SATO SBPL providers

Version 1.0.0

User's Guide

February 14, 2020

NOTE:

This document is translated into English by machine translation.

[Revision History]

Version	Date	Description
1.0.0	2020-02-14	First edition

[Operation check device]

Model	POINTS OF CAUTION
CL4NX-J	Supports TCP/IP connections. Model name: CL4NX-J 305dpi Main firmware version: 1.9.2-r1

Index

1. Introduction

This manual is the user's guide of the CAO provider for controlling the printer made by SATO CORPORATION (hereinafter referred to as SATO CORPORATION) through SBPL[1] commands. Some of the commands supported by this provider are not supported by printers made by SATO, so please use them after checking 6.2 Command reference table. Command reference table

The CAO provider (CaoProvSATOSBPL.dll) described in this document is called SBPL provider. The functions and variables provided by the provider are described in Chapter 2 and later.2

[1] SBPL(SATO Barcode Printer Language) is a common command for controlling SATO's label printer.

2. Provider Overview

2.1. Introduction

SBPL providers provide a CaoController::Execute method for executing commands. CaoController::Execute uses Ethernet interface to send and receive SBPL commands.

2.2. Operating system installation

SBPL providers have a DLL (Dynamic Link Library) file format that is dynamically loaded from the CAO engine when used. You must install ORiN2SDK or manually register the registry by referring to Table 2-1.

Table 2-1 SBPL Provider

File name	CaoProvSATOSBPL.dll
ProgID	CaoProv.SATO.SBPL
Registry registration	Regsvr32 CaoProvSATOSBPL.dll
Deletion of Registry Registration	Regsvr32 /u CaoProvSATOSBPL.dll

2.3. Method Properties

2.3.1. CaoWorkspace::AddController method

SBPL providers refer to the connection parameters at the time of AddController to connect the communication. At this time, specify the connection destination and timeout with the option.

Format AddController (<bstrCtrlName:VT_BSTR>,<bstrProvName:VT_BSTR>,
<bstrPcName:VT_BSTR>, [<bstrOption:VT_BSTR>])

<bstrCtrlName>	:	[in] Controller name
<bstrProvName>	:	[in] Provider name. Fixed value = "CaoProv.SATO.SBPL"
<bstrPcName>	:	[in] Running machine name of the provider (not used)
<bstrOption>	:	[in] option string

The following is the list specified in the option string.

Table 2-2 Optional strings for CaoWorkspace::AddController

OPTIONS	Meaning
Conn = <connection-parameter>	Required. Set the communication mode and its connection parameters. (Refer to 2.3.1.1)2.3.1.1
ConnTimeout[=<timeout>]	Optional. Specifies the timeout in milliseconds for TCP connections. (Default: 500)
Timeout[=<timeout>]	Optional. Specifies the timeout time for command transmission/reception in milliseconds. (Default: 3000)

2.3.1.1. Conn Options

The following are Conn optional connect parameter strings: Square brackets ("[]") indicate optional characters. The underlined portion of the explanation of each parameter indicates the default value when no option is specified.

[Ethernet Devices]

"Conn=ETH:<Dest IP Address>[:<Dest Port No>]"

"Conn=TCP:<Dest IP Address>[:<Dest Port No>]"

<Dest IP Address>	:	IP address of the device to connect to.
<Dest Port No>	:	The connection port number. 9100,9101,...Optional

Examples of use

```
CaoEngine caoEng;
CaoWorkspaces caoWss;
CaoWorkspace caoWs;
CaoControllers caoCtrls;
CaoController caoCtrl;

CaoEng = new CaoEngine();
CaoWss = caoEng.Workspaces;
CaoWs = caoWss.Item(0);
CaoCtrls = caoWs.Controllers;

// Connect by LAN.
CaoCtrl = caoWs.AddController("SBPL_SAMPLE", "CaoProv.SATO.SBPL", null,
                                         "Conn=TCP:192.168.1.2:9100,Timeout=3000");
```

2.3.2. CaoController::AddVariable method

AddVariable method for CaoController classes is a method for creating variable objects. Only the variable names shown in Table 2-3 can be used as variable names.

Format AddVariable (<bstrVariableName:VT_BSTR>, [<bstrOption:VT_BSTR>])

<bstrVariableName>	:	[in] Variable name
<bstrOption>	:	[in] option string

Examples of use

```
// Add variable to get device version
CaoVariable val;
Val = caoCtrl.AddVariable("@DEVICE_VERSION");

// GetValue
Var value = val.Value;
/* example 1.9.2-r1 */
```

2.3.3. CaoController::GetVariableNames method

Retrieves a list of system variable names that can be specified by AddVariable method.

2.3.4. CaoController::Execute method

Execute method of CaoController classes is a method for executing commands. For details of each command, refer to Command Reference.

Format Execute (<bstrCommandName:VT_BSTR>,[<vntParam:VT_VARIANT>])

< bstrCommandName>	:	[in] Command name
<vntParam>	:	[in] Parameters

2.4. List of Variables

2.4.1. CaoController classes

Table 2-3 Table List of System Variables for CaoController Class

Variable name	Data Type	Description	Attribute	
			Get	Put
@MAKER_NAME	VT_BSTR	Returns the manufacturer name="SATO".	✓	-
@VERSION	VT_BSTR	Returns the provider version.	✓	-
@DEVICE_VERSION	VT_BSTR	Returns the firmware version.	✓	-

2.5. Error Codes

SBPL providers return the following unique error codes:

Table 2-4 Unique error codes

Error name	Error Number	Description
No response	0x80100000	Returns when data cannot be received from the printer. Make sure that the printer's communication settings and connection options match.
Received data error	0x80100001	It returns if you receive unexpected data, such as missing data.
Lock specification error	0x80100002	CaoController::Execute("WriteRfidPassLock") Command Returned when "Lock area unspecified (0,0,0,0,0)" is entered in the lock area specifying parameter at the time of the command.
Error response	0x801001XX	Returns when an error is received as the response to the command. The hexadecimal error code[2] received from the printer is inserted into the XX. Ex.) 22 → 0x80100122

[2] For details of error codes, refer to the user's manual of each printer.

3. Limitations

3.1. Communication Protocol

SBPL provider uses the communication protocol status 4 (ENQ response mode). Before using SBPL provider, make sure that the printer's communication protocol is set to status 4 (ENQ response mode).

3.2. Protocol code

SBPL providers control through SBPL commands. Make sure that Protocol Code is set to Enable as SBPL setting for the printer. When disabled, SBPL providers do not operate properly.

3.3. Command transmission interval

Depending on the specifications of the device, it is necessary to send commands continuously at regular intervals.

Table 3-1 Limitation of command sending interval

Processing	Transmit interval (msec)
Get the value of the system variable @DEVICE_VERSION (get_Value property)	5000
CaoController::Execute("Enq") Command	5

3.4. Registering the Print Format

CaoController::Execute("PrintFormat") The command embeds print data in the format registered in the printer in advance. Therefore, be sure to register the print format before using CaoController::Execute("PrintFormat command.

The print format can be registered by using SBPL command sending function of the printer setting tool "All-In-OneTool"[3], or by executing the "Specify Format" and "Specify Field Registration" commands with CaoController::Execute ("Raw") command.

3.5. Status request

CaoController::Execute("Enq") The command can send a status request, but do not request a status while sending print data (STX<A> to <Z>ETX). If a status request is sent while the print data is being transmitted, the status may not be acquired normally or the data may not be printed correctly. If you want to make a status request after the print data transmission is completed, adjust the transmission interval with the actual machine before using.

Reference: When CL4NX-J is connected, normal operation is checked at intervals of 200 ms or more.

[3] All-In-OneTool can be downloaded from Sato's products.

3.6. RFID related commands

This function is available only for RFID compatible models. For RFID compatible models, check the SATO website.

4. Command Reference

4.1. CaoController classes

Table 4-1 CaoController Class Commands

Command	Function	Page
PrintFormat	Format printing instructions.	12
PrintBarcodeEAN13	Bar code (EAN13) printing command.	13
PrintQRCode	QR code (Model 2) print command.	14
PrintRfidUhf	RFID(UHF) Print command.	15
PrintRfidUhfForFormat	Formatted print RFID (UHF) print command.	16
WriteRfidPassLock	RFID data write command with password-lock function.	17
WriteRfidPermLock	RFID data write with permanent lock function.	18
Enq	Status request.	19
Raw	Raw data transmission.	19
GetStatus	Obtain printer status information.	20

4.1.1. CaoController::Execute("PrintFormat") Command

Print using the registered format. Refer to 3.4 before use.

Format PrintFormat(<PrintParam>)

<PrintParam> : [in] print parameters

VT_ARRAY | VT_VARIANT			
0	VT_UI2	The format number.	Required.
1	VT_ARRAY | VT_BSTR	Print data. Fill in the template by field number. 1 to 99 characters can be specified in one field. The number of fields is 99.	Required.
2	VT_UI4	Optional. Number of prints (VT_UI4). Specify a number between 1 and 999999. (Default: 1)	Optional.

Return Values : None

Example 1

```
Var printData = new string[] {"ABCDE", "12345"};
// Print 10 sheets using format number 1
CaoCtrl.Execute("PrintFormat", new object[]{(ushort)1, printData, 10U});
```

Example 2 To print nothing in the specified field, set the field value as blank.

```
Var printData = new string[] {"", "12345"};
// Print 10 sheets using format number 1
CaoCtrl.Execute("PrintFormat", new object[]{(ushort)1, printData, 10U});
```

4.1.2. CaoController::Execute("PrintBarcodeEAN13") Command

Prints a bar code (EAN13).

Format PrintBarcodeEAN13(<PrintParam>)

<PrintParam> : [in] print parameters

VT_ARRAY | VT_VARIANT			
0	VT_BSTR	Print data. Specify a number from 0 to 9 with 11 to 13 digits.	Required.
1	VT_UI2	Print position vertical. Specify a number between 1 and 18000dot. (Default: 1)	Optional.
2	VT_UI2	Print position horizontally. Specify a number between 1 and 18000dot. (Default: 1)	Optional.
3	VT_UI4	Number of sheets printed. Specify a number between 1 and 999999. (Default: 1)	Optional.
4	VT_UI2	Narrow bar width. Specify a number from 1 to 12. (Default: 4)	Optional.
5	VT_UI2	Bar code. Specify a number between 1 and 999. (Default: 120)	Optional.

Return Values : None

Example

Print data "123456789" at 100 dots high, 100 dots wide, 3 dots narrowbar width, and 10 dots on 120 dots high barcode

```
// Print 10 sheet with some setting
CaoCtrl.Execute("PrintBarcodeEAN13", new object[]{"12345678901", (ushort)100, (ushort)100, 10U, (ushort)3,
(ushort)120});
```

4.1.3. CaoController::Execute("PrintQRCode") Command

Print QR code (Model 2). The QR code in normal mode (Model 2) can be printed with this command.

Format PrintQRCode (<PrintParam>)

<PrintParam> : [in] print parameters

VT_ARRAY | VT_VARIANT			
0	VT_BSTR	Print data[4]. Number of data is 1 to 2953.	Required.
1	VT_UI2	Print position vertical. Specify a number between 1 and 18000dot. (Default: 1)	Optional.
2	VT_UI2	Print position horizontally. Specify a number between 1 and 18000dot. (Default: 1)	Optional.
3	VT_UI4	Number of sheets printed. Specify a number between 1 and 999999. (Default: 1)	Optional.
4	VT_BSTR	Error correction level . (Default: L) L : 7% M : 15% Q : 25% H : 30%	Optional.
5	VT_UI2	Size of one side of the cell. Specify a number from 1 to 99. (Default: 4)	Optional.
6	VT_UI1	Data setting mode . (Default: 0) 0 : Automatic Configuration 1 : Digit 2 : Alphanumeric characters 3 : Chinese character	Optional.

Return Values : None

[4] When including alphabetic characters, numeric characters, Chinese characters, and all in the print data, set the data setting mode to 0.

Example

Print data "123456789" at 100 dots high, 100 dots wide, error correction level M (15%), 5 dots per cell side, 10 sheets in data setting mode 1 (number)

```
// Print 10 sheet with some setting
CaoCtrl.Execute("PrintQRCode", new object[]{"123456789", (ushort)100, (ushort)100, 10U, "M", (ushort)5,
(byte)1});
```

4.1.4. CaoController::Execute("PrintRfidUhf") Command

Print RFID (UHF).

Format PrintRfidUhf (<PrintParam>)

<PrintParam> : [in] print parameters

VT_ARRAY | VT_VARIANT			
0	VT_BSTR	EPC area registration data.[5] Characters are 4 to 124.	Required.
1	VT_BSTR	Print data. Half-width alphanumeric characters and Chinese characters can be mixed.	Required.
2	VT_UI2	Print position vertical. Specify a number between 1 and 18000dot. (Default: 1)	Optional.
3	VT_UI2	Print position horizontally. Specify a number between 1 and 18000dot. (Default: 1)	Optional.
4	VT_UI4	Number of sheets printed. Specify a number between 1 and 999999. (Default: 1)	Optional.
5	VT_BSTR	Registered data of USER memory area. Characters are 4 to 128.	Optional.

Return Values : None

Example

5 sheets are printed with EPC area registration data 0000, print data "12345ABCD", vertical 1dot, horizontal 1dot, and USER memory area registration data 9999.

```
// Print 5 sheet with some setting
CaoCtrl.Execute("PrintRfidUhf", new object[]{"0000", "12345ABCDE", (ushort)1, (ushort)1, 5U, "9999"});
```

[5] The number of data characters that can be registered in the EPC area depends on the chip specifications of the RF label to be printed. For details, check the specifications of the RF label to be printed.

4.1.5. CaoController::Execute("PrintRfidUhfForFormat") Command

This function performs RFID (UHF) printing using the registered format for printing RFID labels. Refer to 3.4 before use.

Format PrintRfidUhf (<PrintParam>)

<PrintParam> : [in] print parameters

VT_ARRAY | VT_VARIANT			
0	VT_BSTR	EPC area register data [5]. Characters are 4 to 124.	Required.
1	VT_UI2	The format number.	Required.
2	VT_ARRAY | VT_BSTR	Print data. Fill in the template by field number. 1 to 99 characters can be specified in one field. The number of fields is 99.	Required.
3	VT_UI4	Number of sheets printed. Specify a number between 1 and 999999. (Default: 1)	Optional.
4	VT_BSTR	Registered data of USER memory area. Characters are 4 to 128.	Optional.

Return Values : None

Example

Five sheets are printed using the EPC area registration data 0000 and format number 1 formats and USER memory area registration data 9999.

```
 // Print 5 sheet with some setting
Var printData = new string[] {"ABCDE", "12345"};
 CaoCtrl.Execute("PrintRfidUhfForFormat", new object[]{"0000", (ushort)1, printData, 5U, "9999"});
```

4.1.6. CaoController::Execute("WriteRfidPassLock") Command

Password-lock or unlock RFID with lock function.[6]

Format WriteRfidPassLock (<WriteParam>)

<WriteParam> : [in] Write parameter

VT_ARRAY | VT_VARIANT			
0	VT_UI1	Lock type specification. 0 : Lock 1 : Unlock	Required.
1	VT_ARRAY | VT_UI2	Lock area specification. Specify 1 for the area to be locked or unlocked; otherwise, specify 0. The correspondence between the element number of the array and the area is as follows. 0 USER memory area 1 TID region 2 ACCESS area 3 KILL 4 EPC region	Required.
2	VT_BSTR	Specify the access password. Significant number of digits specified as a hexadecimal string of 8 digits.	Required.
3	VT_ARRAY | VT_BSTR	Either the registered data.EPC area or USER memory area must be specified. The correspondence between array element numbers and data is as follows. 0 EPC area register data [5].[5] Characters are 4 to 124. 1 Registered data of USER memory area. Characters are 4 to 128.	Required.

Return Values : None

[6] The lock specifications vary depending on the chip specifications of the RF label used. For details, check the use of RF labels.

Example

EPC area data 12345678, USER memory area data 12345678123456, locks USER memory area

```
  Var lockBank = new ushort[] {1, 0, 0, 0, 0};
  Var writeData = new string[] {"12345678", "12345678123456"};
   CaoCtrl.Execute("WriteRfidPassLock", new object[] {(byte)0, lockBank, "12345678", writeData});
```

4.1.7. CaoController::Execute("WriteRfidPermLock") Command

Performs a permanent lock or permanent unlock on RFID with lock function [6].

Format WriteRfidPermLock (<WriteParam>)

<WriteParam> : [in] Write parameter

VT_ARRAY | VT_VARIANT			
0	VT_ARRAY | VT_UI2	Lock specification. Specifies the type of lock to be executed for each region. 0 : No lock 1 : Permanent lock 2 : Permanent unlock The correspondence between the element number of the array and the area is as follows. 0 | USER memory area 1 | TID region 2 | ACCESS area 3 | KILL 4 | EPC region	Required.
1	VT_ARRAY | VT_BSTR	Either the registered data.EPC area or USER memory area must be specified. The correspondence between array element numbers and data is as follows. 0 | EPC area register data [5]. Characters are 4 to 124. 1 | Registered data of USER memory area. Characters are 4 to 128.	Required.

Return Values : None

Example

Permanently locks USER memory area with EPC area registration data 12345678 and USER memory area registration data 1234567812345678

```
Var lockBank = new ushort[] {1, 0, 0, 0, 0};
Var writeData = new string[] {"12345678", "1234567812345678"};
CaoCtrl.Execute("WriteRfidPermLock", new object[] {lockBank, writeData});
```

4.1.8. CaoController::Execute("Enq") Command

Sends a status request command to the printer to obtain the status of the currently executed print process. Refer to 6.1.1 Status request for printer status codes and details.

Format Enq()

Argument	:	None
Return Values	:	Printer Status (VT_UI1)

Example

```
Byte status = caoCtrl.Execute("Enq");
```

4.1.9. CaoController::Execute("Raw") Command

Sends SBPL command specified in the arguments to the printer. When specifying SBPL command character string, enclose the command code in <>. When sending to printers, prefix STX and suffix CRLF and ETX to SBPL commands specified in the arguments.

Format Raw (<Command>)

<Command>	:	[in] SBPL Commands (VT_BSTR)
Return Values	:	None

Example 1

```
// [ESC + IK] Send a command to feed 120 dots in paper order
CaoCtrl.Execute("Raw", new object[]{"<A><IK>0,120<Z>"});
```

Example 2 To set <> as the print data in 2 SBPL commands, select <<,>.

```
// [ESC + BG] Transmit CODE128 (128A, 128B, 128C) barcode print command
CaoCtrl.Execute("Raw", new object[]{"<A><V>100<H>200<BG>02120>>GABCD123456<Q>2<Z>"});
```

4.1.10. CaoController::Execute("GetStatus") Command

Sends the printer status information acquisition command to the printer to obtain more detailed printer status than CaoController::Execute("Enq") Command. Refer to 6.1.2 Printer status information.

Format GetStatus ()

Argument : None

Return : Printer Status

Values

VT_ARRAY | VT_UI4		
0	VT_UI4	Printer status.
1	VT_UI4	Receive Buffer Status.
2	VT_UI4	Ribbon status.
3	VT_UI4	Paper status.
4	VT_UI4	Error No.
5	VT_UI4	Battery status.
6	VT_UI4	Remaining quantity issued. Up to 999999 sheets.

Example

```
Uint status[7] = caoCtrl.Execute("GetStatus");
```

5. Sample program

The following is a sample program (C#) that uses this provider to communicate with CLNX-J series and then acquires device data or executes Execute commands. Review the 3 Limitations before running.

Sample	Program.cs

```
... (Abbreviated)...
using ORiN2.ManagedCAO;

namespace SATO_SBPL_Sample
{
    Public partial class Sample : Form
    {
        Private CCaoEngine eng;
        Private CCaoWorkspace ws;
        Private CCaoWorkspaces wss;
        Private CCaoController ctrl;
        Private CCaoControllers ctrls;
        Private CCaoVariable makerName;
        Private CCaoVariable provVersion;
        Private CCaoVariable deviceVersion;

        Public Sample()
        {
            InitializeComponent();
        }

        Private void Program_Load(object sender, EventArgs e)
        {
            // Create CAO Engine
            This.eng = new CCaoEngine();
            This.wss = this.eng.Workspaces;
            This.ws = this.wss[0];
            This.ctrls = this.ws.Controllers;
        }

        Private void btnConnect_Click(object sender, EventArgs e)
        {
            Try
            {
                If (this.ctrl != null)
                {
                    This.ws.Controllers.Remove(this.ctrl.Index);
                    This.ctrl = null;
                }
            // Connect CL4NX-J
                // option:Conn=TCP:192.168.51.20:9100, ConnTimeout=1000, TimeOut=1000
                This.ctrl = this.ws.AddController("Sample",
                                                  "CaoProv.SATO.SBPL",
                                                  Null,
                                                  TextConnOption.Text);

                // Add system variable
                MakerName = this.ctrl.AddVariable("@MAKER_NAME", null);
                ProvVersion = this.ctrl.AddVariable("@VERSION", null);
                DeviceVersion = this.ctrl.AddVariable("@DEVICE_VERSION", null);
            }
            Catch (Exception ex)
            {
                MessageBox.Show(this, ex.Message, this.Text,
                                        MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
```

```
}

Private void btnGetVariable_Click(object sender, EventArgs e)
{
    Try
    {
        // GetValue
        TextMakerName.Text = Convert.ToString(makerName.Value);
        TextVersion.Text = Convert.ToString(provVersion.Value);
        TextDeviceVersion.Text = Convert.ToString(deviceVersion.Value);
    }
    Catch (Exception ex)
    {
        MessageBox.Show(this, ex.Message, this.Text,
                                    MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

Private void btnExecute_Click(object sender, EventArgs e)
{
    Try
    {
        Switch (comboBox1.SelectedIndex)
        {
            Case 0:
                TextResult.Text = ExecPrintFormat();
                Break;
            Case 1:
                TextResult.Text = ExecPrintBarcodeEAN13();
                Break;
            Case 2:
                TextResult.Text = ExecPrintQRCode();
                Break;
            Case 3:
                TextResult.Text = ExecPrintQRCodeJoint();
                Break;
            Case 4:
                TextResult.Text = ExecPrintRfidUhf();
                Break;
            Case 5:
                TextResult.Text = ExecPrintRfidUhfForFormat();
                Break;
            Case 6:
                TextResult.Text = ExecWriteRfidPassLock();
                Break;
            Case 7:
                TextResult.Text = ExecWriteRfidPermLock();
                Break;
            Case 8:
                TextResult.Text = ExecEnq();
                Break;
            Case 9:
                TextResult.Text = ExecRaw();
                Break;
            Case 10:
                TextResult.Text = ExecGetStatus();
                Break;
            Default:
                Break;
        }
    }
    Catch (Exception ex)
    {
        MessageBox.Show(this, ex.Message, this.Text,
                                    MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
```

```
}

Private string ExecPrintFormat()
{
    Var param = new object[] {(ushort)1, new string[] {"ABCDE", "12345"}, 10U};
    Var result = this.ctrl.Execute("PrintFormat", param);
    Return (result != null) ? Convert.ToString(result) : string.Empty;
}

Private string ExecPrintBarcodeEAN13()
{
    Var param = new object[] {"12345678901", (ushort)100, (ushort)100, 10U,
                                            (ushort)3, (ushort)120};
    Var result = this.ctrl.Execute("PrintBarcodeEAN13", param);
    Return (result != null) ? Convert.ToString(result) : string.Empty;
}

Private string ExecPrintQRCode()
{
    Var param = new object[] {"123456789", (ushort)100, (ushort)100, 10U,
                                            "M", (ushort)5, (byte)1};
    Var result = this.ctrl.Execute("PrintQRCode", param);
    Return (result != null) ? Convert.ToString(result) : string.Empty;
}

Private string ExecPrintQRCodeJoint()
{
    Var param = new object[] {"123456789", (ushort)1, (ushort)1, 1U,
                                            "M", (ushort)5, (byte)1, (ushort)4, (byte)1};
    Var result = this.ctrl.Execute("PrintQRCodeJoint", param);
    Return (result != null) ? Convert.ToString(result) : string.Empty;
}

Private string ExecPrintRfidUhf()
{
    Var param = new object[] {"12345678", "12345ABCDE", (ushort)1, (ushort)1,
                                            5U, "9999"};
    Var result = this.ctrl.Execute("PrintRfidUhf", param);
    Return (result != null) ? Convert.ToString(result) : string.Empty;
}

Private string ExecPrintRfidUhfForFormat()
{
    Var param = new object[] {"12345678", (ushort)1,
                                            New string[] {"ABCDE", "12345"}, 5U, null};
    Var result = this.ctrl.Execute("PrintRfidUhfForFormat", param);
    Return (result != null) ? Convert.ToString(result) : string.Empty;
}

Private string ExecWriteRfidPassLock()
{
    Var lockBank = new ushort[] {1, 0, 0, 0, 0};
    Var writeData = new string[] {"12345678", "1234567812345678"};
    Var param = new object[] {(byte)0, lockBank, "12345678", writeData};
    Var result = this.ctrl.Execute("WriteRfidPassLock", param);
    Return (result != null) ? Convert.ToString(result) : string.Empty;
}

Private string ExecWriteRfidPermLock()
{
    Var lockBank = new ushort[] {1, 0, 0, 0, 0};
    Var writeData = new string[] {"12345678", "1234567812345678"};
    Var param = new object[] {lockBank, writeData};
    Var result = this.ctrl.Execute("WriteRfidPermLock", param);
    Return (result != null) ? Convert.ToString(result) : string.Empty;
}
```

```
                Private string ExecEnq()
                {
                    Var result = this.ctrl.Execute("Enq", null);
                    Return (result != null) ? Convert.ToString(result) : string.Empty;
                }

                Private string ExecRaw()
                {
                    Var param = "<A><V>100<H>200<BG>02120>>GABCD123456<Q>2<Z>";
                    Var result = this.ctrl.Execute("Raw", param);
                    Return (result != null) ? Convert.ToString(result) : string.Empty;
                }

                Private string ExecGetStatus()
                {
                    Var result = this.ctrl.Execute("GetStatus", null);
                    If ((result != null) && (result is Array))
                    {
                        Return string.Join(",", (uint[])result);
                    }
                    Return string.Empty;
                }

                Private void Sample_FormClosed(object sender, FormClosedEventArgs e)
                {
                    // Release CAO object
                    If (this.eng != null)
                    {
                        This.eng.Dispose();
                        This.eng = null;
                    }
                }
            }

        }
```

6. APPENDIX

6.1. Printer Status List

6.1.1. Status request

Description			Decimal
Offline status	No error		48
	Ribbon/Label near-end		49
	Buffer near full		50
	Ribbon/Label near-end & Buffer near-full		51
	Printing is stopped (no error)		52
	(unused) battery near end		53
	(unused) battery near-end & ribbon/labelnear end		54
	(Unused) Battery Near End & Buffer Near Full		55
	(Unused) Battery Near End & Ribbon/Label Near End & Buffer Near Full		56
Online status	Waiting for reception	No error	65
		Ribbon/Label near-end	66
		Buffer near full	67
		Ribbon/Label near-end & Buffer near-full	68
		Printing is stopped (no error)	69
		(unused) battery near end	33
		(unused) battery near-end & ribbon/labelnear end	34
		(Unused) Battery Near End & Buffer Near Full	35
		(Unused) Battery Near End & Ribbon/Label Near End & Buffer Near Full	36
	During printing	No error	71
		Ribbon/Label near-end	72
		Buffer near full	73
		Ribbon/Label near-end & Buffer near-full	74
		Printing is stopped (no error)	75
		(unused) battery near end	37
		(unused) battery near-end & ribbon/labelnear end	38
		(Unused) Battery Near End & Buffer Near Full	39
		(Unused) Battery Near End & Ribbon/Label Near End & Buffer Near Full	40
	Waiting (Spider	No error	77
		Ribbon/Label near-end	78

	・ Waiting for cutting)	Buffer near full	79
		Ribbon/Label near-end & Buffer near-full	80
		Printing is stopped (no error)	81
		(unused) battery near end	41
		(unused) battery near-end & ribbon/labelnear end	42
		(Unused) Battery Near End & Buffer Near Full	43
		(Unused) Battery Near End & Ribbon/Label Near End & Buffer Near Full	44
	Analysis ・ Editing	No error[7]	83
		Ribbon/Label Near End [7]	84
		Buffer near full [7]	85
		Ribbon/Label Near End & Buffer Near Full [7]	86
		Print stopped (no errors) [7]	87
		(unused) battery near end	45
		(unused) battery near-end & ribbon/labelnear end	46
		(Unused) Battery Near End & Buffer Near Full	47
		(Unused) Battery Near End & Ribbon/Label Near End & Buffer Near Full	64
Error Detection	Head open		98
	Paper end		99
	Ribbon end		100
	Media error (print jump error)		101
	Sensor error/Paper jam error		102
	Bar code reading/matching error		102
	Bar code reader connection check error		102
	Head error		103
	(Not used) Cover Open		104
	Cutter open error		104
	(Unused) Ribbon core non-lock error		104
	Card error		105
	Cutter error		106
	Other errors		107
	(unused) cutter sensor error		108
	(Unused) Stacker or Rewind Full		109

[7] Depending on the timing of editing and analysis, the number of printed sheets may not be set.

	RFID tag error	110
	RFID protection errors	112
	(Unused) Battery error	113

6.1.2. Printer status information

No.	Meaning	Printer Status Information Name	Printer status information data
1	Printer Status	PS	0: Standby (waiting for reception) 1: Waiting for peeling 2: During the analysis 3: During printing 4: Offline 5: Error occurring
2	Receive Buffer Status	RS	0: Buffer free 1: Buffer near full 2: Buffer full
3	Ribbon status ※ Can be monitored during printing and feeding Accurate values cannot be obtained while operation is stopped.	RE	0: With ribbon 1: Ribbon near end 2: No ribbon 3: Thermal Specifications
4	Paper Status ※ Can be monitored during printing and feeding Accurate values cannot be obtained while operation is stopped.	PE	0: Paper is available (including when starting up) 1: Label near end 2: No paper
5	Error No.[8]	EN	00: Online * Non-error but returned 01: Offline * Non-error but returned 02: Machine error 03: Memory error

[8] The error number also indicates an error that does not occur in this product.

			04: Program errors 05: Setting data error (FLASH-ROM error) 06: Setting data error (EE-PROM error) 07: Download error 08: Parity error 09: Overrun 10: Framing error 11: LAN timeout error 12: Buffer overflow 13: Head open 14: Paper end 15: Ribbon end 16: Media error 17: Sensor error 18: Head error 19: Cover open error 20: Memory/Card type error 21: Memory/Card read/write error 22: Memory/card full error 23: No memory/card battery error 24: Ribbon saver error 25: Cutter error 26: Cutter sensor error 27: Stacker full error 28: Command error 29: Sensor error at power-on 30: RFID tag error 31: Interface card error 32: Rewinder error 33: Other errors 34: RFID control errors 35: Head density error 36: Kanji data error 37: Calendar error 38: Item No. error 39: BCC error

			40: Cutter cover open error 41: Ribbon take-up non-lock error 42: Communication timeout error 43: Lid latch open error 44: Out-of-paper error during power-on 45: SD card access error 46: SD card full error 47: Head lift error 48: Head temperature error 49: SNTP time correction errors 50: CRC error 51: Cutter motor error 53: Scanner read error 54: Scanner Verification Error 55: Scanner connection error 56: Bluetooth module error 57:EAP Authentication Error(EAP failed) 58:EAP Authentication Error(TimeOut) 59: Battery error 60: Low battery error 61: Low-Battery Error (Charging) 62: Battery not installed error 63: Battery temperature error 64: Battery Degraded Error 65: Motor temperature error 66: Temperature error in chassis 67: Jam error 68: SIPL field full error 69: Power-off error during charging 70: WLAN module error 71: Option mismatch error 72: Battery Degraded Error (Caution) 73: Battery Degraded Error (Warning) 74: Unpower off error 75: NonRFID Warnings Errors 76: Bar code reader connection error

			77: Bar code reading error 78: Bar code reading error (abnormal verification start position) 79: Bar code matching error 80: NFC module error 81: NFC command error
6	Battery Status	BT	0: Normal 1: Near end of the battery 2: Battery error
7	Remaining Issued Quantity	Q	000000 - 999999: 6 digits remaining in issue

6.2. Command reference table

Variable name	Get_Value	Put_Value
@DEVICE_VERSION	DC2(12H)+PC	

Execute method names	Command name
PrintFormat	ESC(1BH)+YR,ESC(1BH)+/D
PrintBarcodeEAN13	ESC(1BH)+B
PrintQRCode	ESC(1BH)+2D30
PrintQRCodeJoint	ESC(1BH)+2D30
PrintRfidUhf	ESC(1BH)+IP0
PrintRfidUhfForFormat	ESC(1BH)+IP0,ESC(1BH)+YR, ESC(1BH)+/D
WriteRfidPassLock	ESC(1BH)+IP0
WriteRfidPermLock	ESC(1BH)+IP0
Enq	ENQ(05H)
GetStatus	DC2(12H)+PG
Raw	-