libNPrint.so.2.10

- Environment Construction Procedure
- SDK Guide

Overview

Regarding SDK Overview

API Reference

Regarding API Syntax

2025.9.22

Document disclaimer

- The description in this document is taken all possible measures to ensure the correctness,. However, if you find any suspicious points, errors, omissions, etc., please contact us.

- Descriptions are subject to change without prior notice.
  Please ask for up-to-date information.

- Reprinting, copying, duplicating, or falsifying part or all of the contents of this document without permission is strictly prohibited.

- Please note that we do not take any responsibility for the effects caused by the results of use.

- We assume no responsibility whatsoever for any damages resulting from improper use, the use without understanding of descriptions in this document or the repair and the change by the third party.

Regarding signs

Following signs are used in this guide. Please understand the meaning of each signage before using the product.

Caution	It indicates the precautions to be observed fully when using the product. Disregard of the signs and wrong usage may cause product failure and malfunction.
reference	It indicates supplementary descriptions and relevant matters.

Usage restrictions

In the event that this product is used with devices that require high reliability and safety for function and accuracy, etc. directly on conveyance including aircraft, train, ship or vehicle; etc., disaster prevention and security devices, etc., users should use products after considering the safety design of the whole system by applying fail-safe or redundancy design in order to maintain reliability and safety.
This product is not designed to use with devices that require extremely high reliability and safety including aerospace mechanism, devices for trunk communication, nuclear control device or medical services. Users should ascertain and evaluate suitability of these products for those applications.

Table of Contents

・Overview
- System configuration when using SDK ・・・・・・・・・・・・・・・・・・・・ 5

・Environmental construction procedure
- Environment construction, Regarding serial connection, Regarding sample code compiling ・・・・・・・・・・・・・・・・・・・・・・ 6
- Regarding the characters code setting, Regarding the serial connection, Precautions when using multiple printers ・・・・・・・・・・・・・・・・ 7
- Regarding the log output ・・・・・・・・・・・・・・・・・・・・・・・・・ 8

・ API Reference
- Category of APIs ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・11
- NCallback ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・15
- NSetCallback ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・16
- NEnumPrinters ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・17
- NDeletePrinter ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・19
- NRenamePrinter ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・20
- NGetPrinterInf ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・21
- NGetPrinterFromID ・・・・・・・・・・・・・・・・・・・・・・・・・・・22
- NOpenPrinter ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・23
- NOpenPrinterR ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・24
- NAutoOpen ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・25
- NOpenResult ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・26
- NClosePrinter ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・27
- NClosePrinters ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・28
- NPrint・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・29
- NDPrint ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・31
- NImagePrint ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・32
- NImagePrintF ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・36
- NGetStaus ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・37
- NGetInformation ・・・・・・・・・・・・・・・・・・・・・・・・・・・・38
- Extended Information ・・・・・・・・・・・・・・・・・・・・・・・・・40
- NResetPrinter ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・41
- NStartDoc ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・42
- NEndDoc ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・43
- NCancelDoc ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・44
- NEnumDoc ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・45
- NDeleteDoc ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・46
- NBarcode ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・47
- NFirmwareDL ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・48
- NFirmwareResult ・・・・・・・・・・・・・・・・・・・・・・・・・・・49

Table of Contents

- NInitializeNetwork ・・・・・・・・・・・・・・・・・・・・・・・・・・・・50
- NScanPrinters ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・51
- NTCPPortLock ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・52
- NBufferClear ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・53
- NBlockSendSetting ・・・・・・・・・・・・・・・・・・・・・・・・・・・54
- NSetUSBProtocol ・・・・・・・・・・・・・・・・・・・・・・・・・・・・55
- NFDeleteJob ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・56
- NSetSerialPort ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・57

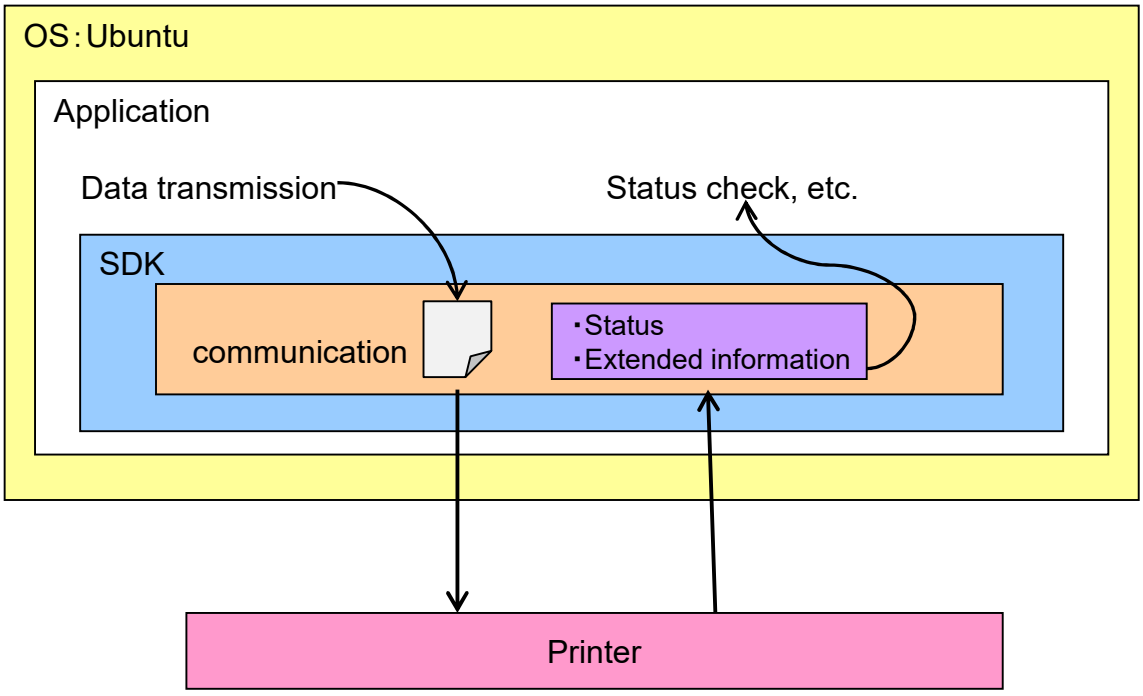
・ Error code table ・・・・・・・・・・・・・・・・・・・・・・・・・・・・58

Overview

Printing and printer monitoring function can be integrated into the application in which customers need to have printing function by using SDK.

The file provided as SDK is delivered with the file name “libNPrint.so.2.10”.

System configuration when using SDK.



- Development Language

C programming language, C++

- Ubuntu compatible version

16.04（32ビット）, 18.04, 20.04, 22.04, 24.04

- Interface

USB, COM

Caution

Although we have done operation tests, please verify and evaluate it sufficiently before using this SDK since errors during communications may occur under customers’ environment.

Various settings / Precautions

Environment construction

- Folder for storing SO library
  - ・Please save it in ”/usr/lib/」OR「/usr/local/lib/” directly.
    * Please update dependency relationship information of the shared library by the command “ldconfig”.

Regarding sample code compiling

- Please execute the following command to compile the sample code.

```
$gcc libNPrint_sample.c /usr/local/lib/libNPrint.so.2 -o test `pkg-config --cflags --libs
opencv` -lm –lpthread
```

Or

In case of OpenCV3

```
$g++ libNPrint_sample.cpp /usr/local/lib/libNPrint.so.2 -o test `pkg-config --cflags --libs
opencv` -lm –lpthread
```

Or

In case of OpenCV3

```
$g++ libNPrint_sample.cpp /usr/local/lib/libNPrint.so.2 -o test `pkg-config --cflags --libs
opencv4` -lm –lpthread
```

reference Although warning may occur depending on the compiler environment and version, there is no problem in operation.

Please specify /usr/local/lib/libNPrint.so.2 for the compiling target as well.
In addition, please link them by specifying –libs when using OpenCV.

There are three kinds of sample codes for OpenCV2 (C), OpenCV3 (C++) and OpenCV4 (C++). Please use one of them depending on the environment.

In case of using the sample code for OpenCV3（C++)

```
#include <opencv2/core.hpp>            // for OpenCV3
#include <opencv2/highgui.hpp>         // for OpenCV3
#include <opencv2/imgcodecs.hpp>   // for OpenCV3
 // comment out
 //#include <opencv4/opencv2/core.hpp>          // to OpenCV4
 //#include <opencv4/opencv2/highgui.hpp>       // to OpenCV4
 //#include <opencv4/opencv2/imgcodecs.hpp>  // to OpenCV4
```

In case of using the sample code for OpenCV3（C++)

```
 // comment out
 //#include <opencv2/core.hpp>           // for OpenCV3
 //#include <opencv2/highgui.hpp>        // for OpenCV3
 //#include <opencv2/imgcodecs.hpp>// for OpenCV3
#include <opencv4/opencv2/core.hpp>          // to OpenCV4
#include <opencv4/opencv2/highgui.hpp>       // to OpenCV4
#include <opencv4/opencv2/imgcodecs.hpp>  // to OpenCV4
```

Various settings / Precautions

Regarding the characters code setting

- When using this SDK, characters conversion code can be specified by allocating the file called **NCharSetInf** under ”**usr/npi/**”. By the default setting, there is no file.

- In order to handle characters strings by Shift JIS code, please make **NCharSentInf** file and write “Shift JIS” in it. The file will be read in when NOpenPrinter function is executed.

Regarding the serial connection

- Although the power supply status is regularly monitored by using the signal from the CTS signal cable, power supply monitoring by the signal from the CTS signal cable will not be done and , it will become online status all the time by locating the file named “**COM**” under “**usr/npi/**”
- Power supply monitoring by the signal from the CTS signal cable cannot be done depending on the types of the serial cable. In that case, it will remain online status setting by turning the power OFF instead of becoming offline status.

Precautions when using multiple printers

- An offline or an automatic recovery may occur temporarily due to a Linux system restriction at the recovery after an offline when it occurs at one of the multiple printers connected by the USB connection.

Regarding authorization permission

Each authorization will be required for opening the printer port or reading files and writing to files with this SDK.
Please assign appropriate authorization to each user who use the program.

[For connecting the printer with USB interface]
/dev/usb/lp* (“*” is a numeric number.) Reading and writing authorization

[For connecting the printer with COM interface]
/dev/ttyS* (“*” is a numeric numer.) Reading and writing authorization

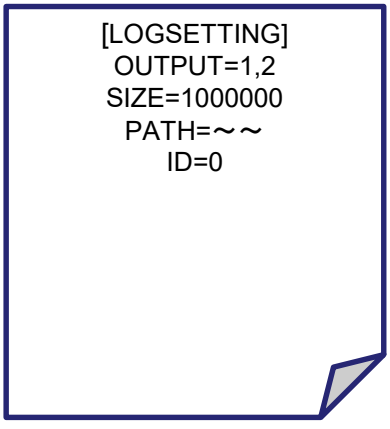
[Common]

/usr/npi/	Reading and writing, execution authorization
/usr/npi/spool	Reading and writing, execution authorization
/usr/npi/temp	Reading and writing, execution authorization
”.inf” file under /usr/npi/	Reading and writing authorization
Log output folder	Reading and writing, execution authorization

(In case of outputting the log according to the steps describe on the next page.)

Regarding log output (output setting)

This SDK outputs the file named **SDKLog.txt** under the specified folder for log files.
For log output, it is necessary to create a log setting file (/usr/npi/NLogInf.inf).
(No log is output if the file does not exist)



Write the configuration file in the following format.

[LOGSETTING]	← fixed
OUTPUT=1,2	← Specifying the log types to be output separated by commas. (In this case 1,2 : only ERROR and WARN are output.)
SIZE=1000000	← Specifying the maximum file size of log writing.
PATH= ～～	← Specifying the log writing folder
ID=0	← Setting the output settings for the process ID and thread ID. (0 : without output, 1 : with output)

Regarding log types (OUTPUT).

1 ERROR	:	Error
2 WARN	:	Warning
3 FUNC	:	Function call (except checking status and extended information)
4 IN	:	Port receiving data
5 OUT	:	Port transmitting data
6 CHK	:	When checking status and extended information
7 PST	:	When printer status value is changing, when receiving extended information

* Log will be output according to specified types.

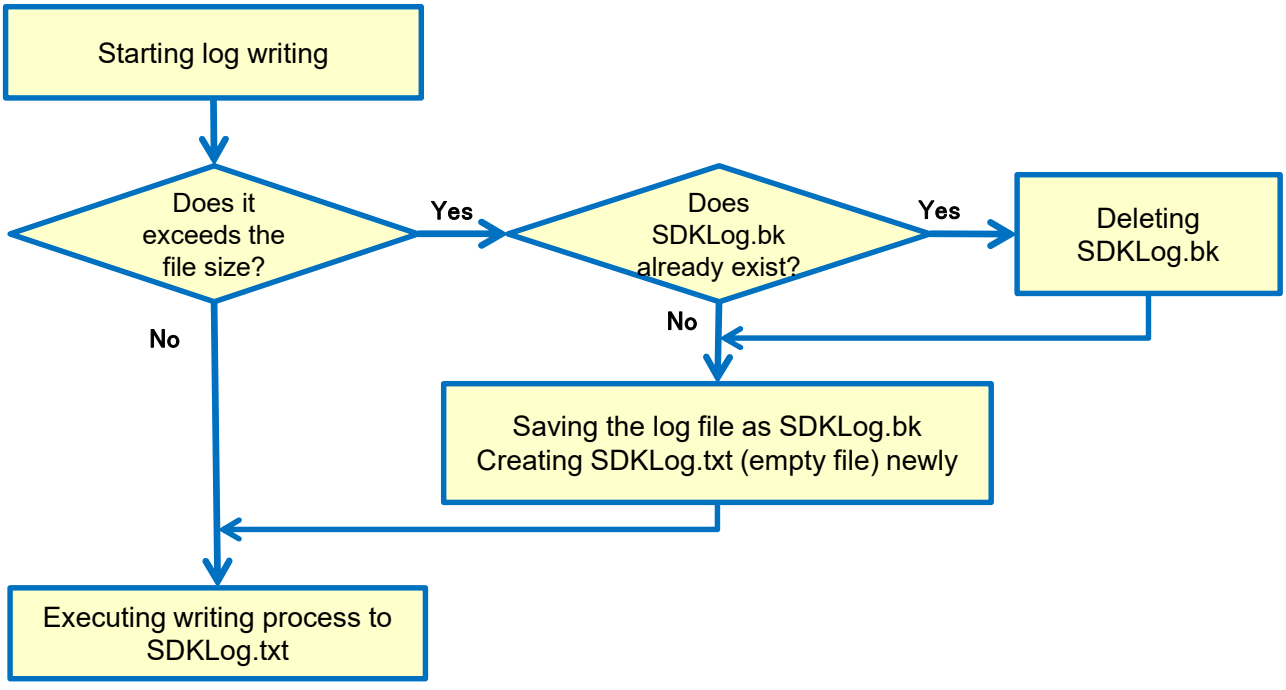
Regarding log output (notes)

*1 Editing the file does not take effect immediately.
The program should be shut and reactivated in order to make it effective.
*2 DO NOT put spaces (both half-width or full-width) before or after the “=".
*3 If nothing is set for OUTPUT, no log is output.
*4 When an invalid log type value (numeric value other than 1 to 7 or character string) is specified, the value is disregarded.
If an invalid log size value (negative size value, character string) is specified, the default value 1MB will be used.
*5 With respect to log types 4 (IN) and 5 (OUT), SDK operation may be slow when the amount of data transferred is large because logging takes a long time.
*6 If you specify a large file size, you may not be able to open the log file. Please change the file size according to your device.

Regarding log output (output file)

If the number of bytes specified in the configuration file is exceeded, the file is saved as a past log (SDKLog.bk) and a new empty file is created.

If SDKLog.bk already exists, it will be deleted (only one log of the past log is stored).



•【Log output sample】
•2017/08/01 14:00:01.000 [FNC] xxxx.
•2017/08/01 14:00:01.100 [WRN] xxxx.
•2017/08/01 14:00:01.200 [ERR] xxxx.
•2017/08/01 14:00:01.350 [FNC] xxxx.
•2017/08/01 14:00:01.614 [I N] xxxx.
•2017/08/01 14:00:02.100 [OUT] xxxx.
•2017/08/01 14:00:03.040 [I N] xxxx.
•2017/08/01 14:00:03.500 [CHK] xxxx.
•2017/08/01 14:00:04.010 [PSK] xxxx.

The date and time will be output in the format of YYYY/MM/DD hh:mm:ss,milliseconds.
Log types are indicated in [ ].
1 ERR : Error
2 WRN : Warning
3 FNC : Function call
4 I N : Port receiving data
5 OUT : Port transmitting data
6 CHK : When checking status and extended information
7 PSK :When printer status value is changing, when receiving extended information

If ID = 1 is set in the log setting file,
information in the form of <process ID, thread ID> will be added after the date and time.

Various settings / Precautions

Regarding USB transmission setting

This SDK reads the file for data transmission setting with USB connection and will make the setting when the printer is connected (NOpenPrinter).
File name : **/usr/npi/NWriteCustom.inf**

Caution

* Please use the transmission setting file（NWiteCustom.inf）as it is without editing it ordinarily. When data will be transmitted at high speed and the printer cannot catch up with the speed, Please edit this file according to the description below and control the transmitted size, etc.
Please verify and evaluate it sufficiently since errors during communications may occur under customers' environment.

If the file does not exist, a file will be created newly and the default setting will be set.
When the file exists, values input to the file will be set.

```
[USB]
SIZE=0
INTERVAL=0
```

How to describe the file is as follows.

[USB] ← fixed
SIZE=0 ← Data size which can be transmitted at one time
INTERVAL=0 ← Interval time between two transmissions

Regarding "SIZE"
- When "0" (default setting) is set, all the data will be transmitted at one time.
- From "1" byte to "15,000" bytes can be set.
- If more than "15,000" is specified, "15,000" will be set.

Regarding "INTERVAL"
- Even if "0" (default setting) is set, there will still be a waiting time for a few milliseconds.
- There is not the upper limit for the set value.

Regarding receiving buffer remaining size setting
- When "data transmission with receiving buffer remaining size monitoring" (default setting) is specified, the receiving buffer remaining size will be highly prioritized than the size specified by this setting file for the data transmission.
- When "data transmission with receiving buffer remaining size monitoring" is not specified, the size has to be specified by "NsetUSBProtocol" function. (Please refer to the other page.)

Category of APIs (1/3)

Following APIs are available.

Application	API	Description
Designate callback function	NSetCallback	Setting Callback function.
Obtaining a list of printer name	NEnumPrinters	Obtaining a list of printer names.
Deleting printer name	NDeletePrinter	Deleting printer name(s) . (Specified printer's name or in batch.)
Changing Printer name	NRenamePrinter	Changing a printer name.
Obtaining Printer Information.	NGetPrinterInf	Obtaining information using the printer name as a key.
Obtaining printer name	NGetPrinterFromID	Obtaining a printer name from various IDs. ***Not supported.**
Open printer	NOpenPrinter	Designating printer name and open it. (It can be opened automatically when it is offline.)
Open printer	NOpenPrinterR	Designating printer name and open it. (It can be opened automatically when it is offline.)
Automatic printer setting	NAutoOpen	Setting automatic printer open function. ***Not supported.**
Checking opening	NOpenResult	Obtaining the result of NOpenPrinter function. ***Not supported.**
Close printer	NClosePrinter	Closing the printer which has been opened.
Close all printers.	NClosePrinters	Closing all printers which have been opened.
Sending command and data	NPrint	Converting the specified hexadecimal character string data and sending it to the printer.
Sending command and data	NDPrint	Sending the specified data to the printer.
Sending image	NImagePrint	Sending the specified image data by the specified image command to the printer.

Category of APIs (2/3)

Application	API	Description
Sending image	NImagePrintF	Sending specified file (bmp) by the specified image command to the printer.
Obtaining status	NGetStatus	Returning the status obtained from the specified printer
Obtaining extended information	NGetInformation	Returning the information of target type ID which has been obtained by the specified printer.
Reset printer	NResetPrinter	Resetting printers with USB interface. ***Not supported.**
Document control	NStartDoc	Starting storing documents.
Document control	NEndDoc	Ending storing documents and send them to the printer.
Document control	NCancelDoc	Canceling storing documents.
Document control	NEnumDoc	Obtaining the transmission waiting documents list. ***Not supported.**
Document control	NDeleteDoc	Deleting the transmission waiting documents.. ***Not supported.**
Generate barcode	NBarcode	Generating barcode image. ***Not supported.**
F/W upgrade	NFirmwareDL	Updating firmware by the target firmware file.
F/W upgrade result check	NFirmwareResult	Checking a result of function NFirmwareDL. ***Not supported.**
UDP Thread Operation	NInitializeNetwork	Starting / stopping UDP receiving thread. ***Not supported.**
Printer Enumeration	NScanPrinters	Detecting TCP/UDP printers. ***Not supported.**
TCP Communication Lock	NTCPPortLock	Locking / unlocking other printers' TCP communication. ***Not supported.**
TCP Buffer clear	NBufferClear	Clearing the printing buffer of TCP/UDP printers. ***Not supported.**

Category of APIs (3/3)

Application	API	Description
TCP Batch Transmission	NBlockSendSetting	Starting / stopping TCP/UDP printers' batch storing transmission. ***Not supported.**
Remaining receiving buffer setting	NSetUSBProtocol	Setting remaining receiving buffer checks during USB transmission. ***Not supported.**
Document control	NDeleteJob	Deleting documents and checking the number of the remaining documents.
Serial communication settingシ	NSetSerialPort	Specifying the settings required for the serial communication.

Interface name	NCallback		
Argument name	IN/OUT	Type	Description
i_prt i_type i_value1 i_value2	I I I I	char* int int int	Printer name Callback Type Execution result 1 Execution result 2
Return value	Void		
Nothing			
Processing description			

- This is the functional form of the callback notification.
  Declaring the function with the same return values and arguments as in this declaration and specifying the function pointer as an argument to the NSetCallback function.

  It can also be used as the third argument to the NOpenPrinter function to specify a callback at the opening.
  (Since NOpenPrinter function is performed asynchronously, the result is obtained as a callback.)

  The conditions for callback notifications are as follows.

Callback conditions	i_prt	i_type	i_value1	i_value2
When the status value is changed. (Including the first receiving.)	Printer name	1	Old status value	New status value
When receiving the extended information.	Printer name	2	Extended information ID	0x00
When the transmitting data cue becomes empty.	Printer name	3	0x00	0x00
NOpenPrinter result notification	Printer name	5	Execution result	0x00

Function name	NSetCallback		
Argument name	IN/OUT	Type	Description
i_callback	I	NCallback	Callback function pointer
Return value	Void		
Nothing			
Processing description			

- Setting the function of NCallback.
  Please execute this function before doing callback notification.
  The interface settings can also be made by using the “NOpenPrinter” function.

  Execute this function before execute the callback notification,
  In addition, you can perform the interface setting in function “NOpenPrinter”.

Caution

Unless null is specified in this function, the callback notification continues to the class once it is set.
Please note that if the class instance you set has already been deleted, an exception will be raised from the SDK.

Please implement the callback function on the application side in order to handle the contents of the callback notification (the argument of NCallback).

Function name		NEnumPrinters	
Argument name	IN/OUT	Type	Description
o_printers o_size	O O	char* unsigned int*	Printer name (csv : Enumerating them in a Comma Separated Value form) Printer name byte size of o_printers (Null can be specified.)
Return value	Int		
·Error(negative value), Normal finish(0) * Please refer to the error code table.			

Process contents

- It creates a printer information management file (NPrinterInf.inf) under “usr/npi/” and stores a list of connection available printer names in the o_printers argument.

  When the empty file named “**NEnumSw**” under “/usr/npi”,the information format t will be “Printer name, Type (please refer to * below), Printer information”.

  e.g.) PRT001,1,/dev/usb/lp0
  PRT002,1,/dev/usb/lp2

  * 0 : COM port (/dev/ttyS*)
  1 : USB (/dev/usb/lp*)

  When the file “ENumSw” does not exist, only printer names are enumerated.

  e.g.) PRT001, PRT002, PRT003

- The name of “PRTxxx” (xxx is one of numeric numbers from 001 to 999) will be allocated to the printer name to be generated after detecting the USB port.
  The “xxx” for allocating the printer names can be generated up to 999 and more than that will be generating errors.
  Operations of deleting unnecessary data manually or renaming the printer names by using “NRenamePrinter” described later, etc. will be required.

Since opening the printer (NOpenPrinter) cannot be done when the printer information management file is not created yet, please be sure to call this function at the start of the usage.
This function is required to be called when printers are added.

Function name	NEnumPrinters
Process contents	

Caution

* Once a printer name is generated, it will not be deleted even if USB is plugged off, etc., If you want to delete it , please execute NDeletePrinter function or edit the “NPrinterInf” file manually (by deleting the row including the target printer).

Function name	NDeletePrinter		
Argument name	IN/OUT	Type	Description
i_prt	I	char*	Printer name to be deleted
Return value	Int		
·Error(negative value), Normal end(0) * Please refer to the error code table.			
Processing description			

- This is used to delete the printer name in the printer information management file.
  Specifying the printer name to be deleted in the argument i_prt.
  By specifying a blank character, the printer information management file itself is deleted (deleting all printer names).

* Please DO NOT use this function if it is open status after NOpenPrinter function is executed so as not to disable closing operation by NClosePrinter function.

Function name	NRenamePrinter		
Argument name	IN/OUT	Type	Description
i_beforeprt i_afterprt	I I	char* char*	Printer name to be changed Printer name after the change
Return value	Int		
·Error(negative value), Normal end(0) * Please refer to the error code table.			

Processing description

- This is used to rename the printer name in the Printer information management file. Although the printer name is created starting with “PRT” like “PRT001” by default, it can be changed to any printer name by using this function.
  The printer name is limited to 50 single-byte alphanumeric characters, and the following characters cannot be used.

  Characters cannot be used（including an space）: **[]¥/:?*”<>|’, .**

* Please DO NOT use this function if it is open status after NOpenPrinter function is executed so as not to disable closing operation by NClosePrinter function.

Function name	NGetPrinterInf		
Argument name	IN/OUT	Type	Description
i_prt	I	char*	Printer name
o_ports	O	char*	Printer information (csv : To enumerate them in a Comma Separated Value form)
o_size	O	unsigned long*	Size of o_ports bytes (Null can be specified.)
Return value	Int		
·Error (negative value), Normal end (0) * Please refer to the error code table			

Processing description

- Searching the printer information management file for the printer name specified by the argument and storing the following information in o _ ports.

  Connection types

  COM : 0

  USB : 1

  ex.) 0,/dev/ttyS0

  1,/dev/usb/lp1

The information obtained by o_ports can also be got by referring to the printer information management file (NPrinterInf.inf).

Caution

- Since the byte size obtained by “o-size” is the size excluding the last “NULL(¥0), it is required to secure the memory size for “o_ports” by “o_size (byte size + 1)”.

Function name	NGetPrinterFromID		
Argument name	IN/OUT	Type	Description
i_ID o_printer	I O	char* char*	ID Printer name（NULL can be specified）
Return value	Int		
･Always “-40” : not supported			
Processing description			

- “-40 (not supported)” is always returned as a returned value although this function is implemented in order to maintain compatibility with other SDK versions for other OSes.

Function name	NOpenPrinter		
Argument name	IN/OUT	Type	Description
i_prt	I	char*	Printer name to be opened
i_statusFlg	I	unsigned char	Status receiving thread activating flag 0 : Receiving the status when receiving the status obtaining request. 1 : Activating the thread and receiving the status. (Without automatic reconnection process at the receiving error.) 2 : Activating the thread and receiving the status.(With automatic reconnection process at the receiving error.)
i_callback	I	NCallback	Function pointer with implemented callback interface (Null can be specified.)
Return value	Int		
･Error (negative value), Normal end (0) * Refer to the error code table			

Processing description

- Executing the printer opening process.
  Specifying the printer name obtained by NEnumPrinters to the argument i_prt.
  Printer opening process can be performed even when the printer is offline.

  The thread will be activated and will receive the status at 50msec intervals by specifying “1” or “2” to the stats receiving thread.
  As a result, the status obtained by NGetStatus will be updated periodically.
  Please DO NOT specify “0” since it is not supported.

- By passing the pointer of the callback function pointer of the application to the third argument “i_callback”, this function enables users to obtain the result as a callback at the end of the opening process.
  (Obtaining the result as a callback is recommended since it is performed asynchronously.)

- When you do not use the callback, please omit the third argument (specifying NULL) .
  In this case, the result of success or failure cannot be obtained.

Function name	NOpenPrinterR		
Argument name	IN/OUT	Type	Description
i_prt	I	char*	Printer name to be opened
i_statusFlg	I	unsigned char	Status receiving thread activating flag 0 : Not supported 1 : Without automatic reconnection process at the receiving error. 2 : With automatic reconnection process at the receiving error.)
i_callback	I	NCallback	Function pointer with implemented callback interface (Null can be specified.)
Return value	Int		
·Error (negative value), Normal end (0) * Refer to the error code table			

Processing description

- Executing the printer opening process.
  The arguments and how to use it are same as “NOpenPrinter”.
  (Although this function exists for the compatibility with Ver,2,2,3, please use “NOpenPrinter ordinarily.)

Caution

- An offline or an automatic recovery may occur temporarily due to a Linux system restriction at the timing of next opening (normal end) when an error occurs by opening the USB printer at offline

Function name	NAutoOpen		
Argument name	IN/OUT	Type	Description
i_flg	I	int	Automatic printer opening flag 0 : Not executing automatic opening (manual opening) 1 : Automatic opening Others : Only obtaining stats (not doing setting)
Return value	Bool		

・Always FALSE : automatic open ineffective

Processing description

- “FALSE (automatic opening ineffective)” is always returned as a returned value although this function is implemented in order to maintain compatibility with other SDK versions for other OSes.

  Please do openings by executing NOpenPrinter.

Function name	NOpenResult		
Argument name	IN/OUT	Type	Description
i_prt	I	char*	Printer name
Return value	Int		
･Always “-40” : not supported			
Processing description			

- “-40 (not supported)” is always returned as a returned value although this function is implemented in order to maintain compatibility with other SDK versions for other OSes.

  Please use callback in order to know a result of NOpenPrinter asynchronously.

Function name	NClosePrinter		
Argument name	IN/OUT	Type	Description
i_prt	I	char*	Printer name
Return value	Int		
･Error (negative value), Normal end (0)　* Please refer to error code table.			
Processing description			

- This function executes the printer closing process.
  It stops the status receiving thread as well.

Function name	NClosePrinters		
Argument name	IN/OUT	Type	Description
—	—	—	—
Return value	Int		
·Error (negative value), Normal end (0)　* Please refer to error code table.			
Processing description			

- Performing the all printers closing process.

Caution

This function returns only “0” (success) as a returned value.
Even when it is already closed, it returns “0” (success). If you would like to get the error value, please use the NClosePrinter function and close printers one by one.

Function name	NPrint		
Argument name	IN/OUT	Type	Description
i_prt	I	char*	Printer name
i_dat	I	char*	Transmitting data (hexadecimal character string)
i_size	I	unsigned int	Number of output bytes
o_jobid	O	unsigned int*	Print job ID (Null can be specified)
Return value	Int		
･Error (negative value), Normal end (0) * Please refer to error code table.			

Processing description

- Transmitting the specified hexadecimal character string data to the printer.
  Data enclosed in “””” (double-quotation mark), “<>” (angle brackets) and “[ ]” (brackets) are transmitted after being transformed as follows.

  1. Character string in “”(double-quotation mark)
     => Converted as character string (”ABC“ => 0x41,0x42, 0x43)
  2. File name in <> (angle brackets) (including path)
     => Output file contents (binary data).
  3. File name in [ ] (brackets) (including path)
     => Output image for bit map file in raster block format.

  Please specify the same number of output bytes as the transmission data size.

Function execution example :

```
char data1[] = “4142434445464748490A”;
NPrint(“PRT001”, data1, 20, null);

char data2[] = “¥”ABCDEFGHI¥”0A”;
NPrint(“PRT001”, data2, 15, null);

char data3[] = “<./Raster(384-256).bin>0A”;
NPrint(“PRT001”, data3, 25, null);

char data4[] = “[./PrintSample.bmp]0A”;
NPrint(“PRT001”, data3, 21, null);
```

Function name	NPrint
Processing description	

reference

- Errors depending on the printer conditions never occurs by this function.
  Data to be printed are stored in transmitting buffer and the data in the buffer will be transmitted to the printer when the printer becomes ready for printing (transmitting).

Caution

- In order to output the data by shift JIS, please allocate **NCharSetInf** file under “use/npi” and write “Shift JIS” in it.
  The printer has to conform to Shift JIS system.
  Please read the Product Specifications of each printer and check the “Selecting Kanji code system” command and the memory switch setting. (It varies depending on the printers.)
- Although print data are stored in the transmitting buffer, data cannot be transmitted by executing this function repeatedly more than once.

  Failure case :

```
// 1B69 full cut command
NPrint(“PRT001”, “1B”, 2, NULL);
NPrint(“PRT001”, “69”, 2, NULL)
```

  Success case :

```
// 1B69 full cut command
NPrint(“PRT001”, “1B69”, 4, NULL);
```

* NCharSetInf file is applied only to NPrint function and it does not exist by the default setting.

Function name	NDPrint		
Argument name	IN/OUT	Type	Description
i_prt i_dat i_size o_jobid	I I I O	char* unsigned char* unsigned int unsigned int*	Printer name Transmitting data (hexadecimal character string) Number of output bytes Print job ID (Null can be specified)
Return value	Int		
·Error (negative value), Normal end (0)　* Please refer to the error code table.			

Processing description

- Transmitting the specified data to the printer (without conversion).
  Please specify the same number of output bytes as the transmission data size.

Function execution example :

```
unsigned char data[10] = { 0x41, 0x42, 0x43, 0x44, 0x45, 0x0A, 0x0A, 0x0A, 0x1B, 0x69 };
NDPrint("PRT001", data, 10, null);
```

reference

- Errors depending on the printer conditions never occurs by this function.
  Data to be printed are stored in transmitting buffer and the data in the buffer will be transmitted to the printer when the printer becomes ready for printing (transmitting).

Caution

- Although print data are stored in the transmitting buffer, data cannot be transmitted by executing this function repeatedly more than once.

Failure case :

```
// 1B69 full cut command
 unsigned char data1[0] = { 0x1B };
 unsigned char data2[0] = { 0x69 };
 NDPrint("PRT001", data1, 1, NULL);
 NDPrint("PRT001", data2, 1, NULL);
```

Success case :

```
// 1B69 full cut command
 unsigned char data[2] = { 0x1B, 0x69 };
 NDPrint("PRT001", data, 2, NULL);
```

Function name	NImagePrint		
Argument name	IN/OUT	Type	Description
i_prt	I	char*	Printer name
i_bufImage	I	unsigned char*	Data array of image (Please refer to the next page)
i_width	I	unsigned int	Width of image file (pixel)
i_height	I	unsigned int	Height of image file (pixel)
i_channels	I	unsigned int	Numbers of image file channels
i_step	I	unsigned int	Numbers of image file steps
i_putType	I	unsigned char	Output system 0x00 : Raster format by line 0x01 : Raster format by block 0x02 : Raster format by block with gray scale expression 0x10 : Bit image format
o_jobid	O	unsigned int*	Print job ID (Null can be specified.)
Return value	Int		
·Error (negative value), Normal end (0) * Please refer to the error code table.			
Processing description			

- Transmitting the specified device context to the printer.

Caution

- It takes longer time to output raster format gray scale expression by block in comparison with other format. Furthermore, some printers do not support this function.
- When the height or the width is specified beyond the image size, an error such as a program suspension may occur. Please specify it so that it will not exceed the image size in the customer's program.

Function name	NImagePrint
Processing description	

reference

- Errors depending on the printer conditions never occurs by this function.
  Data to be printed are stored in transmitting buffer and the data in the buffer will be transmitted to the printer when the printer becomes ready for printing (transmitting).

- How to maintain the compatibility with NImagePrint of Ver,1.4 or lower is described on the page 33.

How to specify NImagePrint parameters when using OpenCV

In case of using OpenCV2.x

```
IplImage* img = cvLoadImage("./Sample.bmp", CV_LOAD_IMAGE_ANYCOLOR |
                                            CV_LOAD_IMAGE_ANYDEPTH);

int channels            = img->nChannels;    // Number of channels
int step                = img->widthStep;    // Number of steps
int width               = img->width;        // Image width
int height              = img->height;       // Image height
unsigned char* bufImage = img->imageData;    // Data array of the image
```

In case of using OpenCV3.x or OpenCV4.x

```
cv::Mat matImg = cv::imread("./Sample.bmp");

int channels            = matImg.channels(); // Number of channels
int step                = matImg.step;       // Number of steps
int width               = matImg.cols;       // Image width
int height              = matImg.rows;       // Image height
unsigned char* bufImage = matImg.data;       // Data array of the image
```

Function execution example

```
NImagePrint("PRT001", bufImage, width, height, channels, step, 0x01, NULL);
```

Caution

If an error message (Corrupt JPEG data, etc.) is displayed or width / height becomes 0 when reading an image file with OpenCV, the image file may be damaged or invalid, and printing may not be possible.

When not using OpenCV, please prepare information regarding data arrangement, width and height, etc. on the customer side.

Regarding the wrapper function of NImagePrint

This descriptions is for using the arguments for NImagePrint function of Ver.1.4 or lower without taking any measures. This wrapper function is described in the sample application “libNprint sample.c” in the same folder (the folder named “Sample” in “Document”.

In case of using OpenCV2.x

```
IplImage* img = cvLoadImage(“./Sample.bmp”, CV_LOAD_IMAGE_ANYCOLOR |
                                                CV_LOAD_IMAGE_ANYDEPTH);
NImagePrintWrap(“PRT001”, img, img->width, img->height, 0x01, NULL);
```

<Wrapper function>

```
int NImagePrintWrap(char* i_prt, IplImage* i_bmp, unsigned int i_width,
unsigned int  i_height, unsigned char i_putType, unsigned int* o_jobid)
{
  int ret = 0;
  int channels = i_bmp->nChannels;                  // Channel count of Image
  int step = i_bmp->widthStep;                      // Step count of Image
  unsigned char* bufImage = i_bmp->imageData;       // Data array of Image

  ret = NImagePrint(i_prt, bufImage, i_width, i_height, channels, step, i_putType, o_jobid);
  return ret;
}
```

In case of using OpenCV3.x or OpenCV4.x

```
cv::Mat img = cv::imread(“./Sample.bmp”);
NImagePrintWrap(“PRT001”, img, img.cols, img.rows, 0x01, NULL);
```

<Wrapper function>

```
int NImagePrintWrap(char* i_prt, cv::Mat i_bmp, unsigned int i_width, unsigned int i_height,
unsigned char i_putType, unsigned int* o_jobid)
{
  int ret = 0;
  int channels = i_bmp.channels();                  // Channel count of Image
  int step = i_bmp.step;                            // Step count of Image
  unsigned char* bufImage = i_bmp.data;                             // Data array of Image

  ret = NImagePrint(i_prt, bufImage, i_width, i_height, channels, step, i_putType, o_jobid);
  return ret;
}
```

* Only function name of NImagePrint calling in Ver1.4 or lower needs to be changed into "NImangePrintWrap".

Function name	NImagePrintF		
Argument name	IN/OUT	Type	Description
i_prt	I	char*	Printer name
i_bmp	I	char*	Image file name (including path)
i_putType	I	unsigned char	Output system 0x00 : Raster format by line 0x01 : Raster format by block 0x02 : Raster format by block with gray scale expression 0x10 : Bit image format
o_jobid	O	unsigned int*	Print job ID (Null can be specified.)
Return value	Int		
・Error (negative value), Normal end (0)　* Please refer to the error code table.			
Processing description			

- Transmitting the specified BMP file to the printer.
  Since the structure became the system with which OpenCV is not used inside the SDK after ver2.0, only bmp can be used.

Caution

- It takes longer time to output raster format gray scale expression by block in comparison with other format. Furthermore, some printers do not support this function.

reference

- Errors depending on the printer conditions never occurs by this function.
  Data to be printed are stored in transmitting buffer and the data in the buffer will be transmitted to the printer when the printer becomes ready for printing (transmitting).

Function name	NGetStatus		
Argument name	IN/OUT	Type	Description
i_prt o_status	I O	char* unsigned long*	Printer name Status
Return value	Int		
·Error (negative value), Normal end (0)　* Please refer to the error code table.			
Processing description			

- Returning the status obtained from the specified printer.

  * Please refer to the target printer's Product Specifications for values to be returned.

Caution

- Although the returned value of this function execution after the power OFF during COM (serial) communication will not be an error value (negative value), statuses cannot be obtained.

Function name	NGetInformation		
Argument name	IN/OUT	Type	Description
i_prt	I	char*	Printer name
i_id	I	unsigned char	Type ID
o_dat	O	void*	Extended information storing area
o_time	O	unsigned long*	Update flag (Obtaining the current time of the system in milliseconds.) (Null can be specified.)

Return value	Int

·Error (negative value), Normal end (0) * Please refer to the error code table.

Processing description

- Obtaining information stored in target type of ID of extended information.
  * It is necessary to send a request of desired information from a higher-level application to the printer in advance.
    (Some do not require requests for extended status, transmission completion, print completion information, etc.)
  * Please refer to the part of the command《ESC v》in target printer's Product Specifications for values to be returned.

Extended Information

Type1 : 4 bytes (fixed) : update flag (4bytes)	<Extended status> 1st byte : 7~0, 2nd byte : 15~8, 3rd byte : 23~16, 4th byte : 31~24
Type 2 :32 bytes (delimiter) : update flag (4bytes)	<Model name>
Type 3 : 8 bytes (fixed) : update flag (4 bytes)	<F/W version>
Type 4 : 8 bytes (fixed) : update flag (4 bytes)	<Boot version>
Type 5 : 4 bytes (fixed) : update flag (4 bytes)	<Reserved>
Type 6 : 4 bytes (fixed) : update flag (4 bytes)	<Number of dot lines energizing head>
Type 7 : 4 bytes (fixed) : update flag (4 bytes)	<Number of fed dot lines>
Type 8 : 4 bytes (fixed) : update flag (4 bytes)	<Number of cuts>
Type 9 :16 bytes (fixed) : update flag (4 bytes)	<User maintenance counter: Number of dot lines energizing head, Number of fed dot lines, Number of cuts, Reserved>
Type10 :16 bytes (fixed) : update flag (4 bytes)	<Reserved>
Type11 :64 bytes (delimiter) : update flag (4 bytes)	
Type12 :32 bytes (delimiter) : update flag (4 bytes)	
Type13 :32 bytes (fixed) : update flag (4 bytes)	<NV registration status>
Type14 :32 bytes (fixed) : update flag (4 bytes)	<Reserved>
Type15 :16 bytes (fixed) : update flag (4 bytes)	
Type16 :16 bytes (fixed) : update flag (4 bytes)	
Type17 :16 bytes (fixed) : update flag (4 bytes)	
Type18 :16 bytes (fixed) : update flag (4 bytes)	
Type19 : 8 bytes (fixed) : update flag (4 bytes)	<End of print notification: Arbitrary ID and end status are described at the time of end command processing by specifying the print start/end command.>
Type20 : 8 bytes (fixed) : update flag (4 bytes)	<Reserved>
Type21 : 8 bytes (fixed) : update flag (4 bytes)	
Type22 : 8 bytes (fixed) : update flag (4 bytes)	
Type23 : 8 bytes (fixed) : update flag (4 bytes)	
Type24 : 4 bytes (fixed) : update flag (4 bytes)	<Reserved>
Type25 : 4 bytes (fixed) : update flag (4 bytes)	<Transfer completion notification: transferred job ID will be described.>
Type26 : 4 bytes (fixed) : update flag (4 bytes)	<Reserved>
Type27 : 4 bytes (fixed) : update flag (4 bytes)	<Reserved>
Type28 : 2 bytes (fixed) : update flag (4 bytes)	<F/W checksum>
Type29 : 2 bytes (fixed) : update flag (4 bytes)	
Type30 : 2 bytes (fixed) : update flag (4 bytes)	
Type31 : 2 bytes (fixed) : update flag (4 bytes)	<Communication status information : USB: 0x0000 fixed COM: 1st byte CTS 2nd byte DSR * Final signal status obtaining time stamp is set to update flag.>

* Except for types 25 and 31, the obtained information is not valid for information that is not implemented by the printer.
* The contents described here may not be available for all printers.

```
Type 32: 12 bytes (fixed): Update flag (4 bytes) <General information 8 bytes>
                                                    Byte 0 = Sub ID,Bytes
                                                         1-3 = Fixed to 00h,Bytes
                                                         4-11 = General information
Type 33: 8 bytes (fixed): Update flag (4 bytes) <General information 4 bytes>
                                                    Byte 0 = Sub ID,Bytes
                                                         1-3 = Fixed to 00h,Bytes
                                                         4-7 = General information
Type 34: 4 bytes (fixed): Update flag (4 bytes) <General information 2 bytes>
                                                    Byte 0 = Sub ID,Byte
                                                         1 = Fixed to 00h,Bytes
                                                         2-3 = General information
Type 35: 8 bytes (fixed): Update flag (4 bytes)
Type 36: 8 bytes (fixed): Update flag (4 bytes)
Type 37: 4 bytes (fixed): Update flag (4 bytes)
Type 38: 4 bytes (fixed) : Update flag (4 bytes)
Type 39: 2 bytes (fixed) : Update flag (4 bytes)
Type 40: 2 bytes (fixed) : Update flag (4 bytes))
Type 41: 65536 bytes (fixed) : Update flag (4 bytes) <Log Information>
                                                    Cutter Retry Count (4 bytes fixed)
                                                    Printer Error Log (2048 bytes fixed)
                                                                                    (128 attempts)
                                                    Receive Log (15360 bytes fixed) (960 attempts)
                                                    Transmit Log (5120 bytes fixed) (320 attempts)

Type 42: Variable length : Update flag (4 bytes) <Get Printer Settings Bulk>


*The general information for types 32-34 varies depending on the printer. Please contact us for
details.
```

Function name	NResetPrinter		
Argument name	IN/OUT	Type	Description
i_prt	I	char*	Printer name
Return value	int		

・Always “-40” : not supported

Processing description

- This function became unsupported because of the impact which the fact that libusb had not been used brought. “-40 (not supported)” is always returned as a returned value.

Function name	NStartDoc		
Argument name	IN/OUT	Type	Description
i_prt o_jobid	I O	char* unsigned int*	Printer name Print job ID
Return value	Int		
·Error (negative value), Normal end (0)　* Please refer to the error code table.			

Processing description

- Starting the document.
  After nstartDoc is published, NPrint data, NDPrint data, NImagePrint data and NImagePrintF data are stored in a memory.

  The saved data can be output to the printer by calling NEndDoc.
  The accumulated data will be cleared by calling NCancelDoc.

  One document can be handled per printer.

reference

The number of job IDs which can be obtained is increased by one from one every time this function is executed and is returned to one when this number exceeds 255.
In addition, it also is returned to one when the printer closing process by NClosePrinter, etc. is executed.

Function name	NEndDoc		
Argument name	IN/OUT	Type	Description
i_prt	I	char*	Printer name
Return value	Int		
·Error (negative value), Normal end (0)　* Please refer to error code table.			
Processing description			

- Ending the document.
  The data stored after calling NStartDoc is output to the printer.
  If the stored data does not exist, processing is not performed.

Function name	NCancelDoc		
Argument name	IN/OUT	Type	Description
i_prt	I	char*	Printer name
Return value	Int		
·Error (negative value), Normal end (0)　* Please refer to the error code table.			
Processing description			

- Canceling the document.
  If the stored data does not exist, processing is not performed.

Function name	NEnumDoc		
Argument name	IN/OUT	Type	Description
i_prt olist	I O	char* char*	— —
Return value	Int		
･Always “-40” : not supported			
Processing description			

- “-40 (not supported)” is always returned as a returned value although this function is implemented in order to maintain compatibility with other SDK versions for other OSes.

Function name	NDeleteDoc		
Argument name	IN/OUT	Type	Description
i_prt	I	char*	—
Return value	Int		
·Always “-40” : not supported			
Processing description			

- “-40 (not supported)” is always returned as a returned value although this function is implemented in order to maintain compatibility with other SDK versions for other OSes.

Function name	NBarcode		
Argument name	IN/OUT	Type	Description
i_prt	I	char*	Printer name
i_fontName	I	char*	Barcode font name
o_bufImage	IO	unsigned char*	Array of created barcode
i_dat	I	unsigned char*	Barcode data
i_size	I	unsigned int	Data size
Return value	Int		

・Always “-40” : not supported

Processing description

- This function became unsupported from Ver.2.3. “-40” (not supported) is always returned as a returned value.

Function name	NFirmwareDL		
Argument name	IN/OUT	Type	Description
i_prt	I	char*	Printer name
i_file	I	char*	Fwf file name (Null can be specified.)
i_errflg	I	unsigned char	Error check 0x00 : Invalid (Forced transmission) 0x01 : Valid
io_chksum	IO	unsigned int*	Fwf file checksum (Null can be specified. It is required for firmware check.)
o_jobid	O	unsigned int*	Print job ID (Null can be specified.)

Return value	Int

・Error (negative value), Normal end or checksum match (0)
* Please refer to error code table.

Processing description

(Firmware download)

- Transmitting the specified fwf file to the printer.

(Firmware check)
If the fwf file is specified as Null, the checksum is obtained from the printer and compared with the checksum specified by the argument.

When the error check is specified by “0x00”, it is forcibly transmitted even if there is an error in the printer status.

Caution

- It is not available while the document is started.
  Please call NEndDoc, NCancelDoc in order to end the document.

Function name	NFirmwareResult		
Argument name	IN/OUT	Type	Description
i_prt	I	char*	Printer name
Return value	Int		

・Always “-40” : not supported

Processing description

- “-40 (not supported)” is always returned as a returned value although this function is implemented in order to maintain compatibility with other SDK versions for other OSes.

  Please use callback in order to know a result of NFirmwareDL asynchronously.

Function name	NInitializeNetwork		
Argument name	IN/OUT	Type	Description
i_flg	I	int	UDP receiving thread activating / stopping flag. 0: Stop 1: Activate
Return value	Int		

・Always “-40” : not supported

Processing Description

- “-40 (not supported)” is always returned as a returned value although this function is implemented in order to maintain compatibility with other SDK versions for other OSes.

Function name	NScanPrinters		
Argument name	IN/OUT	Type	Description
i_waitmsec	I	unsigned long	Number of milliseconds to wait for enumeration response / data creation
Return value	Int		

・Always “-40” : not supported

Processing Description

- “-40 (not supported)” is always returned as a returned value although this function is implemented in order to maintain compatibility with other SDK versions for other OSes.

Function name	NTCPPortLock		
Argument name	IN/OUT	Type	Description
i_prt i_type	I I	char* unsigned char	Target printer name Setting type
Return value	Int		
･Always “-40” : not supported			
Processing Description			

- “-40 (not supported)” is always returned as a returned value although this function is implemented in order to maintain compatibility with other SDK versions for other OSes.

Function name	NBufferClear		
Argument name	IN/OUT	Type	Description
i_prt	I	char*	Target printer name
Return value	Int		

・Always “-40” : not supported

Processing Description

- “-40 (not supported)” is always returned as a returned value although this function is implemented in order to maintain compatibility with other SDK versions for other OSes.

Function name	NBlockSendSetting		
Argument name	IN/OUT	Type	Description
i_prt i_type	I I	char* unsigned char	Target printer name Setting type
Return value	Int		
･Always “-40” : not supported			
Processing Description			

- “-40 (not supported)” is always returned as a returned value although this function is implemented in order to maintain compatibility with other SDK versions for other OSes.

Function name	NSetUSBProtocol		
Argument name	IN/OUT	Type	Description
i_prt i_type	I I	char* unsigned int	Printer name Checking flag of receiving buffer remaining size 0：Monitoring receiving buffer remaining size and transmitting (default). 1：Transmitting without checking receiving buffer remaining size 2：Returning the current flag value for checking the receiving buffer remaining size. (It will not set the flag value.)
Return value	Int		

・Error (negative value), Current (after setting) receiving butter size checking flags (0,1)
* Please refer to error code table.

Processing description	**Specifying whether checking USB receiving buffer remaining size at data transmission.**

・Specifying whether USB receiving buffer remaining size at data transmission or not. Default value is “0” and data will be transmitted with the receiving buffer remaining size checked**. It is required to be specify “1”** for the printer which does not respond the receiving buffer remaining size.

・Set result will be effective only during USB interface communication. It will be disregarded when COM interface is used.

・Please use this function afte4r opening the printer by using “NOpenPrinter” function.

・Since the setting made by this function will be deleted when the printer is closed by using “NClosePrinter” function, this function has to be executed again when changing the value after reopening the printer.

Function name	NDeleteJob		
Argument name	IN/OUT	Type	Description
i_prt i_cmp	I I	char* unsigned char	Printer name Document flag 0 : Deleting the first document file in waiting document files. 1 : Deleting the first document file (even while being transmitted). 2 : Checking the number of waiting document files after deleting.
Return value	Int		

・Checking the number of document files during waiting after deletion or error (negative value)
* Please refer to the error code table.

Processing description

- Deleting the document files or checking the number of the document files.

Caution

- Printing data enclosed by the StartDoc function and the EndDoc function becomes one document.
- Regarding document files
  In this SDK, printing data executed by the NPrint/NDPrint/NImagePrint/NImagePrintF functions, etc. is stored once as a file (document) under “usr/npi/spool” and is processed sequentially..

Function name	NSetSerialPort		
Argument name	IN/OUT	Type	Description
i_prt	I	char*	Printer name
i_baudRate	I	int	Communication speed (bps)
i_csize	I	int	Character size (data bit) 7,8 (bit)
i_parity	I	char	Parity 'e' : even number, 'o' : odd number, 'n' : not specified
i_stopbit	I	int	Stop bit 0 : stop bit (1 bit) 1 : stop bit (1.5 bits) 2 : stop bit (2 bits)
i_flow	I	char	Flow control 'n' : not specified, 'x' : XON/XOFF 'p' : hardware flow control (RTS / CTS terminals)
Return value	Int		

・Error (negative value), Normal end or checksum match (0)
* Please refer to error code table.

Processing description

- Specifying the settings required for the serial communication.
- By specifying the settings with this function before connecting to the printer, it will be connected in accordance with set conditions at connecting to the printer (NOpenPrinter).
- If the connected printer is disconnected (NClosePrinter, NClosePrinters), the settings will be those set before connecting.

Caution

- Serial communication settings are required to be done by the printer side. Please refer to the switch settings part of the Product Specifications.
- DTR / RTS terminals for flow controlling are not supported under Linux.
- For establishing the serial connection (NOpenPrinter) after the setting, permission has to be given unless there as an authority for reading and writing the serial connection terminal devices (e.g. /dev/ttyS0).

Error code table（1/2）

SUCCESS	0	Normal end
ERR_HANDLE	-1	Handle error
ERR_PRTOPEN	-2	Printer open error
ERR_OFFLINE	-5	Offline
ERR_UNEXPLAINED_OFFLINE	-6	Offline (Output error <data transmitting error>) * This error status will not be released until the data transmission succeeds. (It will not be changed to ERR_OFFLINE until then.) * All data receiving errors are ERR_OFFLINE (-5).
ERR_FILEOPEN	-10	File open error
ERR_PRTOUTPUT	-13	Printer output error
ERR_FILEWRITE	-15	File writing error
ERR_FILEREAD	-16	File reading error
ERR_MAKEDIR	-18	Directory generating error
ERR_PRTALREADYOPEN	-21	Printer has been already opened.
ERR_NOHANDLE	-22	Printer information does not exist.
ERR_LACKRESOURCE	-31	Lack of resources
ERR_LOADFROMFILE	-50	Failed to load the image file
ERR_RESETPRINTER	-60	Resetting failure
ERR_STARTDOC	-70	StartDoc function error
ERR_DOCNOTSTARTED	-71	Not in the document start state
ERR_ALREADYSTARTDOC	-72	Already in the document start state
ERR_ENDDOC	-73	EndDoc function error
ERR_FWFFILE	-80	fwt file error
ERR_FWF_CHECKSUM	-81	The checksum entered in the argument does not match the checksum obtained from the printer.
ERR_FWCHK_TIMEOUT	-83	Time-out of checking the firmware checksum
ERR_FWCHK_FOUNDERROR	-84	Detecting an error in checking the status during firmware download

Error code table（2/2）

ERR_ARGUMENT	-90	Argument error
ERR_ARGUMENT_01	-91	1st argument error
ERR_ARGUMENT_02	-92	2nd argument error
ERR_ARGUMENT_03	-93	3rd argument error
ERR_ARGUMENT_04	-94	4th argument error
ERR_ARGUMENT_05	-95	5th argument error
ERR_ARGUMENT_06	-96	6th argument error
ERR_ARGUMENT_07	-97	7th argument error
ERR_ARGUMENT_08	-98	8th argument error
ERR_ARGUMENT_09	-99	9th argument error
ERR_PRTINFO_GET	-136	Printer information cannot be obtained.
ERR_PRTINFO_ENTRYMAX	-138	Process cannot be performed because the printer information has reached the limit.
ERR_DEVICE_NOTSUPPORT	-150	Connection type not supported
ERR_CMDLINE	-170	Command line error (When the command can not be executed.)
ERR_SEARCH	-180	Searching error (When the searching is executed inside SDK.)