rate : 1200
		CLS_COM_BAUDRATE_2400	int	2400	Baud rate : 2400
		CLS_COM_BAUDRATE_4800	int	4800	Baud rate : 4800
		CLS_COM_BAUDRATE_9600	int	9600	Baud rate : 9600
		CLS_COM_BAUDRATE_19200	int	19200	Baud rate : 19200
		CLS_COM_BAUDRATE_38400	int	38400	Baud rate : 38400
		CLS_COM_BAUDRATE_57600	int	57600	Baud rate : 57600
		CLS_COM_BAUDRATE_115200	int	115200	Baud rate : 115200
		CLS_COM_PARITY_NONE	int	0	Parity : NONE
		CLS_COM_PARITY_ODD	int	1	Parity : ODD
		CLS_COM_PARITY_EVEN	int	2	Parity : EVEN
		CLS_COM_HANDSHAKE_DTRDSR	int	0	Flow control : DTR/DSR
		CLS_COM_HANDSHAKE_XONXOFF	int	1	Flow control : XON/XOFF
6	Locale	CLS_LOCALE_JP	int	0	Japanese model
		CLS_LOCALE_OTHER	int	1	Non-Japanese model
		CLS_LOCALE_CN	int	2	Chinese model
		CLS_LOCALE_KR	int	3	Korean model
7	Font typeface	CLS_PRT_FNT_0	int	0	System font : 0
		CLS_PRT_FNT_1	int	1	System font : 1
		CLS_PRT_FNT_2	int	2	System font : 2
		CLS_PRT_FNT_3	int	3	System font : 3
		CLS_PRT_FNT_4	int	4	System font : 4
		CLS_PRT_FNT_5	int	5	System font : 5

8		CLS_PRT_FNT_6	int	6	System font : 6
		CLS_PRT_FNT_7	int	7	System font : 7
		CLS_PRT_FNT_8	int	8	System font : 8
		CLS_PRT_FNT_TRIUMVIRATE	int	9	Smooth font (Triumvirate)
		CLS_PRT_FNT_TRIUMVIRATE_B	int	10	Smooth font (Triumvirate Bold)
		CLS_PRT_FNT_KANJI	int	11	Kanji (Writing from left to right)
		CLS_PRT_FNT_KANJIT	int	12	Kanji (Writing from top to bottom)
	Font size	CLS_PRT_FNT_SIZE_4	int	0	4pt
		CLS_PRT_FNT_SIZE_5	int	1	5pt
		CLS_PRT_FNT_SIZE_6	int	2	6pt
		CLS_PRT_FNT_SIZE_8	int	3	8pt
		CLS_PRT_FNT_SIZE_10	int	4	10pt
		CLS_PRT_FNT_SIZE_12	int	5	12pt
		CLS_PRT_FNT_SIZE_14	int	6	14pt
		CLS_PRT_FNT_SIZE_18	int	7	18pt
		CLS_PRT_FNT_SIZE_24	int	8	24pt
		CLS_PRT_FNT_SIZE_30	int	9	30pt
		CLS_PRT_FNT_SIZE_36	int	10	36pt
		CLS_PRT_FNT_SIZE_48	int	11	48pt
		CLS_PRT_FNT_KANJI_SIZE_16	int	100	Kanji 16dot
		CLS_PRT_FNT_KANJI_SIZE_24	int	101	Kanji 24dot
		CLS_PRT_FNT_KANJI_SIZE_32	int	102	Kanji 32dot
		CLS_PRT_FNT_KANJI_SIZE_48	int	103	Kanji 48dot
9	Encoding	CLS_ENC_CDPG_DEFAULT	int	0	Default value
		CLS_ENC_CDPG_IBM037	int	37	IBM037 IBM EBCDIC (USA-Canada)
		CLS_ENC_CDPG_IBM437	int	437	IBM437 OEM USA
		CLS_ENC_CDPG_IBM500	int	500	IBM500 IBM EBCDIC (International)
		CLS_ENC_CDPG_IBM737	int	737	ibm737 Greek (DOS)
		CLS_ENC_CDPG_IBM775	int	775	ibm775 Baltic (DOS)
		CLS_ENC_CDPG_IBM850	int	850	ibm850 Western European(DOS)
		CLS_ENC_CDPG_IBM852	int	852	ibm852 Central Europe (DOS)
		CLS_ENC_CDPG_IBM855	int	855	IBM855 OEM Cyrillic
		CLS_ENC_CDPG_IBM857	int	857	ibm857 Turkish (DOS)
		CLS_ENC_CDPG_IBM860	int	860	IBM860 Portuguese (DOS)
		CLS_ENC_CDPG_IBM861	int	861	ibm861 Icelandic (DOS)
		CLS_ENC_CDPG_IBM863	int	863	IBM863 French (Canada)(DOS)
		CLS_ENC_CDPG_IBM865	int	865	IBM865 Norwegian (DOS)
		CLS_ENC_CDPG_CP866	int	866	cp866 Cyrillic (DOS)
		CLS_ENC_CDPG_IBM869	int	869	ibm869 Greek modern (DOS)
		CLS_ENC_CDPG_WINDOWS_874	int	874	windows-874 Thai (Windows)
		CLS_ENC_CDPG_CP875	int	875	cp875 IBM EBCDIC (Greek modern)
		CLS_ENC_CDPG_SHIFT_JIS	int	932	shift_jis Japanese (Shift JIS)
		CLS_ENC_CDPG_GB2312	int	936	gb2312 simplified Chinese (GB2312)
		CLS_ENC_CDPG_KS_C_5601_1987	int	949	ks_c_5601-1987 Korean
		CLS_ENC_CDPG_BIG5	int	950	big5 traditional Chinese (Big5)
		CLS_ENC_CDPG_IBM1026	int	1026	IBM1026 IBM EBCDIC (Turkish Latin-5)
		CLS_ENC_CDPG_UTF_16	int	1200	utf-16 Unicode
		CLS_ENC_CDPG_UNICODEFFFE	int	1201	unicodeFFFE Unicode (Big-Endian)
		CLS_ENC_CDPG_WINDOWS_1250	int	1250	windows-1250 Central European

			(Windows)
CLS_ENC_CDPG_WINDOWS_1251	int	1251	windows-1251 Cyrillic (Windows)
CLS_ENC_CDPG_WINDOWS_1252	int	1252	Windows-1252 Western European (Windows)
CLS_ENC_CDPG_WINDOWS_1253	int	1253	windows-1253 Greek (Windows)
CLS_ENC_CDPG_WINDOWS_1254	int	1254	windows-1254 Turkish (Windows)
CLS_ENC_CDPG_WINDOWS_1255	int	1255	windows-1255 Hebrew (Windows)
CLS_ENC_CDPG_WINDOWS_1256	int	1256	windows-1256 Arabic (Windows)
CLS_ENC_CDPG_WINDOWS_1257	int	1257	windows-1257 Baltic (Windows)
CLS_ENC_CDPG_WINDOWS_1258	int	1258	windows-1258 Vietnamese (Windows)
CLS_ENC_CDPG_JOHAB	int	1361	Johab Korean (Johab)
CLS_ENC_CDPG_MACINTOSH	int	10000	macintosh Western European (Mac)
CLS_ENC_CDPG_X_MAC_JAPANESE	int	10001	x-mac-japanese Japanese (Mac)
CLS_ENC_CDPG_X_MAC_CHINESETRAD	int	10002	x-mac-chinesetrad traditional Chinese (Mac)
CLS_ENC_CDPG_X_MAC_KOREAN	int	10003	x-mac-korean Korean (Mac)
CLS_ENC_CDPG_X_MAC_GREEK	int	10006	x-mac-greek Greek (Mac)
CLS_ENC_CDPG_X_MAC_CYRILLIC	int	10007	x-mac-cyrillic Cyrillic (Mac)
CLS_ENC_CDPG_X_MAC_CHINESEIMP	int	10008	x-mac-chinesesimp simplified Chinese (Mac)
CLS_ENC_CDPG_X_MAC_ROMANIAN	int	10010	x-mac-romanian Romanian (Mac)
CLS_ENC_CDPG_X_MAC_UKRAINIAN	int	10017	x-mac-ukrainian Ukrainian (Mac)
CLS_ENC_CDPG_X_MAC_CE	int	10029	x-mac-ce Central Europe (Mac)
CLS_ENC_CDPG_X_MAC_ICELANDIC	int	10079	x-mac-icelandic Icelandic (Mac)
CLS_ENC_CDPG_X_MAC_TURKISH	int	10081	x-mac-turkish Turkish (Mac)
CLS_ENC_CDPG_X_MAC_CROATIAN	int	10082	x-mac-croatian Croat (Mac)
CLS_ENC_CDPG_X_CHINESE_CNS	int	20000	x-Chinese-CNS traditional Chinese (CNS)
CLS_ENC_CDPG_US_ASCII	int	20127	us-ascii US-ASCII
CLS_ENC_CDPG_X_CP20261	int	20261	x-cp20261 T.61
CLS_ENC_CDPG_IBM290	int	20290	IBM290 IBM EBCDIC (Japanese Katakana)
CLS_ENC_CDPG_KOI8_R	int	20866	koi8-r Cyrillic (KOI8-R)
CLS_ENC_CDPG_EUC_JP_JIS	int	20932	EUC-JP Japanese (JIS 0208-1990 and 0212-1990)
CLS_ENC_CDPG_X_CP20936	int	20936	x-cp20936 simplified Chinese (GB2312-80)
CLS_ENC_CDPG_X_CP20949	int	20949	x-cp20949 Korean Wansung
CLS_ENC_CDPG_X_CP21027	int	21027	x-cp21027 Ext Alpha Lowercase
CLS_ENC_CDPG_KOI8_U	int	21866	koi8-u Cyrillic (KOI8-R)
CLS_ENC_CDPG_ISO_8859_1	int	28591	iso-8859-1 Western European (ISO)
CLS_ENC_CDPG_ISO_8859_2	int	28592	iso-8859-2 Central Europe (ISO)
CLS_ENC_CDPG_ISO_8859_4	int	28594	iso-8859-4 Baltic (ISO)
CLS_ENC_CDPG_ISO_8859_5	int	28595	iso-8859-5 Cyrillic (ISO)
CLS_ENC_CDPG_ISO_8859_7	int	28597	iso-8859-7 Greek (ISO)
CLS_ENC_CDPG_ISO_8859_9	int	28599	iso-8859-9 Turkish (ISO)
CLS_ENC_CDPG_ISO_8859_13	int	28603	iso-8859-13 Estonian (ISO)
CLS_ENC_CDPG_ISO_8859_15	int	28605	iso-8859-15 Latin 9 (ISO)
CLS_ENC_CDPG_ISO_2022_JP	int	50220	iso-2022-jp Japanese (JIS)
CLS_ENC_CDPG_CSISO2022JP	int	50221	csISO2022JP Japanese (JIS-Allow 1 byte Kana)



		CLS_ENC_CDPG_ISO_2022_JP_S	int	50222	iso-2022-jp Japanese (JIS-Allow 1 byte Kana - SO/SI)
		CLS_ENC_CDPG_ISO_2022_KR	int	50225	iso-2022-kr Korean (ISO)
		CLS_ENC_CDPG_X_CP50227	int	50227	x-cp50227 simplified Chinese (ISO-2022)
		CLS_ENC_CDPG_EUC_JP	int	51932	euc-jp Japanese (EUC)
		CLS_ENC_CDPG_EUC_CN	int	51936	EUC-CN simplified Chinese (EUC)
		CLS_ENC_CDPG_EUC_KR	int	51949	euc-kr Korean (EUC)
		CLS_ENC_CDPG_HZ_GB_2312	int	52936	hz-gb-2312 simplified Chinese (HZ)
		CLS_ENC_CDPG_GB18030	int	54936	GB18030 simplified Chinese (GB18030)
		CLS_ENC_CDPG_UTF_7	int	65000	utf-7 Unicode (UTF-7)
		CLS_ENC_CDPG_UTF_8	int	65001	utf-8 Unicode (UTF-8)
10	Font style	CLS_FNT_DEFAULT	int	0	None
		CLS_FNT_BOLD	int	8	Bold
		CLS_FNT_REVERSE	int	16	Reverse
		CLS_FNT_UNDERLINE	int	128	Underline
		CLS_FNT_ITALIC	int	256	Italic
		CLS_FNT_STRIKEOUT	int	512	Strikethrough
11	Direction of rotation	CLS_RT_NORMAL	int	1	No rotation
		CLS_RT_RIGHT90	int	2	Rotate 90 CW
		CLS_RT_ROTATE180	int	3	Rotate 180
		CLS_RT_LEFT90	int	4	Rotate 90 CCW
12	Barcode symbology	CLS_BCS_CODE39	int	100	Code 3 of 9
		CLS_BCS_UPCA	int	101	UPC-A
		CLS_BCS_UPCE	int	102	UPC-E
		CLS_BCS_INTERLEAVED25	int	103	Interleaved 2 of 5
		CLS_BCS_CODE128	int	104	Code 128
		CLS_BCS_EAN13	int	105	EAN-13(JAN-13)
		CLS_BCS_EAN8	int	106	EAN-8(JAN-8)
		CLS_BCS_HIBC	int	107	HIBC
		CLS_BCS_CODABAR	int	108	CODABAR(NW-7)
		CLS_BCS_INT25	int	109	Int 2 of 5
		CLS_BCS_PLESSEY	int	110	Plessey
		CLS_BCS_CASECODE	int	111	CASE CODE
		CLS_BCS_UPC2DIG	int	112	UPC 2DIG ADD (supplementary code of 2 digits for UPC)
		CLS_BCS_UPC5DIG	int	113	UPC 5DIG ADD (supplementary code of 5 digits for UPC)
		CLS_BCS_CODE93	int	114	Code93
		CLS_BCS_ITF14	int	115	(Japanese model)ITF-14
		CLS_BCS_ZIP	int	116	(Non-Japanese model)ZIP
		CLS_BCS_ITF16	int	117	(Japanese model)ITF-16
		CLS_BCS_UCCEAN128	int	118	(Non-Japanese model)UCC/EAN-128
		CLS_BCS_INDUSTRIAL25	int	119	(Japanese model)Industrial 2 of 5
		CLS_BCS_UCCEAN128KMART	int	120	(Non-Japanese model) UCC/EAN-128(for K-MART)
		CLS_BCS_COOP25	int	121	(Japanese model)COOP 2 of 5
		CLS_BCS_UCCEAN128RANDOMWEIGHT	int	122	(Non-Japanese model) UCC/EAN-128 Random Weight

		CLS_BCS_TELEPEN	int	123	Telepen
13	Human readable text of barcode	CLS_BCS_TEXT_HIDE	int	0	Hide
		CLS_BCS_TEXT_SHOW	int	1	Show
14	Error correction level (PDF417)	CLS_PDF417_EC_LEVEL_0	int	0	Level 0
		CLS_PDF417_EC_LEVEL_1	int	1	Level 1
		CLS_PDF417_EC_LEVEL_2	int	2	Level 2
		CLS_PDF417_EC_LEVEL_3	int	3	Level 3
		CLS_PDF417_EC_LEVEL_4	int	4	Level 4
		CLS_PDF417_EC_LEVEL_5	int	5	Level 5
		CLS_PDF417_EC_LEVEL_6	int	6	Level 6
		CLS_PDF417_EC_LEVEL_7	int	7	Level 7
		CLS_PDF417_EC_LEVEL_8	int	8	Level 8
15	Error correction level (Data Matrix)	CLS_DATAMATRIX_EC_LEVEL_200	int	200	
16	Error correction level (QR Code)	CLS_QRCODE_EC_LEVEL_L	int	0	Error correction level L (7%)
		CLS_QRCODE_EC_LEVEL_M	int	1	Error correction level M (15%)
		CLS_QRCODE_EC_LEVEL_Q	int	2	Error correction level Q (25%)
		CLS_QRCODE_EC_LEVEL_H	int	3	Error correction level H (30%)
17	Error correction level (Aztec)	CLS_AXTEC_EC_LEVEL_000	int	0	Error correction ratio 23%
18	Barcode symbology (GS1 DataBar)	CLS_GS1_DATABAR_OMNI_DIRECTIONAL	int	0	GS1DataBar Omni-directional
		CLS_GS1_DATABAR_COMPOSITE	int	1	GS1DataBar Composite
		CLS_GS1_DATABAR_TRUNCATION	int	2	GS1DataBar Truncation
		CLS_GS1_DATABAR_STACKED	int	3	GS1DataBar Stacked
		CLS_GS1_DATABAR_STACKED_OMNI_DIRECTIONAL	int	4	GS1DataBar Stacked Omni-directional
		CLS_GS1_DATABAR_LIMITED	int	5	GS1DataBar Limited
		CLS_GS1_DATABAR_EXPANDED	int	6	GS1DataBar Expanded
19	Shaded pattern	CLS_SHADED_PTN_0	int	0	Shaded pattern : 000
		CLS_SHADED_PTN_1	int	1	Shaded pattern : 001
		CLS_SHADED_PTN_2	int	2	Shaded pattern : 002
		CLS_SHADED_PTN_3	int	3	Shaded pattern : 003
		CLS_SHADED_PTN_4	int	4	Shaded pattern : 004
		CLS_SHADED_PTN_5	int	5	Shaded pattern : 005
		CLS_SHADED_PTN_6	int	6	Shaded pattern : 006
		CLS_SHADED_PTN_7	int	7	Shaded pattern : 007
		CLS_SHADED_PTN_8	int	8	Shaded pattern : 008
		CLS_SHADED_PTN_9	int	9	Shaded pattern : 009
		CLS_SHADED_PTN_10	int	10	Shaded pattern : 010
		CLS_SHADED_PTN_11	int	11	Shaded pattern : 011
20	Measurement unit	CLS_UNIT_MILLI	int	0	Metric
		CLS_UNIT_INCH	int	1	Inch
21	Speed setting	CLS_SPEEDSETTING_1	int	1	1 : 1.0inch( 25.4mm) / sec
		CLS_SPEEDSETTING_2	int	2	2 : 2.0inch( 50.8mm) / sec
		CLS_SPEEDSETTING_3	int	3	3 : 3.0inch( 76.2mm) / sec
		CLS_SPEEDSETTING_4	int	4	4 : 4.0inch(101.6mm) / sec
		CLS_SPEEDSETTING_5	int	5	5 : 5.0inch(127.0mm) / sec

		CLS_SPEEDSETTING_6	int	6	6 : 6.0inch(152.4mm) / sec
		CLS_SPEEDSETTING_7	int	7	7 : 7.0inch(177.8mm) / sec
		CLS_SPEEDSETTING_8	int	8	8 : 8.0inch(203.2mm) / sec
		CLS_SPEEDSETTING_9	int	9	9 : 9.0inch(228.6mm) / sec
		CLS_SPEEDSETTING_A	int	10	A : 1.0inch( 25.4mm) / sec
		CLS_SPEEDSETTING_B	int	11	B : 1.0inch( 25.4mm) / sec
		CLS_SPEEDSETTING_C	int	12	C : 2.0inch( 50.8mm) / sec
		CLS_SPEEDSETTING_D	int	13	D : 2.0inch( 50.8mm) / sec
		CLS_SPEEDSETTING_E	int	14	E : 3.0inch( 76.2mm) / sec
		CLS_SPEEDSETTING_F	int	15	F : 3.0inch( 76.2mm) / sec
		CLS_SPEEDSETTING_G	int	16	G : 4.0inch(101.6mm) / sec
		CLS_SPEEDSETTING_H	int	17	H : 4.0inch(101.6mm) / sec
		CLS_SPEEDSETTING_I	int	18	I : 5.0inch(127.0mm) / sec
		CLS_SPEEDSETTING_J	int	19	J : 5.0inch(127.0mm) / sec
		CLS_SPEEDSETTING_K	int	20	K : 6.0inch(152.4mm) / sec
		CLS_SPEEDSETTING_L	int	21	L : 6.0inch(152.4mm) / sec
		CLS_SPEEDSETTING_M	int	22	M : 7.0inch(177.8mm) / sec
		CLS_SPEEDSETTING_N	int	23	N : 7.0inch(177.8mm) / sec
		CLS_SPEEDSETTING_O	int	24	O : 8.0inch(203.2mm) / sec
		CLS_SPEEDSETTING_P	int	25	P : 8.0inch(203.2mm) / sec
		CLS_SPEEDSETTING_Q	int	26	Q : 9.0inch(228.6mm) / sec
		CLS_SPEEDSETTING_R	int	27	R : 9.0inch(228.6mm) / sec
		CLS_SPEEDSETTING_S	int	28	S : 10.0inch(254.0mm) / sec
		CLS_SPEEDSETTING_T	int	29	T : 10.0inch(254.0mm) / sec
		CLS_SPEEDSETTING_U	int	30	U : 11.0inch(279.4mm) / sec
		CLS_SPEEDSETTING_V	int	31	V : 11.0inch(279.4mm) / sec
		CLS_SPEEDSETTING_W	int	32	W : 12.0inch(304.8mm) / sec
		CLS_SPEEDSETTING_X	int	33	X : 12.0inch(304.8mm) / sec
22	Media handling	CLS_MEDIAHANDLING_NONE	int	0	None
		CLS_MEDIAHANDLING_TEAROFF	int	1	Tear-off
		CLS_MEDIAHANDLING_DISPENSES	int	2	Dispense
		CLS_MEDIAHANDLING_PAUSE	int	3	Pause
		CLS_MEDIAHANDLING_CUT	int	4	Cut
		CLS_MEDIAHANDLING_CUTANDPAUSE	int	5	Cut and Pause
		CLS_MEDIAHANDLING_PEELOFF	int	6	Peel-off
		CLS_MEDIAHANDLING_REWIND	int	7	Rewind
23	Label sensor	CLS_SELSENSOR_NONE	int	0	None
		CLS_SELSENSOR_SEETHROUGH	int	1	See Through
		CLS_SELSENSOR_REFLECT	int	2	Reflect
24	Print method	CLS_PRTMETHOD_TT	int	0	Thermal transfer
		CLS_PRTMETHOD_DT	int	1	Direct thermal
25	Sensor location	CLS_SENS_LOCATION_FRONT	int	0	Front sensor
		CLS_SENS_LOCATION_ADJUSTABLE	int	1	Rear sensor

CITIZEN Android Label Print SDK - Programming Manual

October 22, 2025 Version 1.05

CITIZEN SYSTEMS JAPAN CO., LTD.

<http://www.citizen-systems.co.jp/>