

- **Environment Construction Procedure**
- **SDK Guide (Android)**

Overview

Descriptions regarding SDK Overview

API Reference

Descriptions regarding API Syntax

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.
- Bluetooth and the Bluetooth logo are registered trademarks of Bluetooth SIG. Inc.
- MS-DOS®, Microsoft®, Win32®, Windows®, Windows Vista®, Visual Studio®, Visual Basic®, Visual C++®, Visual C# Visual C++®, Visual C#® are trademarks or registered trademarks of Microsoft Corporation in the United States and other countries.
- All other product and company names are trademarks or registered trademarks of their respective owners.

About Signage

Following signage 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
- Environment Construction Procedure
 - [Environment construction](#) 6
 - [Setting of developer options](#) 6
 - [Regarding the file allocation path](#) 7
 - [Regarding the file to be set](#) 7
 - [Regarding barcode setting](#) 7
 - [Regarding Bluetooth connection](#) 10
 - [Regarding character code setting](#) 10
 - [Regarding log output](#) 11
- API Reference
 - [How to use functions](#) 14
 - [Category of API](#) 15
 - [NPrinterLib\(Class\)](#) 17
 - [NCarryLibrary\(Class\)](#) 18
 - [NCallback\(Interface\)](#) 19
 - [NSetCallback](#) 22
 - [NEnumPrinters](#) 23
 - [NDeletePrinter](#) 24
 - [NRenamePrinter](#) 25
 - [NGetPrinterInf](#) 26
 - [NGetPrinterFromID](#) 27
 - [NOpenPrinter](#) 28
 - [NOpenResult](#) 30
 - [NClosePrinter](#) 33
 - [NClosePrinters](#) 34
 - [NPrint](#) 35
 - [NDPrint](#) 37
 - [NImagePrint](#) 38
 - [NImagePrintF](#) 39
 - [NGetStatus](#) 40
 - [NGetInformation](#) 41

Table of Contents

- API Reference

— NResetPrinter	44
— NStartDoc	45
— NEndDoc	46
— NCancelDoc	47
— NEnumDoc	48
— NDeleteDoc	49
— NBarcode	50
— NFirmwareDL	52
— NFirmwareResult	53
— NSetUSBProtocol	54
— NSetSendRetry	55
— NInitializeNetwork	56
— NScanPrinters	57
— NTCPPortLock	59
— NBufferClear	60
— NBlockSendSetting	61
▪ Error code table	62
▪ List of returned value for each function	65

Overview

SDK enables application to have functions such as printing and monitoring printers.

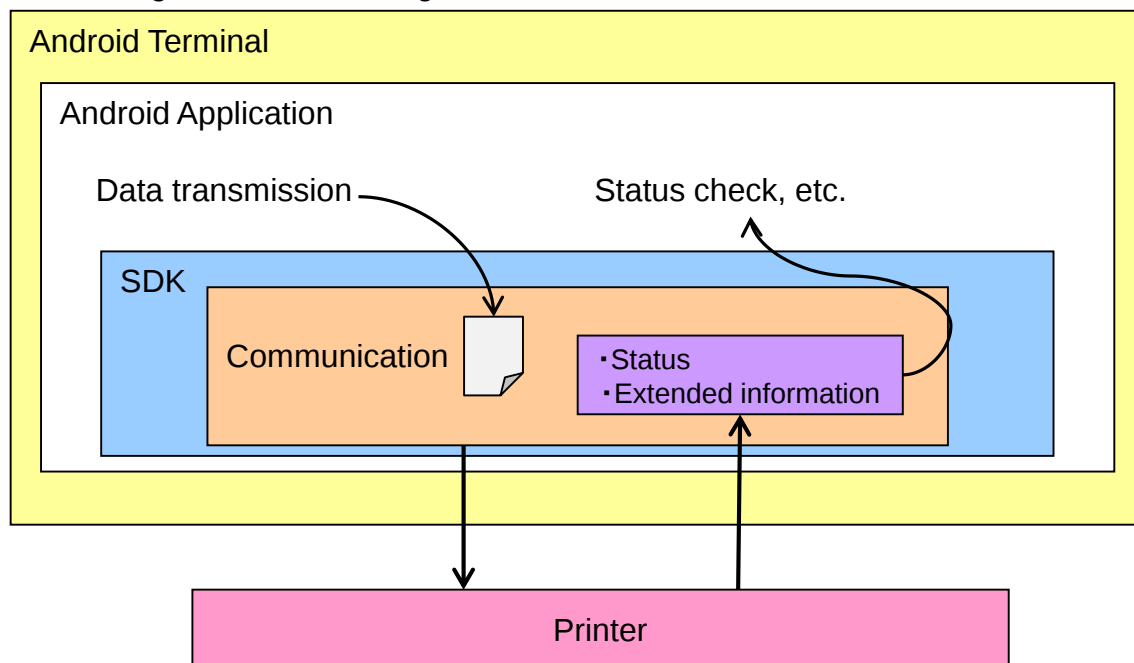
The files distributed as SDK are provided as jar files.

File name: NPrinterLib.jar

core.jar (for generating barcodes)

javase.jar (for generating barcodes)

System configuration when using SDK.



- **Development language**

java, Kotlin

- **Applicable Android versions**

From 7 to 14

- **Interface**

USB, Bluetooth, TCP/UDP(WLAN)

Caution

Although we have made checks for operations, communication may not be established properly depending on the terminal to be used.
Please check and evaluate it sufficiently on the user's side.

Environment Construction Procedure

Environment construction

- Please store the three provided “jar” files in the “libs” folder of Android project under development. Accordingly classes such as NPrinterLib, etc. can be used.
- * It can be read only by the above way ordinarily. If it cannot be read, please conduct the following steps.
 - Project Structure → Dependencies → Modules → Declared Dependencies, pushing the button → Jar Dependencies → Inputting “libs” for the folder name at 1st step. → Clicking “OK”
- Please add the following items to AndroidManifest.xml when configuring application. (Please refer to AndroidManifest.xml of sample program details.)

(1) It is needed for reading and writing the file allocation path (allocating the setting file, etc.).

<!-- Use file read/write permissions below API Level 33 -->.

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

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

(2) It is needed for Needs for Bluetooth communication.

<Please use the old “Permission” with the level lower than “!-- API Level 30”.>

```
<uses-permission android:name="android.permission.BLUETOOTH"
  android:maxSdkVersion="30" ></uses-permission>
```

```
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN"
  android:maxSdkVersion="30" ></uses-permission>
```

< Please use the following new “Permission” with the level “!-- API Level 30” or higher.>

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

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

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

(3) It is needed for TCP/UDP communication.

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

(4) It is needed for sharing the data of NPrinterLib class.

(Passing the NPrinterLib class from the main screen to other screens.)

```
<application
  android:name="npi.sdk.NCarryLibrary" >
```

Environment Construction Procedure

(5) When using Android 12 or later, exported attribute has to be declared explicitly in the activity tag.

```
<application
  android:name="npi.sdk.NCarryLibrary>
  <activity
    android:name="npi.sdksample.NPISDK"
    android:exported="true">
  </activity>
</application>
```

Developer options setting

- Please select "OFF" for "Don't keep activities" in Developer Options in Android settings. (Selection will not be needed if the developer option is not displayed.)

Regarding the file allocation path

If the location of the configuration file is an application-specific folder [only the first argument of the NPrinterLib constructor (android.content.Context class) is specified, or the third argument (configuration file creation/reference path) is blank ("") or null], The path varies depending on the Android device.

It can be obtained by executing **getSettingPath function of NPrinterLib class** from the program. The same process is built in the "Information/Setting" -> "File path check" button of the sample program.

Be sure to obtain the file path using **getSettingPath function of NPrinterLib class**. The path may differ depending on the user, even for the same terminal.

Hereafter in this guide, we will use the notation "file placement path" to describe the placement path of the setting file specified in the constructor of NPrinterLib.

Notes regarding the file location

Creation, reading, and writing operations of folders and files generated by applications from Android 13 and later are restricted.

The restrictions are as follows

- Folders and files can be created, read, and written in the Documents (/storage/emulated/0/Documents) and Download (/storage/emulated/0/Download) folders in the internal storage.
- Folders and files cannot be created, read, or written in folders that require Android administrator privileges (e.g. /storage/emulated/0/Android/data).
- Folders and files can be created by the program in paths other than the above two or in application-specific folders, but cannot be read or written by the file manager.

※To manipulate configuration files, use the File Manager or the SDK configuration file editing functions (NLogSetProperty, NSetCharSetInf, NSetWriteCustom, NBarcode1DSetting).

Notes regarding the file authority

Creation, reading, and writing operations of folders and files generated by applications from Android 13 and later are restricted.

The restrictions are as follows

- The various setting inf files (NPrinterInf.inf, NWriteCustom.inf, NLogInf.inf, NBarcodeInf.inf, NCharSetInf.inf) and log files issued by the SDK are **authorized by the application** that created them.

It is not possible to read or write files from another application that is not the file creator.

※If NOpenPrinter fails to read the NPrinterInf.inf file, error code -137 (printer information file is invalid) will occur.

Only the application from which the file was created can read/write the file.

- The various setting inf files (NPrinterInf.inf, NWriteCustom.inf, NLogInf.inf, NBarcodeInf.inf, NCharSetInf.inf) and log files issued by the SDK are copied and duplicated by the file manager.

inf file cannot be read or written from the original application that created the various setting inf files.

Environment Construction Procedure

Regarding the file to be set

In the sample program, there is a button to create various setting file folders in the Android terminal (file allocation path).

Each operation of the sample program is designed to operate with the file created by this button.

The file can be generated by [Information/Setting]→[Default setting file create] button in sample program. After creating the file, please close the application and start it up again. (Because the configuration file is loaded at the start of the application.)

Setting folder will be generated automatically with the name with [npi] under file application path at the application start. Please delete the folder manually because it will not be deleted even if the application is uninstalled.

Regarding Authority

Since file operation Bluetooth connection are used with this SDK, the permitted authority is required for using this SDK with the application which this SDK is set in. Please assign the permitted authority in the authority assigning dialog displayed when the sample program is activated. (Refer to [Android] How to use NPI SDK.pdf)

* In Android 12, the authorization is "Detect, connect, and locate nearby devices" (Bluetooth) and "Access photos, media, and files on the device" (read/write files).

For Android 13 and 14, only "Detect, connect, and locate nearby devices" (Bluetooth) is required, and the required permissions vary depending on the version of Android being used.

Service" application (android.app.Service) that incorporates this SDK must be authorized in advance.

(Android device "Settings" → "Apps" → "NPI SDK" → "Permissions" → "Select Permissions" → "Allow")

(Refer to [Android] How to use NPI SDK.pdf)

Regarding barcode setting

1. Barcode setting file allocation

Making the setting file is required in order to use "NBarcode" function with this SDK.

To create or edit a setting file, use NBarcode1DGetData and NBarcode1DSetting functions manually or by using NBarcode1DGetData and NBarcode1DSetting functions.

Please install application for reading and writing file such as file manager.

(If it has been already installed, it is unnecessary to install it.)

Barcode file name: NBarcodeInf.

This file needs to be allocated under [inter storage]/np. Please make the npi folder if the folder does not exist.

The way to write a file is described on the following 2 pages.

Environment Construction Procedure

Regarding barcode setting

2. How to write a barcode setting file (1D barcodes).

Please set Barcode1 to Barcode10 with “ini” file format under the following specifications.

Please note that font name and each item distinguish large and small letters, so be careful to input them correctly without mistakes.

[Barcode1] ← **Barcode1 to Barcode10 can be specified. (Barcode font name)**

TYPE=0
WIDTH=2
HEIGHT=100
HRI=1
HEXMODE=0
ROTATE=0
STARTBIT=0
STOPBIT=0
FILENAME=barcode01.bmp

The following default value is used when there is not NBarcodeInf file.

TYPE : 0 STARTBIT : 0
WIDTH : 2 STOPBIT : 0
HEIGHT : 100 FILENAME : barcodeXX.bmp
HRI : 0 (“XX” is between 01 to 10.)
HEXMODE : 0
ROTATE : 0

[Barcode2]
TYPE=1
WIDTH=3
HEIGHT=150
HRI=1
HEXMODE=0
ROTATE=0
STARTBIT=0
STOPBIT=0
FILENAME=barcode02.bmp

...
...

[Barcode10]

TYPE

0:UPC-A
1:UPC-E
2: EAN13
3: EAN8
4: CODE39
5: ITF
6: CODABAR
7: CODE128
8: CODE93

WIDTH

Setting magnification of data width between 2 and 4.

HEIGHT

Setting data height by dots.

HRI

0: None
1: Top (Font A)
2: Bottom (Font A)
3: Top&Bottom (Font A)
4: Top (Font B)
5: Bottom (Font B)
6: Top&Bottom (Font B)

HEXMODE

0: Input data normally (0,1,2,...)
1: Input data hexadecimally (0x30,0x31,...)

ROTATE

0: No rotation
1: 90° rotation
2: 180° rotation
3: 270° rotation

STARTBIT,STOPBIT (Only CODABAR setting is valid)

0: A
1: B
2: C
3: D

FILENAME

Assign image file name of barcode generated on [Internal storage]/npi/image.
(When blank is selected, file will not be generated)

Environment Construction Procedure

Regarding barcode setting

3. How to write a barcode setting file. (2D barcodes).

Please set 2D-Barcode1 to 2D-Barcode5 with "ini" file format under the following specifications.

Please note that font name and each item distinguish large and small letters, so be careful to input them correctly without mistakes.

[2D-Barcode1] ← 2D-Barcode1 to 2D-Barcode5 can be specified. (Barcode font name)

TYPE=0

SIZE=100

HEIGHT=

HEXMODE=0

ERROR=L

ROTATE=0

FILENAME=2d-barcode01.bmp

[2D-Barcode2]

TYPE=1

SIZE=200

HEIGHT=8

HEXMODE=0

ERROR=

ROTATE=0

FILENAME=2d-barcode02.bmp

...

...

[2D-Barcode5]

The following default value is used when there is not NBarcodeInf file.

TYPE : 0

SIZE : 2

HEIGHT : 1

HEXMODE : 0

ERROR : L

ROTATE : 0

FILENAME : 2d-barcodeXX.bmp ("XX" is between 01 to 05.)

TYPE

0: QRCode Model2

1: PDF417

SIZE

< QRCode Model2 >

Setting magnification of data size between 2 and 8.

< PDF417 >

Setting magnification of data size between 2 and 4.

HEIGHT

(Only PDF417 setting is available.)

Setting height ratio between 1 and 6.

HEIGHT = height ratio × WIDTH

HEXMODE

0: Input data normally (0,1,2,...)

1: Input data hexadecimally (0x30,0x31,...)

ERROR (Error level)

< QRCode Model2 >

Setting L, M, Q or H.

< PDF417 >

Auto(non-input), setting any between 0 and 8

ROTATE

0: No rotation

1: 90° Rotation

2: 180° Rotation

3: 270° Rotation

FILENAME

Setting image file name of barcode generated by [internal storage]/npi/image.

(When blank is selected, file will not be generated.)

Regarding barcode setting

4. Description of how to set barcode file (regarding HEXMODE) .

It is described for the case of inputting the barcode data (the 8th argument of the NBarcode function) as an byte array .

Normal input (by specifying HEXMODE = 0)

If "0x31, 0x32, 0x33, 0x34 ("1234")" is set, the data will be transmitted as it is.

Hexadecimal number input (by specifying HEXMODE = 1)

If "0x31, 0x32, 0x33, 0x34 ("1234")" is set, "... 0x12, 0x34" will be transmitted.

5. Description of how to set barcode file (regarding FILENAME) .

An image file will be created under [file allocation path]/npi/image at the timing of Bitmap class generation by specifying the file name in FILENAME and executing the NBarcode function.

Image creation path cannot be changed.

Please specify it by the way of excluding path (file name only).

When the blank is specified and executed, file will not be generated (only Bitmap class is generated).

Regarding Bluetooth connection

Pairing the Android terminal and the printer is required before Bluetooth connection in this SDK. this SDK. Please refer to manuals for each Android terminal regarding pairing and set it.

Regarding character code setting

In this SDK, character string converting code inside the SDK can be specified by allocating **NCharSetInf.inf** under [file allocation path]/npi/. This file does not exist with the default setting.

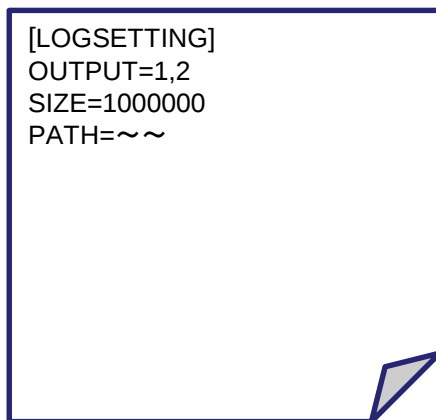
In case of handling character string of Shift JIS code, please generate "**NCharSetInf.inf**" file and write "Shift JIS" in it. The file will be read in at the timing of generating "NPrinterLib".

If an Android terminal supports a character code, describing other codes to the file can be done.

Furthermore, loaded character codes are stored in variable "**mStrCharSet**" in NPrinterLid class.

Regarding log output (Output setting)

This SDK outputs the log file named SDKLog.txt under the [file allocation path]/npi/log. For log output, it is necessary to create a log setting file the ([file allocation path]/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=/storage/emulated/0/Android/data/[application package name]/files/npi/log
← Specifying the log writing folder.
(in this case, [file allocation path]/npi/log)

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)

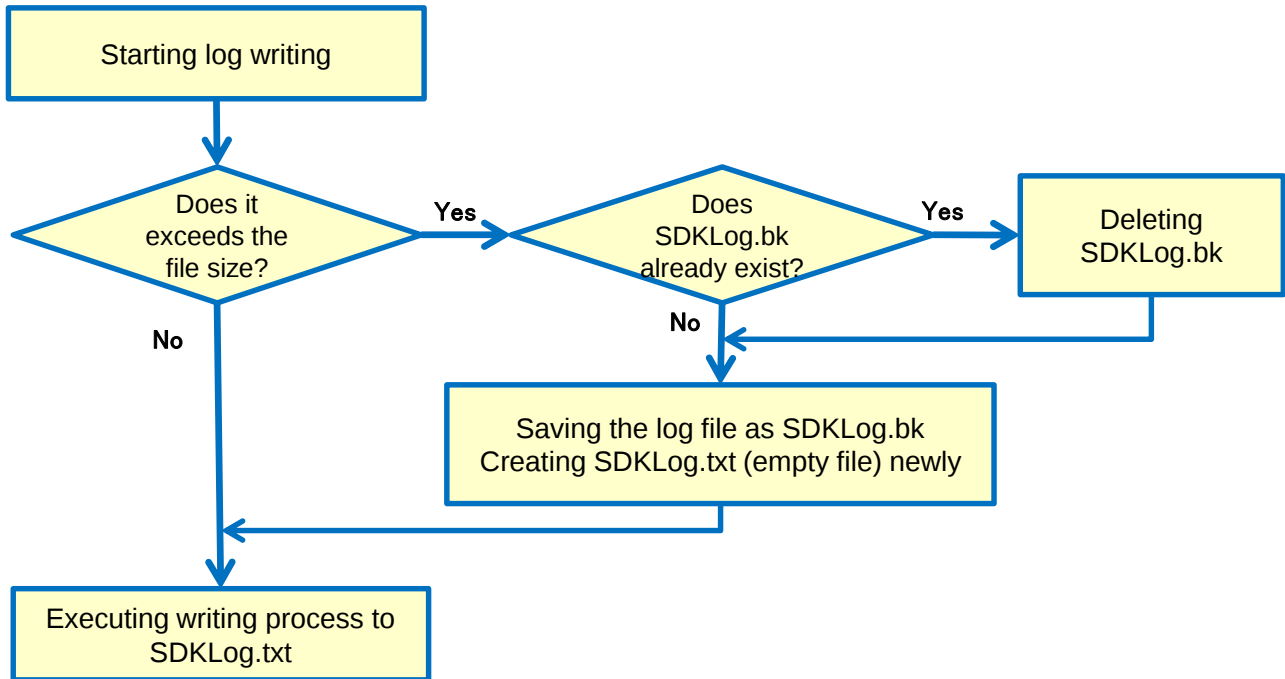
- *1 Editing the file does not take effect immediately.
The reflected timing is when the NPrinterLib class is generated.
- *2 DO NOT put spaces (both half-width or full-width) before or after the "=".
- *3 If nothing is set for OUTPUT, no log is output.
- *4 When an invalid log type value (numeric value 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.
- *7 Although the path to the log output location can be changed, reading or writing something can be done only inside the folder which is unique to the application since security has been reinforced. Logs cannot be created and handled with the output destination outside the application unique folder.

Environment Construction Procedure

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.

Regarding USB transmission setting

In this SDK, setting will be done by reading the file related to the data transmission under USB connection for when connecting the printer (“NOpenPrinter”).

File name : **[file location path]/]/npi/NWriteCustom.inf**

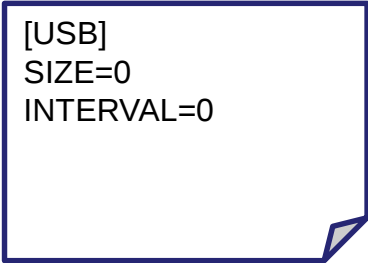
Caution

In regular cases, please use the transmission setting file (NWriteCustom.inf) as it is without editing it.

Only when the printer cannot catch up the transmission speed which is transmitted from the Android terminal with high speed, Please edit this file before use in order to control the transmitting size, etc.

Since communication may not be established properly depending on the terminal to be used. Please check and evaluate it sufficiently on the user's side.

When this file does not exist, it will be newly made and default values will be set.
When it exists, input values will be set to it.



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

How to set values in this file is as follows.

[USB] ← Fixed
SIZE=0 ← Data size (bytes) which can be transmitted at one time.
INTERVAL=0 ← Interval (msec) between transmission

Regarding the size

When “0” (default value) is set, all the data will be transmitted at one time.
Values from 1 to 15,000 (bytes) can be specified.

Regarding the size

Even when “0” (default value) is set. waiting time will be approx. 50msec.
There is no upper limit for the setting value.

Regarding receiving buffer remaining size setting

When “transmission with receiving buffer remaining size monitoring” (default) is set, the size specified by this file will be less prioritized than the receiving buffer remaining size for data transmission.

In order to make “transmission with receiving buffer remaining size monitoring” (default) unset, it is required to set it by using the function “NSetUSBProtocol”. Please refer to the page **XXX**.

How to use functions

- Please generate NPrinterLib class and execute the process by calling each function.
Please use NCarryLibrary class when sharing NPrinterLib by multiple activities (application sharing).
- When OUT parameter is specified by SDK, the following exclusive classes will be used (except for array and objects).
Normal types (int, boolean, etc.) will be used when IN parameter is specified.
Each class is provided with setValue(value) and getValue() functions and it can set values and obtain values.

(Package name: npj.sdk.data)

NBoolean	(boolean)
NByte	(byte)
NDouble	(double)
NFloat	(float)
NInt	(int)
NLong	(long)
NShort	(short)
NString	(String)

*Please refer to the sample program for details

Caution

- Regarding sequential printing and batch printing
If one of NPrint, NDPrint, NImagePrint, NImagePrintF functions is executed solely, data will be transmitted to the printer each time. Batch data transmission to the printer can be executed by using NStartDoc or NEndDoc function.
Please use sequential printing or batch printing according to the purpose of use.
- Regarding automatic open function
Automatic open function was got abolished at version 3.0.0.0. (NAutoOpen function has been deleted.)
Please execute the opening process by NOpenPrinter function before executing printing and obtaining the status.
- Please design and develop your applications after checking statuses and commands with the Product Specifications and the Command Reference of the printer model to be used.

Category of API (1/2)

Following APIs are available.

Application	A P I	Description
Setting Callback	NSetCallback	Setting Callback functions.
Obtaining printer information	NEnumPrinters	Obtaining a list of printer names.
Deleting printer name	NDeletePrinter	Deleting printer name(s) . (Specified a printer's name or in batch processing.)
Changing printer name	NRenamePrinter	Changing a printer name.
Obtaining printer information.	NGetPrinterInf	Obtaining information using the printer name as a key.
Obtaining printer name from ID	NGetPrinterFromID	Obtaining a printer name from various IDs.
Opening printer	NOpenPrinter	Specifying the printer name and opening it.
Closing printer	NClosePrinter	Closing the printer which has already been opened.
Closing all printers.	NClosePrinters	Closing all printers which have already been opened.
Transmitting the command and the data	NPrint	Converting the specified hexadecimal character string data and transmitting it to the printer.
Transmitting the command and the data	NDPrint	Sending the specified data to the printer.
Image output	NImagePrint	Transmitting the specified device context as a raster image or a bit image.
Image output	NImagePrintF	Transmitting the specified file (bmp/jpg/png) as a raster image or a bit image.
Obtaining status	NGetStatus	Returning the status obtained from the specified printer.
Obtaining extended information	NGetInformation	Obtaining the information in which the target ID is stored.
Resetting the printer	NResetPrinter	Resetting the printer connected with USB or WLAN interface.

Category of API (2/2)

Following APIs are available.

Application	A P I	Description
Document control	NStartDoc	Starting the document.
Document control	NEndDoc	Ending the document.
Document control	NCancelDoc	Canceling the document.
Document control	NEnumDoc	Returning the list of the document numbers which is in transmission waiting status.
Document control	NDeleteDoc	Deleting the documents which are waiting for being transmitted.
Generating the barcode	NBarcode	Generating the barcode image.
Firmware upgrade	NFirmwareDL	Updating the firmware by the target fwf file.
Setting the receiving butter remaining size check	NSetUSBProtocol	Setting the receiving buffer remaining size check for transmitting the data with USB interface.
Transmission retry setting	NSetSendRetry	Setting whether to retry the data transmission or not when failing.
UDP receiving thread control	NInitializeNetwork	Starting / stopping the UDP receiving thread.
Requesting printer enumeration	NScanPrinters	Transmitting requests for the information simultaneously to the multiple printers connected with TCP/UDP interface.
TCP communication port lock	NTCPPortLock	Locking / unlocking the TCP communication port (only for TCP / UDP interface).
Buffer clearing	NBufferClear	Clearing the receiving buffer. (only for TCP/UDP interface)
Setting batch transmission	NBlockSendSetting	Turning on and off the function which enables data storing and transmitting them (only for TCP/UDP interface).
Checking opening	NOpenResult	Obtaining the result of the function "NOpenPrinter". *Not recommended function.
Checking the result of firmware updating	NFirmwareResult	Checking the result of the function "NFirmwareDL". *Not recommended function.

Class name	NPrinterLib		
Constructor argument name	IN/OUT	Type	Description
i_context	I	Context	android.content.Context class
Return value			
Processing contents	This is the main class being used by this SDK.		

Executing processes by calling each function after generating this class.
Log setting file and barcode setting file will be read at generating this class.
(This class will be required to be regenerated again when the set file is changed.)

The context file being set by the argument can be specified by passing
[Activity class name].this from the application which has been made (please refer to the sample program).

Context class designated by argument is assigned by receiving.

[Activity class name].this from generated application. (Please refer to sample program)

e.g.) sample program

```
NPrinterLib objLib = new NPrinterLib(NPISDK.this);
```

Obtaining the version information (It can be obtained by String.)

```
objLib.SDKVERSION [Version / YYYY.MM.DD (release date)]
```

e.g.)

```
SDKVERSION = "1.1.0.0 / 2014.02.14";
```

Caution

Please DO NOT generate multiple NPrinterLib classes in one application because it will not be possible to get the printer status correctly in TCP/UDP and it will be impossible to send it.

Multiple applications which use this SDK cannot be activated from an Android terminal

Class name	NCarryLibrary			
Constructor argument name	IN/OUT	Type	Description	
Return value				
Processing contents	Common class for sharing NPrinterLib			

Please use NCarryLibrary class in order to pass NPrinterLib class to another activity.
Please use it after setting application to AndroidManifest.xml.

e.g.) Excerpted from sample program

* Setting NPrinterLib as follows.

```
NPrinterLib objLib = new NPrinterLib(NPISDK.this);
```

```
NCarryLibrary objCarry = (NCarryLibrary) this.getApplication();
```

```
objCarry.setLibrary(objLib);    // setting
```

* Obtaining the class in another activity as follows.

```
NCarryLibrary objCarry = (NCarryLibrary) this.getApplication();
```

```
NPrinterLib objLib = objCarry.getLibrary();    // obtaining
```

Caution

Please generate NPrinterLib class once per an application and implement it so as to be shared the class by each activity. If classes are generated for each activity separately, the setting or the information regarding connection cannot be taken over.

When changing the authority for allocation, etc. during Android setting after the transition to the different "Activity", the application will be reactivated. Since "NPrinterLib" class generated by this class becomes "null", "NPrinterLib" class has to be regenerated after returning to the original "Activity".

When the customer shares the class among activities by other ways, this class does not have to be used.

Interface name	NCallback
Processing description	This interface is used for callback notification.

This interface was newly added at Ver.2.0.0.0 and configurationally changed at Ver.3.0.0.0.

For functions that can obtain results asynchronously and for functions that require notification, the execution results (error codes) are reported using callbacks.

This is not a class but an interface.

Interface will be implemented by “**implements**” when asynchronous functions are called, etc. The following callback functions are required to be implemented by interface implementation.

public void onFinish(String i_prt, int i_type, int i_value1, int i_value2);

Conditions for callback notification and arguments of functions are described below.

Callback conditions	i_type	i_value1	i_value2
When receiving a status.	1	Old status value	New status value
When receiving an extended information.	2	Extended information ID	0x00
When the transmitting data cue becomes empty.	3	0x00	0x00
NScanPrinters result notification	4	Execution result	Number of detected cases
NOpenPrinter result notification	5	Execution result	0x00
NResetPrinter result notification	6	Execution result	0x00
NFirmwareDL result notification	7	Execution result	0x00
NTCPPortLock result notification	8	Execution result	0x00
NBufferClear result notification	9	Execution result	0x00
NBlockSendSetting result notification	10	Execution result	0x00
NBtPairing result notification	11	Execution result	0x00

Each callback condition (argument with i_type from 1 to 11) are described on the next page.

Interface name	NCallback
Processing description	This interface is used for callback notification. 1. When a status is changed. It will be notified when the printer status is first obtained after printer connection and the status is changed since then. The value to be notified is same as the value obtained by NGetStatus function. It will never be notified when the printer is not opened. 2. When receiving an extended information (except for the Extended Information ID1). It will be notified when an Extended Information is received by SDK from the printer. Since it notified Extended Information ID, data itself needs to be obtained by NGetInformation function. 3. When the transmitting data cue becomes empty. Data transmitting functions in SDK (NPrint, NDPrint, NImangePrinte and NImageprintF) store the data in the transmitting que inside SDK, and then transmitting data to the printer sequentially. When the que becomes empty, it will be notified. It does not mean that all data processing has been completed on the printer side. Whether all commands have been completed or one or some of them have failed in the printer can be judged by using the printing start / end commands of the printer. Results of asynchronous functions execution of 4 to 11 will be notified by callbacks. Caution Please implement processes inside each callback functions on the customer side. Screen drawing process cannot be performed from the callback functions. In this case, please use android.os.Handler. * Implementation of this call back can be done by the program of sample application. It is described on the next page.

<Excerpt from sample application (NPI SDK program)>

```
public class NPISDK extends Activity implements OnClickListener, NCallback
{
    // NPI SDK Class
    private NPrinterLib objLib;
```



Implement NCallback

```
@Override
public void onClick(View v)
{
    Intent intent;
    int nmsRet;

    if(v == btnPrtInf)
    {
        intent = new Intent(this.getContext(), SubForm01.class);
        startActivity(intent);
    }
    else if(v == btnOpenPrt)
    {
        if(!chkSendRetry.isChecked())
        {
            objLib.NSetSendRetry(NCommon.SEND_RETRY_ON);
        }
        else
        {
            objLib.NSetSendRetry(NCommon.SEND_RETRY_OFF);
        }

        objLib.NOpenPrinter(editPrtName.getText().toString(), this);
    }
}
```



Executing NOpenPrinter, and the following function will be called and after that process

```
@Override
public void onFinish(String i_prt, int i_type, int i_value1, int i_value2)
{
    Message msg = new Message();
    msg.arg1 = i_type;
    msg.arg2 = i_value1;

    mCallbackHandler.sendMessage(msg);
}
```

Callback function

Process of screen drawing, etc. will be performed by using the Handler.

* Please refer to sample program itself for detail information.

Function name		NSetCallback	
Argument name	IN/OUT	Type	Description
i_callback	I	np.sdk.NCallback	Class implementing callback interface
Return value	int		
·Error(negative value), Normal finish(0) *Please refer to error code table on another page.			
Processing description		Setting the callback notification.	
- Setting the callback interface implementing class. Please execute this function before doing callback notification. The interface settings can also be made by using the “NOpenPrinter” function.			
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 interface abstract method (callback function) on the class side. Screen drawing processing cannot be performed from within the callback function. Please use android.os.Handler.			

Function name		NEnumPrinters	
Argument name	IN/OUT	Type	Explanation
o_printers	O	npk.sdk.data.NString	Printer name (csv : Enumerating them in a Comma Separated Value form)
o_size	O	npk.sdk.data.NInt	Printer name byte size of o_printers (Null can be specified.)
Return value	int		
·Error(negative value), Normal finish(0) *Please refer to error code table on another page.			
Processing description		Generating printer names and enumerating them.	
- It creates a printer information management file (NPrinterInf.inf) under [file allocation path]/npk and stores a list of printer names in the o_printers argument. e.g.) PRT001,PRT002,USB-XXX,AAA It detects USB, Bluetooth and TCP/UDP (*1) ports, assigns a name of "PRTxxx" (xxx is a number from 001 to 999), enumerates and returns them in csv (comma separated). Please make sure to call this function in order to open the printer when printer information management file is not generated because it (NOpenPrinter) cannot be done then. This function also has to be called when a printer is added. In addition, although, o_size exists for the compatibility with Windows version SDK, it will never be used with Android version. Printer list obtained by this function can also be obtained by referring to printer information management file (NPrintInf.inf). *1 Pairing with the printer is required before calling this function when using it with Bluetooth and executing NScanPrinters has to be done before using it with TCP/UDP. *2 Once printer name is generated, it will not be deleted by releasing the pairing or pulling off the USB plug. Please use NDeletePrinter function in order to delete it. *3 Maximum 999 xxx for printer name allocation can be generated. If more than that is tried to generate, an error will occur. Deleting unnecessary printer names by using NDeletePrinter or renaming them by using NRenamePrinter, etc. will be required.			

Function name		NDeletePrinter	
Argument name	IN/OUT	Type	Description
i_prt	I	String	Printer name to be deleted
Return value	Int		
·Error(negative value), Normal end(0) *Please refer to error code table.			
Processing description		Deleting printer names.	
- 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).			
Caution An error will occur when the printer information management file does not exist or it contains nothing.			

Function name	NRenamePrinter		
Argument name	IN/OUT	Type	Description
i_beforeprt	I	String	Printer name to be changed
i_afterprt	I	String	Printer name after the change
Return value	Int		
·Error(negative value), Normal end(0) *Please refer to error code table.			
Processing description	Renaming printer names. - 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 names 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) : []¥/:?*”<> ’, Furthermore, an error message will be returned when the printer to be renamed is open, Please use this function with the status of closing the connection to the target printer.		

Function name	NGetPrinterInf		
Argument name	IN/OUT	Type	Description
i_prt	I	String	Printer name
o_ports	O	np.sdk.data.NString	Port information (csv : To enumerate them in a Comma Separated Value form)
o_size	O	np.sdk.data.Nint	Size of o_ports bytes (Null can be specified.)
Return value	Int		
·Error (negative value), Normal end (0) * Please refer to error code table.			
Processing description	Obtaining detailed information for the specified printer name.		
- Searching the printer information management file for the printer name specified by the argument and storing the following information in o _ ports. •Connection types - USB : 1 - Bluetooth : 2 - TCP/UDP : 4 •Connection information - USB : vendor ID 4 digits + protocol ID 4digits (Both of them are hexadecimal character strings.) = 8digits - Bluetooth : name of the paired device/serial ID - TCP/UDP : printer model name : IP address : port number : MAC address e.g.) 1, 10511000 2, AAA-XX123/0003F3598C1A 4, NP-M320:192.168.92.1:9001:ABCDABCDABCD Information obtained by o_ports can also be obtained by referring to printer information management file (NPrintInf.inf). In addition, although, o_size exists for the compatibility with Windows version SDK, it will never be used with Android version. (It can be obtained by requesting o_ports size information.)			
Caution In case of using the same pairing name by multiple among printers with Bluetooth or using same printer model name with TCP/UDP, please distinguish by serial ID and MAC address. (described by red characters above)			

Function name	NGetPrinterFromID		
Argument name	IN/OUT	Type	Description
i_ID	I	String	Inputting one of the followings. •Bluetooth serial ID •USB vendor ID + product ID •MAC address of the TCP printer
o_printer	O	Nstring	Printer name
Return value	Int		
·Error (negative value), Normal end (0) * Please refer to error code table.			
Processing description	Obtaining the printer names for various IDs.		
- Printer name will be stored in the argument o_printer by specifying one of the followings to the argument i_ID. •Bluetooth serial ID •USB vendor ID + product ID •MAC address of the TCP printer Bluetooth serial ID and MAC address can be checked by executing self-diagnostic printing.			
Caution Printer name is required to be generated in the printer information management file (NPrintInf.inf) by NEnumPrinters function beforehand.			

Function name	NOpenPrinter		
Argument name	IN/OUT	Type	Description
i_prt	I	String	Printer name to open be opened
i_callback	I	npi.sdk.NCallback	Class implementing callback interface (When it is omitted, no callback notification will be returned.)
Return value	Void		
·Nothing (Please use the callback using the function NCallback or the function NOpenResult function in order to get the result.)			
Processing description			
Connecting the application to the printer. - Executing the printer opening process. Specifying the printer name obtained by NEnumPrinters to the argument i_prt. The result of the opening process by this function can be checked by the following 2 ways. 1. Implementing “NCallback” interface and receiving the result by using the callback function. 2. Checking the result by calling “NOpenResult” function (not recommended). Specifying the class implementing callback interface to the argument i_callback. Please implement the abstract method (callback function) to the class side. Since this function is processed asynchronously, the result of opening will be notified to the callback function. Please make sure to execute the printing process like NPrint, etc. after getting the value indicating success for the opening process. <When connected by USB interface> If the USB communication permission has not been granted to the device, a communication permission dialog is output. If communication has already been permitted, no dialog is output.			

Caution

If the power of the printer opened by this function is turned off, retry connection will be tried. When the power is turned on again, a USB communication permission dialog box will be displayed during the USB connection. Please Click “OK”. If "Cancel" is clicked, it will be terminated with an error as a retried connection failure.

Please use “NClosePrinter” on page 29 in order to stop the retry process itself. If the retry processing has been continued for 5 minutes, the reconnection process will be stopped..

Function name	NOpenPrinter
Processing description	

Caution

- Once callback class is set, callback notification is sent to the class once set unless null is specified in the NSetCallback function.
When the class instance has already been deleted, an exception will be generated by SDK.
- Please make sure to execute closing (NClosePrinter) after opening. If the application is stopped without this closing, closing cannot be detected on the printer side during TCP/UDP connection.
- Status information will be checked at the opening from the printer. A time-out error will be returned if the status cannot be received after 10 seconds from the opening, In this case, please retry the process after closing process (NClose Printer).
- After opening (including automatic reconnection), please confirm that the status can be obtained before transmitting data.
- When using it with TCP/UDP connection, UDP receiving thread is required to be activated by NInitializeNetwork function beforehand.

Function name	NOpenResult		
Argument name	IN/OUT	Type	Description
i_prt	I	String	Printer name
Return value	Int		
·Error (negative value), Normal end (0) , still during opening processing (positive value) * Please refer to error code table.			
Processing description	(Obtaining the result of opening. This function became not recommended function at Ver.3.0.0.0.)		

- The result of the function NOpenPrinter is obtained by this function.
NOpenPrinter is working in progress if a number from 100 to 199 is obtained. Please continue obtaining it until Error (negative value) or Normal end (0) is obtained.

Reference

From Ver. 3.0.0.0, it has been recommended to get the result by callback function using NCallback interface.

After calling NOpenPrinter, it is required to leave the main thread process one.

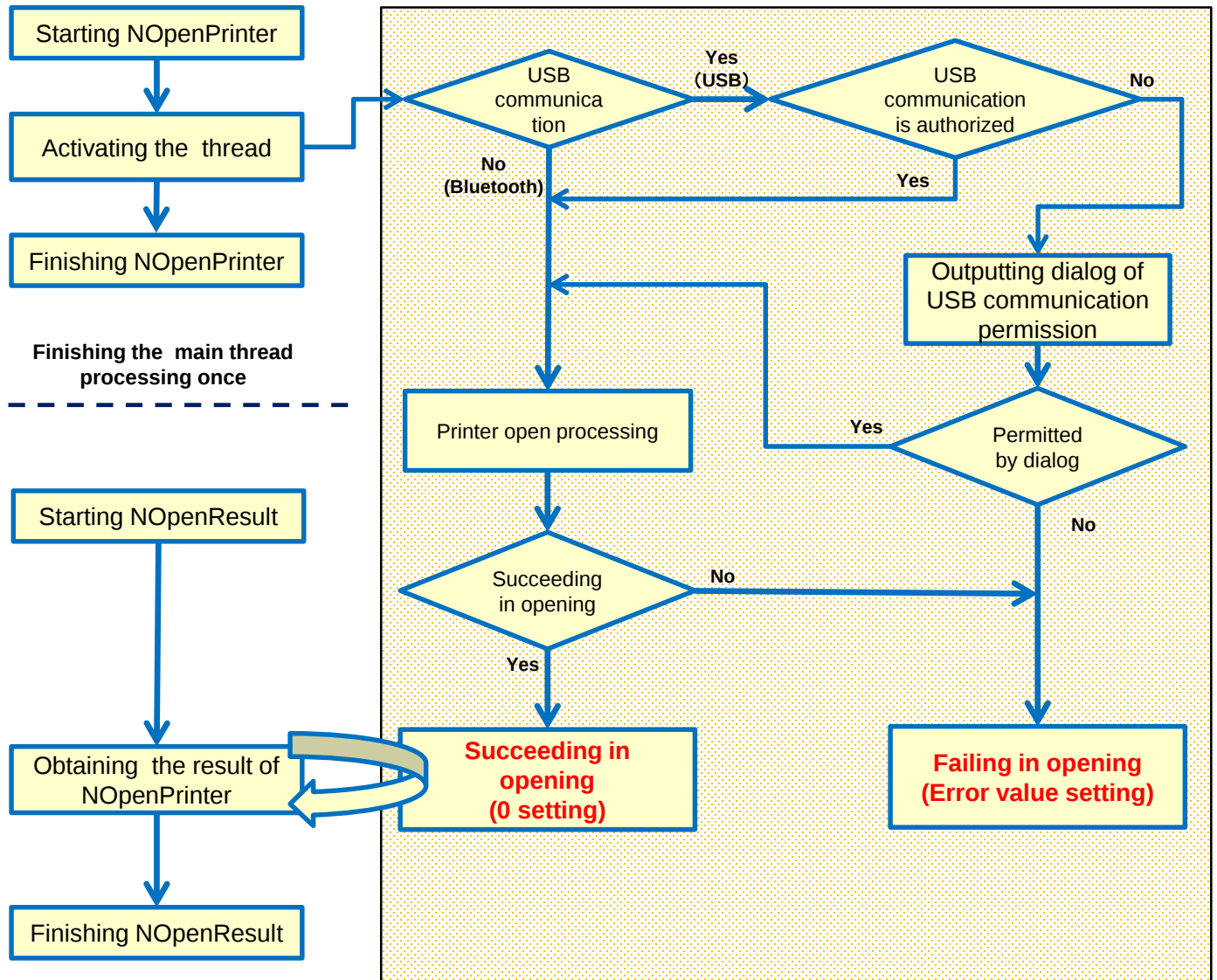
A series of consecutive "NOpenPrint => NOpenResult" processes cannot be executed. Checking will be easier by using NCallback for obtaining the result as callback or using java.util.Timer for calling NOpenResult periodically.

When making the result of NOpenResult return to the main thread from the timer thread, please implement it by the process transmitting the message to android.os.Handler.

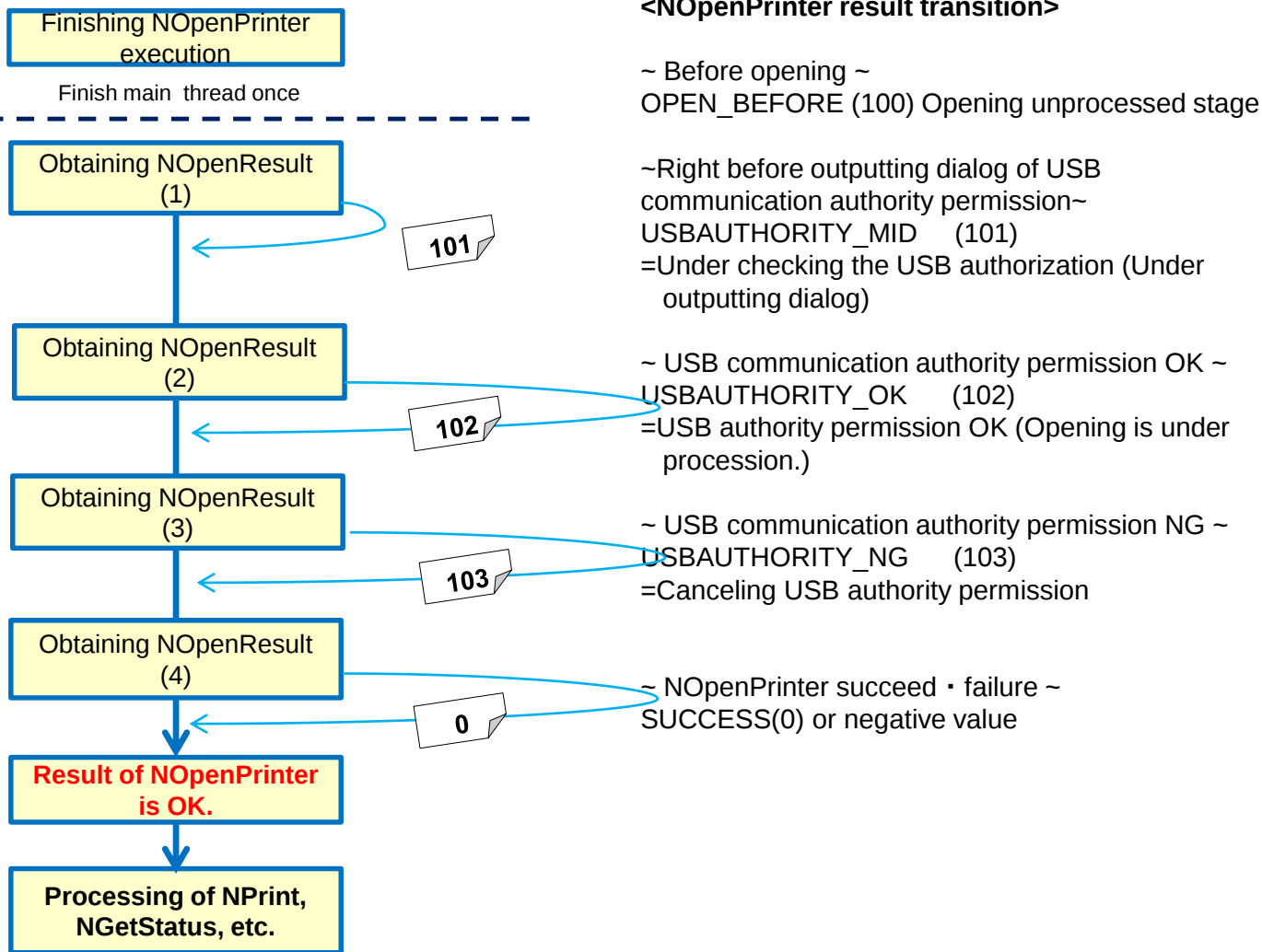
NOpenPrinter / NOpenResult Processing overview chart

Obtaining the result as callback by NCallback has been recommended from Ver.3.0.0.0.

The following process chart is for the compatibility with old versions.



Android API Reference



Please configure it so as to finish main thread once after calling `NOpenPrinter`.

After that, please call `NOpenResult` repeatedly until obtaining "0" or negative value.

If a number from 100 to 199 is obtained, it has not been opened yet or it is under USB authority permission process.
(`NOpenPrinter` execution has not be finished.)

Open processing will be completed at the stage of obtaining normal end (0).
Please follow these procedures in order to print or obtain status.

Function name	NClosePrinter		
Argument name	IN/OUT	Type	Description
i_prt	I	String	Printer name
Return value	Int		
·Error (negative value), Normal end (0) * Please refer to error code table.			
Processing description	Disconnecting the connection to the printer having been connected.		
- This function executes the printer closing process. Processes attempting reconnection (retry connection) will also be stopped.			

Function name	NClosePrinters		
Argument name	IN/OUT	Type	Description
Return value	Int		
·Normal end (0)			
Processing description	Disconnecting the connection to all printers having been connected.		
- Closing all opened printers.			
Caution This function returns only “0” (normal end) as a returned value. Even when it is already closed, it returns “0” (normal end). If you would like to get the error value, please use the NClosePrinter (closing only one opened printer) function.			

Function name		NPrint	
Argument name	IN/OUT	Type	Description
i_prt	I	String	Printer name
i_dat	I	String	Transmitting data (hexadecimal character string)
i_size	I	int	Number of output bytes
o_jobid	O	npd.sdk.data.NInt	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 data to the printer.		
- 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) => Being converted as character string ("ABC" => 0x41,0x42, 0x43) 2. File name in <> (angle brackets) (including path) => Outputting file contents (binary data). 3. File name in [] (brackets) (including path) => Outputting the image file with raster block format. (The image printing conforms to the NImagePrint function.) 4. Character string with ' (single quotation mark) at the head. => Being processed as a comment. (Not being output.) How to handle character code is described on the next page.			

Caution

- Please allocate NCharSetInf.inf file under [file allocation path]/npi/]in order to output as Shift JIS. "Shift JIS" is required to be input inside the file. Also, the printer must be set to the Shift JIS.

After checking Product Specifications for each printer, please check the "Selecting Kanji code system" command•memory switch setting. (It differs depending on printers.)

- * NCharSetInf file is applied only to NPrint function. Furthermore, this file does not exist by default. Read character code will be stored in mStrCharSet parameter as character strings.
- * NCharSetInf Please calculate the number of output bytes by Shift JIS during Shift JIS transformation.
=> The format will be `String.getBytes("Shift_JIS").length`.

Function name		NDPrint	
Argument name	IN/OUT	Type	Description
i_prt	I	String	Printer name
i_dat	I	byte[]	Transmitting data
i_size	I	int	(hexadecimal character string)
o_jobid	O	np.sdk.data.NInt	Number of output bytes 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 data to the printer.		
·Transmitting the specified bytes array data to the printer. It will not be transformed.			

Function name		NImagePrint	
Argument name	IN/OUT	Type	Description
i_prt	I	String	Printer name
i_bmp	I	Bitmap	Bitmap class for Android (android.graphics.Bitmap)
i_width	I	int	Width
i_height	I	int	Height
i_putType	I	byte	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	npi.sdk.data.Nint	Print job ID (Null can be specified.)
Return value	Int		
·Error (negative value), Normal end (0) * Please refer to error code table.			
Processing description		Specifying the bitmap class and performing image printing.	
- Transmitting the specified Bitmap class to the printer. The height and width beyond the Bitmap class size cannot be specified.			
Caution - Faster printing can be executed by registering them into fixed bit image in the printer in order to print the same image every time such as receipt logo printing, etc. - 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. Please refer to Product Specification of each printer for details.			

Function name	NImagePrintF		
Argument name	IN/OUT	Type	Description
i_prt	I	String	Printer name
i_bmp	I	String	Image file name (including path)
i_putType	I	byte	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	np.sdk.data.NInt	Print job ID (Null can be specified.)
Return value	Int		
·Error (negative value), Normal end (0) * Please refer to error code table.			
Process description	Specifying the name of the image file and performing image printing.		
- Transmitting the specified bmp/jpg/png file to the printer. Caution - The Bluetooth connection may become unstable (disconnected) because printing from the file name consumes a lot of memory during images reading. Therefore there is a 500msec long sleep process in order to avoid disconnection during data transmission. However, a sleep process will not be processed during the document opening by calling NStartDoc. - Faster printing can be executed by registering them into fixed bit image in the printer in order to print the same image every time such as receipt logo printing, etc. - 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. Please refer to Product Specification of each printer for details.			

Function name	NGetStatus		
Argument name	IN/OUT	Type	Description
i_prt	I	String	Printer name
o_status	O	npi.sdk.data.NInt	Status
Return value	Int		
·Error (negative value), Normal end (0) * Please refer to error code table.			
Process description	Obtaining the printer status.		
- Returning the status obtained from the specified printer. * Please refer to the target printer's Product Specifications for values to be returned.			

Caution

If the printer goes offline immediately after opening the printer, please execute this function again. It may be before receiving the status from the printer.

This function does not perform log output with log type 3 (FUNC / function call). Please use log type 6 (when checking CHK / status / extended information).

Function name		NGetInformation	
Argument name	IN/OUT	Type	Description
i_prt	I	String	Printer name
i_id	I	byte	ID type
o_dat	O	byte[]	Storing area of extended information
o_time	O	npi.sdk.data.NLong	Update flag(Acquire the time of system with msec)(null is assignable)
Return value		Int	

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

Processing description

Obtaining printer's extended information.

- Obtaining information stored in target type 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.

Caution

This function does not perform log output with log type 3 (FUNC / function call).
Please use log type 6 (when checking CHK / status / extended information).

Extended Information (1/2)

Type 1 : 4 bytes (fixed) : update flag (4bytes)

Type 2 : 32 bytes (delimiter) : update flag (4bytes)

Type 3 : 8 bytes (fixed) : update flag (4 bytes)

Type 4 : 8 bytes (fixed) : update flag (4 bytes)

Type 5 : 4 bytes (fixed) : update flag (4 bytes)

Type 6 : 4 bytes (fixed) : update flag (4 bytes)

Type 7 : 4 bytes (fixed) : update flag (4 bytes)

Type 8 : 4 bytes (fixed) : update flag (4 bytes)

Type 9 : 16 bytes (fixed) : update flag (4 bytes)

Type 10 : 16 bytes (fixed) : update flag (4 bytes)

Type 11 : 64 bytes (delimiter) : update flag (4 bytes)

Type 12 : 32 bytes (delimiter) : update flag (4 bytes)

Type 13 : 32 bytes (fixed) : update flag (4 bytes))

Type 14 : 32 bytes (fixed) : update flag (4 bytes)

Type 15 : 16 bytes (fixed) : update flag (4 bytes)

Type 16 : 16 bytes (fixed) : update flag (4 bytes)

Type 17 : 16 bytes (fixed) : update flag (4 bytes)

Type 18 : 16 bytes (fixed) : update flag (4 bytes)

Type 19 : 8 bytes (fixed) : update flag (4 bytes)

Type 20 : 8 bytes (fixed) : update flag (4 bytes)

Type 21 : 8 bytes (fixed) : update flag (4 bytes)

Type 22 : 8 bytes (fixed) : update flag (4 bytes)

Type 23 : 8 bytes (fixed) : update flag (4 bytes)

Type 24 : 4 bytes (fixed) : update flag (4 bytes)

Type 25 : 4 bytes (fixed) : update flag (4 bytes)

Type 26 : 4 bytes (fixed) : update flag (4 bytes)

Type 27 : 4 bytes (fixed) : update flag (4 bytes)

Type 28 : 2 bytes (fixed) : update flag (4 bytes)

Type 29 : 2 bytes (fixed) : update flag (4 bytes)

Type 30 : 2 bytes (fixed) : update flag (4 bytes)

Type 31 : 2 bytes (fixed) : update flag (4 bytes)

<Extended status> 1st byte : 7~0, 2nd byte : 15~8,
3rd byte : 23~16, 4th byte : 31~24

<Model name>

<F/W version>

<Boot version>

<Memory switch setting information>

<Number of dot lines energizing the head>

<Number of fed dot lines>

<Number of cuts>

<User maintenance counter:

Number of dot lines energizing the head,

Number of fed dot lines,

Number of cuts,

Reserved>

<Reserved>

<Multipurpose information 63 bytes>

The head byte is "Sub ID". (It should be more than "1".)

63 bytes from 1st byte to 63th are for multipurpose information.

<NV (non-volatile memory) registration status, up to 256 images>

<Reading all settings simultaneously>

<Notifying the end of printing : Arbitrary ID and end status are described at the time of end command processing by specifying in the print start / end command.>

<USB serial ID>

<NV (non-volatile memory) area remaining size>

<Transmission completion notification: transferred job IDs having been transferred will be described.>

<Reserved>

<NV (non-volatile memory) used size by area>

<F/W checksum>

<Communication status information: USB: 0x0000 fixed

COM: 1st Byte, CTS

2nd byte, DSR

* Final signal status obtaining time stamp is set to the 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.

Extended Information (2/2)

Type 32 : 12 bytes (fixed) : update flag (4 bytes)

<Multipurpose information 8 bytes>

The head byte is "Sub ID".

3 bytes from 1st byte to 3rd byte are fixed to "00h".

9 bytes from 4th byte to 11th byte are for multipurpose information.

Type 33 : 8 bytes (fixed) : update flag (4 bytes)

<Multipurpose information 4 bytes>

The head byte is "Sub ID".

3 bytes from 1st byte to 3rd byte are fixed to "00h".

4 bytes from 4th byte to 7th byte are for multipurpose information.

Type 34 : 4 bytes (fixed) : update flag (4 bytes)

<Multipurpose information 4 bytes>

The head byte is "Sub ID".

1st byte to 3rd byte is fixed to "00h".

2nd byte and 3rd to byte are for multipurpose 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 hexadecimal character string

Type 40 : 2 bytes hexadecimal character string

* Since multipurpose information of type 32, 33 and 34 are different depending on printer models, please contact us separately

Function name		NResetPrinter	
Argument name	IN/OUT	Type	Description
i_prt	I	String	Printer name
Return value	Int		
·Error (negative value), Normal end (0) * Please refer to error code table.			
Processing description	Resetting the printer.		
- Resetting the printer. The print job being printed is also canceled. This function can be used during USB or TCP/UDP connection. An error will occur during Bluetooth connection. Since the reset request to the printer is performed by asynchronous processing, the processing result must be received by the callback function using NCallback. (Please refer to page 19.)			

Caution

<In case of USB connection>
Depending on printer models, there are some models that goes off-line for a certain period of time after this function is executed. If the printer goes off-line, a USB communication permission dialog will be output in retried connection. Please press “OK” in order to reconnect it. When this function is executed before reconnection, an error will occur.

<In case of TCP/UDP connection>
When the printer is reset, it will be restarted and WLAN will be disconnected. Please make WLAN reconnected and repeat the printer closing and opening in order to do printing again.
(Executing NClosePrinter and NOpenPrinter)

Function name	NStartDoc		
Argument name	IN/OUT	Type	Description
i_prt	I	String	Printer name
o_jobid	O	npi.sdk.data.NInt	Print job ID
Return value	Int		
·Error (negative value), Normal end (0) * Please refer to error code table.			
Processing description	Starting storing the transmitting data.		
- 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.			

Function name	NEndDoc		
Argument name	IN/OUT	Type	Description
i_prt	I	String	Printer name
Return value	Int		
·Error (negative value), Normal end (0) * Please refer to error code table.			
Processing description	Transmitting the stored data to the printer.		
- 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. Reference This function does not make an offline judgment because it is confirmed that the printer is online at the start of the document.			

Function name	NCancelDoc		
Argument name	IN/OUT	Type	Description
i_prt	I	String	Printer name
Return vale	Int		
·Error (negative value), Normal end (0) * Please refer to error code table.			
Processing description	Discarding the stored data.		
- 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	I	String	Target printer name
o_docnolist	O	npi.sdk.data.NString	Transmitting data Job ID left in the transmitting cue
Return vale	Int		

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

Processing description

Obtaining job IDs of the transmitting data in the transmission cue.

- Canceling the document. Returning the transmitting data job ID left in the transmitting cue to the argument o_docnolist in comma separated format.

When one of the data transmitting functions (NPrint, NDPrint, NImagePrint and NImagePrintF) is executed, it will first store data in transmitting cue inside SDK and then transmit data to the printer sequentially. Although data is transmitted from the transmitting cue immediately and is deleted from the cue normally, data waiting for transmission will be stored when there are extraordinarily many number of data.

Reference

Process by this function has no relations to the process by NStartDoc, NCancelDoc and NEndDoc.

This function is used for checking data waiting for transmission when the number of transmitting data is big.

Function name	NDeleteDoc		
Argument name	IN/OUT	Type	Description
i_prt	I	String	Target printer name
i_docno	I	int	Transmitting data Job ID to be deleted
Return value	Int		

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

Processing description

Deleting the untransmitted data remaining in the transmission cue.

- Deleting the printer name and the transmitting data job ID transmission from transmission waiting document. All the transmission waiting data will be targeted by specifying less than 0 as job ID.
- There may be a case that the specified job ID specified has already been deleted (been transmitted) from the cue at execution. In this case, it is not an error and WRN_NOEXISTDOC (the document does not exist) is returned.

Reference

Process by this function has no relations to the process by NStartDoc, NCancelDoc and NEndDoc.

This function is used for deleting data waiting for transmission when the number of transmitting data is big.

Function name	NBarcode		
Argument name	IN/OUT	Type	Description
i_prt	I	String	Printer name
i_fontName	I	String	Barcode font name
i_bmp	IO	Bitmap	Bitmap class (android.graphics.Bitmap)
i_x	I	int	Left
i_y	I	int	Top
i_width	I	int	Width
i_height	I	int	Height
i_dat	I	byte[]	Barcode data
i_size	I	int	Data size
Return value	Int		
·Error (negative value), Normal end (0) * Please refer to error code table.			
Processing description	Generating the barcode image.		
- Drawing a barcode and a 2D barcode with settings set to the barcode font and 2D barcode font in the printer on specified Bitmap. Please specify the font name used for barcode setting file (NBarcodeInf) to the font name of 2nd argument. Please refer to “Regarding barcode setting” in this document for NBarcodeInf.inf. Caution If the barcode data is large and the created bar code exceeds the data size, no error occurs and the barcode data without the excess part is created. In this case, the created barcode cannot be read. Please adjust the data size so that all the data can be included.			

Function name	NBarcode
Processing description	
Image of setting arguments	Bitmap class area (third argument – gray color frame part  The diagram shows a gray rectangular frame representing a bitmap. Inside the frame, there is a white rectangular area. Within the white area, a barcode is displayed with the numbers '1922033013002' below it. A red square encloses the barcode and the numbers. Dimension lines with arrows indicate the following parameters: 'i_x' is the horizontal distance from the left edge of the gray frame to the left edge of the white area; 'i_y' is the vertical distance from the top edge of the gray frame to the top edge of the white area; 'i_width' is the horizontal distance from the left edge of the gray frame to the right edge of the gray frame; 'i_height' is the vertical distance from the bottom edge of the gray frame to the top edge of the gray frame. * Setting the width, height, presence or absence of HRI characters, etc. of the barcode part enclosed in red square in the barcode setting file (NBarcodeInf). * The barcode rotation setting rotates the part enclosed in red square. Since the white part in the above figure does not rotate, please set the area with the width and height after rotation.

Function name		NFirmwareDL	
Argument name	IN/OUT	Type	Description
i_prt	I	String	Printer name
i_file	I	String	Fwf file name (Null can be specified.)
i_errflg	I	byte	Error check 0x00: Invalid (Forced transmission) 0x01: Valid
io_chksum	IO	npi.sdk.data.NShort	Fwf file checksum (Null can be specified. It is required for firmware check.)
io_jobid	IO	npi.sdk.data.NInt	Print job ID (Null can be specified.)
Return value	Int		
·Error (negative value), Normal end (0) * Please refer to error code table.			
Processing description		Updating the printer firmware.	
- Transmitting the specified fwf file to the printer. (Firmware download) Since the process by this function will be asynchronous, please obtain the result of the process by using callback function. Using this function during USB connection is recommended. It will take long time to complete the process due to the slow transmission speed during Bluetooth and TCP/UDP connection. If the fwf file is specified as Null, the checksum is obtained from the printer and compared with the checksum specified by the argument. (Firmware check) 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. - The processing result is required to be obtained by callback function when this function is used with effective error check. - Please use this function only for printers with implemented “printing start/end setting” command.			

Function name	NFirmwareResult		
Argument name	IN/OUT	Type	Description
i_prt	I	String	Printer name
Return value	Int		
·Error (negative value), Normal end (0), Firmware DL processing (+ value) * Please refer to error code table.			
Processing description	(Obtaining the result of the function “NFirmwareDL”. This function became not recommended function at Ver.3.0.0.0.)		
- The result of NFirmwareDL function (except for checksum obtaining during effective error check) can be obtained by this function. NFirmwareDL is still under processing when a positive value is obtained. Positive value means the status value. Please repeat obtaining values until normal end (0) or error (negative value) is obtained. When NFirmwareDL (error check with valid) is in practice, it will transition with the following return values. 1. Before calling NFirmwareDL function. (return vale 0) 2. Executing firmware downloading. (return vale 128) ···Bit7 of status turns ON 3. Finishing download, resetting process. (return value 255) ···Turning to off-line temporarily 4. Reconnection dialog is displayed. (It will be reconnected to the printer by pressing OK in this step.) 5. After calling NFirmwareDL function. (When it is ended normally, 0 will be returned.) * After firmware download is finished, printer reset will be executed.. If this function is called before the printer reset, an error will occur. Please implement the system so that the result of download can be obtained after the reconnection.			

Reference

It has been recommended to obtain the result by callback function with NCallback interface since Ver. 3.0.0.0.

Function name	NSetUSBProtocol		
Argument name	IN/OUT	Type	Description
i_prt i_type	I I	String 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 setting value by returning value doing the setting.
Return value	Int		
·Current (after setting) value of the check flag of receiving buffer amount (0,1) or error (negative value) * Please refer to error code table on another page.			
Processing description	Setting whether USB receiving buffer remaining size will be checked when transmitting the data.		
- This is a flag for determining whether or not to transmit data after checking the receiving buffer remaining size. The default setting is “0” and transmission after checking the receiving buffer remaining size will be executed. Please be sure to set “1” to it when using the printer which does not respond to receiving buffer remaining size checking. - This setting will be effective only with USB connection and it will be disregarded with Bluetooth or TCP/UDP connection.			

Function name	NSSetSendRetry		
Argument name	IN/OUT	Type	Description
i_Retry	l	int	The flag for retransmission when a data transmission error occurs. 0 : OFF (Retransmission will not be executed.) 1 : ON (Retransmission will be executed.) (default value)
Return value	Int		
·The flag value after being set			
Processing description	Specifying whether trying to retransmitting the data.		
- Whether executing retransmission in case of data transmission error or not can be set. Retransmission will executed when it is ON (1) which is the default setting.			
Caution - This setting will not be applied to the printer already being opened. Please close the printer once and reopen it after setting by this function.			

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		
·Error (negative value), Normal end (0) * Please refer to error code table.			
Processing description	(Only available for TCP/UDP communication) Starting / stopping the UDP receiving thread.		
- Activating / stopping UDP receiving thread. It is stopped at the initial state. During TCP / UDP communication, please activate the UDP receiving thread by this function before printer detection (NScanPrinters). When TCP/UDP communication is not used, please DO NOT execute this function.			

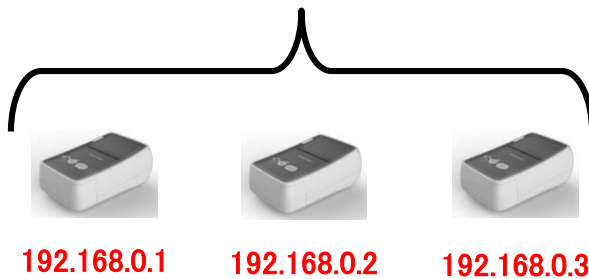
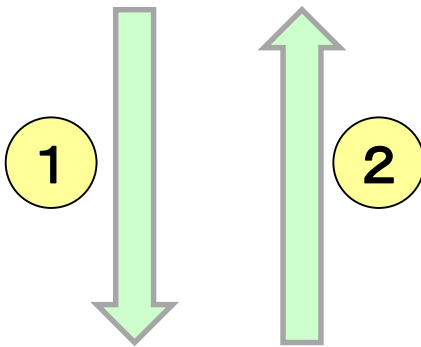
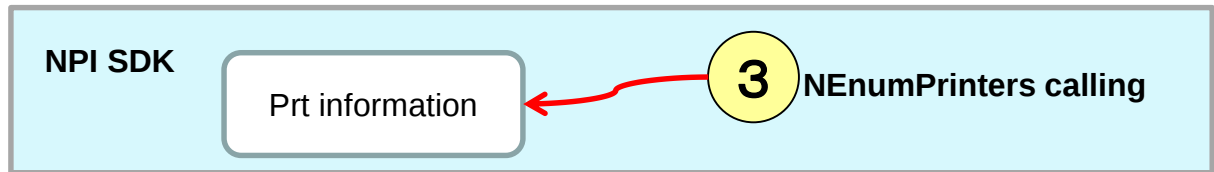
Function name	NScanPrinters		
Argument name	IN/OUT	Type	Description
i_waitmsec	l	int	Number of milliseconds to wait for printer enumeration response / data creation (Null can be specified.)
Return value	Int		
·Error (negative value), Normal end (0) * Please refer to error code table.			
Processing description	(Only available for TCP/UDP communication) Detecting TCP/UDP printers.		
- This is a function for detecting printers with TCP/UDP connection. This function does not have to be executed when printers with TCP/UDP are not used. The printer enumeration request of maintenance protocol is transmitted to broadcast addresses. It will receive printer enumeration responses for milliseconds specified by the argument i_waitmsec (Null can be specified. Default setting 3,000ms.) and then the data will be generated. “NEnumPrinters should be executed for printer information (getting the obtained data). (NEnumPrinters function is used for obtaining printer information separately.) This function detects printers asynchronously. Please obtain the process completion notification and the result (error code, number of detected printers) for completing detection by callback after implementing NCallback interface.			

Android API Reference

Printer name obtaining when using TCP/UDP(WLAN)



PC / tablet, etc.



* NScanPrinters function is called to create printer information.

Executing this function is needed when

- TCP/UDP printer is newly added.
- IP address of the printer is changed.

This function does not have to be executed every time NEnumPrinters function is used.

1 NScanPrinters calling
(Printer enumeration request : UDP / transmitting it widely to each printer)

2 Printer enumeration response from each printer with TCP/UDP connection :
receiving UDP as printer information in SDK.
(Please receive the completion of NScanPrinters process by the callback function.)

3 NEnumPrinters calling
(Obtaining TCP/UDP printer information based on the receiving data.)

Function name	NTCPPortLock		
Argument name	IN/OUT	Type	Description
i_prt i_type	I I	String int	Printer name Setting type Lock : 0 Unlock : 1
Return value	Int		
·Error (negative value), Normal end (0) * Please refer to error code table.			
Processing description	(Only available for TCP/UDP communication) Locking / unlocking the TCP/UDP communication.		
- Locking / unlocking TCP communication. Only the locked users can send data via TCP communication. Other users can be connected but cannot transmit data. An error will occur when it is not a printer with TCP/UDP connection.			

Function name	NBufferClear		
Argument name	IN/OUT	Type	Description
i_prt	I	String	Printer name
Return value	Int		
·Error (negative value), Normal end (0) * Please refer to error code table.			
Processing description	(Only available for TCP/UDP communication) Clearing the receiving buffer.		
- Clearing the receiving buffer (all data in the buffer). An error will occur when it is not a printer with TCP/UDP connection.			
Caution - The result is required to be obtained by the callback function using NCallback because the request to the printer is processed asynchronously. (Please refer to the page 19.)			

Function name	NBlockSendSetting		
Argument name	IN/OUT	Type	Description
i_prt i_type	I I	String int	Printer name Setting type Start : 0 End : 1
Return value	Int		
·Error (negative value), Normal end (0) * Please refer to error code table.			
Processing description	(Only available for TCP/UDP communication) Setting TCP batch transmission starting and ending. - Setting the TCP batch transmission start and end. An error will occur when it is not a printer with TCP/UDP connection.		
Caution - The result is required to be obtained by the callback function using NCallback because the request to the printer is processed asynchronously. (Please refer to the page 19.)			

Error code table (1/3)

This is the list of error codes used by this SDK. Although these are mainly used for the return values from the functions there are some error codes which are used only for SDK internal processes, and others are for maintaining compatibility to other OSes. Please refer to “List of error codes by functions” for return values from the functions.

N_SUCCESS	0	Normal end
N_ERR_PRTOPEN	-2	Printer open error
N_ERR_OFFLINE	-5	Off-line
N_ERR_PRTCLOSE	-6	Printer close error
N_ERR_STATUSTIMEOUT	-7	A time-out occurred during obtaining the status.
N_ERR_BTPAIRING	-8	Failed in Bluetooth pairing.
N_ERR_FILEOPEN	-10	File open error
N_ERR_PRTOUTPUT	-13	Printer output error
N_ERR_PRTINPUT	-14	Printer data receiving error
N_ERR_CHARSET	-15	Character code is not supported.
N_ERR_PRTILLEGAL	-16	The connected printer information is invalid.[
N_ERR_NONE_PRTLST	-21	No printer can be used.
N_ERR_NOHANDLE	-22	Printer is not opened.
N_ERR_LACKRESOURCE	-31	Lack of resource
N_ERR_NOTSUPPORTED	-40	Not supported by Android SDK.
N_ERR_LOADFROMFILE	-50	Failed in reading image file.
N_ERR_IMAGESIZE	-51	Invalid image size
N_ERR_LOADBITMAP	-52	Failed in reading image data from the Bitmap class.
N_ERR_LOADFORMFILE2	-53	Not being able to read image file during the data transmission.
N_ERR_RESETPRINTER	-60	Resetting failure
N_ERR_STARTDOC	-70	StartDoc function error
N_ERR_DOCNOTSTARTED	-71	Not in document start status
N_ERR_ALREADYSTARTED	-72	Already in document start status
N_ERR_FWFFILE	-80	fwf file error
N_ERR_FWF_CHECKSUM	-81	The checksum entered in the argument does not match the checksum obtained from the printer.
N_ERR_FWDL_TIMEOUT	-82	Firmware download time-out (Time-out of checking the printing start command)
N_ERR_FWCHK_TIMEOUT	-83	Time-out of checking the firmware checksum
N_ERR_FWDL_FOUNDERERROR	-84	Detecting an error in checking the status during firmware download.
N_ERR_FWCHK_FOUNDERERROR	-85	Detecting an unexpected error during firmware checking.

Error code table (2/3)

N_ERR_ARGUMENT	-90	Argument incorrect value error
N_ERR_ARGUMENT_01	-91	1st argument error
N_ERR_ARGUMENT_02	-92	2nd argument error
N_ERR_ARGUMENT_03	-93	3rd argument error
N_ERR_ARGUMENT_04	-94	4th argument error
N_ERR_ARGUMENT_05	-95	5th argument error
N_ERR_ARGUMENT_06	-96	6th argument error
N_ERR_ARGUMENT_07	-97	7th argument error
N_ERR_ARGUMENT_08	-98	8th argument error
N_ERR_ARGUMENT_09	-99	9th argument error
N_ERR_SOCKET_INI	-101	Socket error (Filled in initializing.)
N_ERR_SOCKET	-102	Socket error
N_ERR_ASCHICONVERT1	-103	Invalid ASCII conversion (size error)
N_ERR_ASCHICONVERT2	-104	Invalid ASCII conversion
N_ERR_SOCKCONNECT	-105	Connection error
N_ERR_WSAENOTCONN	-106	A socket is not connected.
N_ERR_SOCSENDDTIMEOUT	-107	Data transmission time-out
N_ERR_SOCKRECV	-108	Receiving error
N_ERR_SOCKRECVTIMEOUT	-109	Receiving time-out
N_ERR_HOSTNAME	-110	Host name absence
N_ERR_UDPTHREADSTARTED	-111	UDP thread has already been started.
N_ERR_UDPTHREADSTOPPED	-112	UDP thread is stopped.
N_ERR_UDPTHREADSTOP	-113	UDP thread is not able to be stopped.
N_ERR_BRDADDR	-114	Broadcast address were not be able to be obtained.
N_ERR_LOGWRITE	-120	Failed in writing a log.
N_ERR_PRTINFO_CREATE	-130	Failed in creating the printer information file.
N_ERR_PRTINFO_READ	-131	Failed in reading the printer information file.
N_ERR_PRTINFO_WRITE	-132	Failed in writing the printer information file.
N_ERR_PRTNAME_ALLOC	-133	Failed in allocating the printer name.
N_ERR_PRTRENAME_BEFORE	-134	Printer name before being changed does not exist.
N_ERR_PRTRENAME_AFTER	-135	Printer name after being changed has already been used.
N_ERR_PRTINFO_GET	-136	Failed in obtaining printer information.
N_ERR_PRTINFO_ILLEGAL	-137	Incorrect printer information file
N_ERR_PRTINFO_DELETE	-138	Failed in deleting the printer name
N_ERR_PRTINFO_NOTFOUND	-139	Not existing printer name

Error code table (3/3)

N_ERR_DEVICE_NOTSUPPORT	-150	Connection type is not supported.
N_ERR_BCDSETTINGFILE	-160	Invalid barcode setting file
N_ERR_CREATEBCDFILE	-161	Failed in creating the barcode file.
N_ERR_CREATEBCDDATA	-162	Failed in creating the barcode data.
N_ERR_MNT_HEADER	-200	Maintenance header is invalid.
N_ERR_FLAGSQR	-201	QR flag of the maintenance response (Flags) is invalid.
N_ERR_FLAGSFORMAT	-202	A format error is detected in the maintenance response (Flags).
N_ERR_FLAGSBUSY	-203	"Busy" is detected in the maintenance response (Flags).
N_ERR_FLAGSUNDEFINED	-204	"Not implemented" is detected in the maintenance response (Flags).
N_ERR_FLAGSREJECT	-205	A refusal is detected in the maintenance response (Flags).
N_ERR_FLAGSOTHER	-206	One of other errors is detected in the maintenance response (Flags).
N_ERR_MNT_QID	-207	Query of the maintenance response is invalid.
N_ERR_MNT_QR	-208	Spare flag of the maintenance response is invalid.
N_ERR_MNT_QPARAM	-209	Query parameter of the maintenance response is invalid.
N_ERR_MNT_OTHER	-210	Other errors of the maintenance response
N_ERR_PERMISSION	-220	Required authority is not assigned.
N_WRN_PRTALREADYOPEN	10	Printer has already been opened.
N_WRN_NOTEXISTDOC	11	Document does not exist.
OPEN_BEFORE	100	The stage when open has not been processed
USBAUTHORITY_MID	101	Under checking the USB authorization.(A dialogue is being output.)
USBAUTHORITY_OK	102	USB authorization is OK. (Open process is being executed.)
USBAUTHORITY_NG	103	USB authorization has been canceled.

Android API Reference

List of returned value for each function (1/7)

* Functions to be responded by callbacks are indicated in the column of “Notes”.

Function name	Definition name	Defined value	Notes
NSetCallback	No return value	—	
NEnumPrinters	N_SUCCESS	0	
	N_ERR_NONE_PRTLIST	-21	
	N_ERR_ARGUMENT_01	-91	
	N_ERR_PRTINFO_CREATE	-130	
	N_ERR_PRTINFO_READ	-131	
	N_ERR_PRTINFO_WRITE	-132	
	N_ERR_PRTNAME_ALLOC	-133	
	N_ERR_PERMISSION	-220	
NDeletePrinter	N_SUCCESS	0	
	N_WRN_PRTALREADYOPEN	1	
	N_ERR_PRTINFO_READ	-131	
	N_ERR_PRTINFO_WRITE	-132	
	N_ERR_PRTINFO_ILLEGAL	-137	
	N_ERR_PRTINFO_DELETE	-138	
	N_ERR_PRTINFO_NOTFOUND	-139	
	N_ERR_PERMISSION	-220	
NRenamePrinter	N_SUCCESS	0	
	N_WRN_PRTALREADYOPEN	1	
	N_ERR_ARGUMENT_01	-91	
	N_ERR_ARGUMENT_02	-92	
	N_ERR_PRTINFO_READ	-131	
	N_ERR_PRTINFO_WRITE	-132	
	N_ERR_PRTRENAME_BEFORE	-134	
	N_ERR_PRTRENAME_AFTER	-135	
	N_ERR_PRTINFO_ILLEGAL	-137	
	N_ERR_PERMISSION	-220	
NGetPrinterInf	N_SUCCESS	0	
	N_ERR_ARGUMENT_01	-91	
	N_ERR_ARGUMENT_02	-92	
	N_ERR_PRTINFO_READ	-131	
	N_ERR_PRTINFO_GET	-136	
	N_ERR_PRTINFO_ILLEGAL	-137	
	N_ERR_PERMISSION	-220	
NGetPrinterFromID	N_SUCCESS	0	
	N_ERR_ARGUMENT_01	-91	
	N_ERR_ARGUMENT_02	-92	
	N_ERR_PRTINFO_READ	-131	
	N_ERR_PRTINFO_GET	-136	
	N_ERR_PERMISSION	-220	

Android API Reference

List of returned value for each function (2/7)

Function name	Definition name	Defined value	Notes
NOpenPrinter	N_SUCCESS	0	Callback
	N_WRN_PRTALREADYOPEN	11	Callback
	N_ERR_PRTOPEN	-2	Callback
	N_ERR_STATUSTIMEOUT	-7	Callback
	N_ERR_ARGUMENT_01	-91	Callback
	N_ERR_UDPTHREADSTOPPED	-112	Callback
	N_ERR_PRTINFO_READ	-131	Callback
	N_ERR_PRTINFO_GET	-136	Callback
	N_ERR_PRTINFO_ILLEGAL	-137	Callback
	N_ERR_PRTINFO_NOTFOUND	-139	Callback
	N_ERR_DEVICE_NOTSUPPORT	-150	Callback
	N_ERR_PERMISSION	-220	Callback
NClosePrinter	N_SUCCESS	0	
	N_ERR_PRTCLOSE	-6	
	N_ERR_PRTILLEGAL	-16	
	N_ERR_NONE_PRTLIST	-21	
	N_ERR_NOHANDLE	-22	
	N_ERR_ARGUMENT_01	-91	
NClosePrinters	N_SUCCESS	0	
NPrint	N_SUCCESS	0	
	N_ERR_PRTOUTPUT	-13	
	N_ERR_CHARSET	-15	
	N_ERR_NONE_PRTLIST	-21	
	N_ERR_NOHANDLE	-22	
	N_ERR_LACKRESOURCE	-31	
	N_ERR_ARGUMENT	-90	
	N_ERR_ARGUMENT_01	-91	
	N_ERR_ARGUMENT_02	-92	
	N_ERR_ARGUMENT_03	-93	
NDPrint	N_SUCCESS	0	
	N_ERR_PRTOUTPUT	-13	
	N_ERR_NONE_PRTLIST	-21	
	N_ERR_NOHANDLE	-22	
	N_ERR_ARGUMENT_01	-91	
	N_ERR_ARGUMENT_02	-92	
	N_ERR_ARGUMENT_03	-93	

Android API Reference

List of returned value for each function (3/7)

Function name	Definition Name	Defined value	Notes
NImagePrint	N_SUCCESS	0	
	N_ERR_PRTOUTPUT	-13	
	N_ERR_NONE_PRTLIST	-21	
	N_ERR_NOHANDLE	-22	
	N_ERR_LOADBITMAP	-52	
	N_ERR_ARGUMENT	-90	
	N_ERR_ARGUMENT_01	-91	
	N_ERR_ARGUMENT_02	-92	
	N_ERR_ARGUMENT_03	-93	
	N_ERR_ARGUMENT_04	-94	
	N_ERR_ARGUMENT_05	-95	
NImagePrintF	N_SUCCESS	0	
	N_ERR_PRTOUTPUT	-13	
	N_ERR_NONE_PRTLIST	-21	
	N_ERR_NOHANDLE	-22	
	N_ERR_LOADFROMFILE	-50	
	N_ERR_LOADBITMAP	-52	
	N_ERR_LOADFROMFILE2	-53	
	N_ERR_ARGUMENT	-90	
	N_ERR_ARGUMENT_01	-91	
	N_ERR_ARGUMENT_02	-92	
	N_ERR_ARGUMENT_03	-93	
NGetStatus	N_SUCCESS	0	
	N_ERR_OFFLINE	-5	
	N_ERR_ARGUMENT_01	-91	
	N_ERR_ARGUMENT_02	-92	
NGetInformation	N_SUCCESS	0	
	N_ERR_NONE_PRTLIST	-21	
	N_ERR_NOHANDLE	-22	
	N_ERR_ARGUMENT_01	-91	
	N_ERR_ARGUMENT_02	-92	
	N_ERR_ARGUMENT_03	-93	
NResetPrinter	N_SUCCESS	0	Returned value by the function, callback
	N_ERR_NONE_PRTLIST	-21	Returned value by the function, callback
	N_ERR_NOHANDLE	-22	Returned value by the function, callback
	N_ERR_NOTSUPPORTED	-40	Returned value by the function, callback
	N_ERR_RESETPRINTER	-60	Callback
	N_ERR_ARGUMENT_01	-91	Returned value by the function, callback
	N_ERR_FLAGSQR	-201	Callback
	N_ERR_FLAGSFORMAT	-202	Callback
	N_ERR_FLAGSBUSY	-203	Callback
	N_ERR_FLAGSUNDEFINED	-204	Callback
	N_ERR_FLAGSREJECT	-205	Callback
	N_ERR_FLAGSOTHER	-206	Callback
	N_ERR_MNT_QPARAM	-209	Callback

Android API Reference

List of returned value for each function (4/7)

Function name	Definition name	Defined value	Notes
NStartDoc	N_SUCCESS	0	
	N_ERR_NONE_PRTLIST	-21	
	N_ERR_NOHANDLE	-22	
	N_ERR_ALREADYSTARTDOC	-72	
	N_ERR_ARGUMENT_01	-91	
	N_ERR_ARGUMENT_02	-92	
NEndDoc	N_SUCCESS	0	
	N_ERR_NONE_PRTLIST	-21	
	N_ERR_NOHANDLE	-22	
	N_ERR_DOCNOTSTARTED	-71	
	N_ERR_ARGUMENT_01	-91	
NCancelDoc	N_SUCCESS	0	
	N_ERR_NONE_PRTLIST	-21	
	N_ERR_NOHANDLE	-22	
	N_ERR_DOCNOTSTARTED	-71	
	N_ERR_ARGUMENT_01	-91	
NEnumDoc	N_SUCCESS	0	
	N_ERR_NONE_PRTLIST	-21	
	N_ERR_NOHANDLE	-22	
	N_ERR_ARGUMENT_01	-91	
NDeleteDoc	N_SUCCESS	0	
	N_WRN_NOTEXISTDOC	10	
	N_ERR_NONE_PRTLIST	-21	
	N_ERR_NOHANDLE	-22	
NBarcode	SUCCESS	0	
	ERR_ARGUMENT	-90	
	ERR_ARGUMENT_01	-91	
	ERR_ARGUMENT_02	-92	
	ERR_ARGUMENT_03	-93	
	ERR_ARGUMENT_04	-94	
	ERR_ARGUMENT_05	-95	
	ERR_ARGUMENT_06	-96	
	ERR_ARGUMENT_07	-97	
	ERR_ARGUMENT_08	-98	
	ERR_ARGUMENT_09	-99	
	ERR_CREATEBCDDATA	-161	
NInitializeNetwork	SUCCESS	0	
	ERR_ARGUMENT_01	-91	
	ERR_UDPTHREADSTARTED	-111	
	ERR_UDPTHREADSTOPPED	-112	
	ERR_UDPTHREADSTOP	-113	

List of returned value for each function (5/7)

Function name	Definition name	Defined value	Notes
NFirmwareDL (Firmware check) (Firmware download)	N_SUCCESS	0	Returned value by the function, callback
	N_ERR_NONE_PRTLIST	-21	Returned value by the function, callback
	N_ERR_NOHANDLE	-22	Returned value by the function, callback
	N_ERR_ARGUMENT_01	-91	Returned value by the function, callback
	N_ERR_ARGUMENT_02	-92	Returned value by the function, callback
	N_ERR_NOHANDLE	-22	Returned value by the function, callback
	N_ERR_FWFFILE	-80	Returned value by the function, callback
	N_ERR_FWF_CHECKSUM	-81	Returned value by the function, callback
	N_ERR_FWCHK_TIMEOUT	-83	Returned value by the function, callback
	N_ERR_FWCHK_OTHER	-85	Returned value by the function, callback
	N_ERR_ALREADYSTARTDOC	-72	Callback
	N_ERR_FWDL_TIMEOUT	-82	Callback
	N_ERR_FWDL_FOUNDERERROR	-84	Callback
	N_ERR_ARGUMENT_3	-93	Returned value by the function, callback
NSetUSBProtocol	N_SUCCESS	0	
	N_ERR_NONE_PRTLIST	-21	
	N_ERR_NOHANDLE	-22	
	N_ERR_ARGUMENT_01	-91	
	N_ERR_ARGUMENT_02	-92	
	N_ERR_DEVICE_NOTSUPPORT	-150	
	The flag for checking the receiving buffer remaining size after setting.	positive value	
NSetSendRetry	The value set to the retransmitting flag.	boolean	
NInitializeNetwork	N_SUCCESS	0	
	N_ERR_ARGUMENT_01	-91	
	N_ERR_UDPTHREADSTARTED	-111	
	N_ERR_UDPTHREADSTOPPED	-112	
	N_ERR_UDPTHREADSTOP	-113	

Android API Reference

List of returned value for each function (6/7)

Function name	Definition name	Defined value	Notes
NScanPrinters	N_SUCCESS	0	Returned value by the function, callback
	N_ERR_ARGUMENT_01	-91	Returned value by the function, callback
	N_ERR_SOCKET	-102	Callback
	N_ERR_UDPTHREADSTOPPED	-112	Returned value by the function, callback
	N_ERR_BRDADDR	-114	Callback
	N_ERR_PERMISSION	-220	Returned value by the function
NTCPPortLock	N_SUCCESS	0	Returned value by the function, callback
	N_ERR_PRTOUTPUT	-13	Returned value by the function, callback
	N_ERR_NOHANDLE	-22	Returned value by the function, callback
	N_ERR_NOTSUPPORTED	-40	Returned value by the function, callback
	N_ERR_ARGUMENT_01	-91	Returned value by the function, callback
	N_ERR_ARGUMENT_02	-92	Returned value by the function, callback
	N_ERR_FLAGSQR	-201	Callback
	N_ERR_FLAGSFORMAT	-202	Callback
	N_ERR_FLAGSBUSY	-203	Callback
	N_ERR_FLAGSUNDEFINED	-204	Callback
	N_ERR_FLAGSREJECT	-205	Callback
	N_ERR_FLAGSOTHER	-206	Callback
NBufferClear	N_SUCCESS	0	Returned value by the function, callback
	N_ERR_PRTOUTPUT	-13	Returned value by the function, callback
	N_ERR_NOHANDLE	-22	Returned value by the function, callback
	N_ERR_NOTSUPPORTED	-40	Returned value by the function, callback
	N_ERR_ARGUMENT_01	-91	Returned value by the function, callback
	N_ERR_FLAGSQR	-201	Callback
	N_ERR_FLAGSFORMAT	-202	Callback
	N_ERR_FLAGSBUSY	-203	Callback
	N_ERR_FLAGSUNDEFINED	-204	Callback
	N_ERR_FLAGSREJECT	-205	Callback
	N_ERR_FLAGSOTHER	-206	Callback
	N_ERR_MNT_QPRAM	-209	Callback
NBlockSendSetting	N_SUCCESS	0	Returned value by the function, callback
	N_ERR_PRTOUTPUT	-13	Returned value by the function, callback
	N_ERR_NOHANDLE	-22	Returned value by the function, callback
	N_ERR_NOTSUPPORTED	-40	Returned value by the function, callback
	N_ERR_ARGUMENT_01	-91	Returned value by the function, callback
	N_ERR_ARGUMENT_02	-92	Returned value by the function, callback
	N_ERR_FLAGSQR	-201	Callback
	N_ERR_FLAGSFORMAT	-202	Callback
	N_ERR_FLAGSBUSY	-203	Callback
	N_ERR_FLAGSUNDEFINED	-204	Callback
	N_ERR_FLAGSREJECT	-205	Callback
	N_ERR_FLAGSOTHER	-206	Callback
	N_ERR_MNT_QPRAM	-209	Callback

List of returned value for each function (7/7)

* The following functions are not recommended.
When obtaining results by asynchronous functions, getting them by callback functions are recommended.

Function name	Definition name	Defined value	Notes
NOpenResult	N_SUCCESS	0	
	N_WRN_PRTALREADYOPEN	11	
	OPEN_BEFORE	100	
	USBAUTHORITY_MID	101	
	USBAUTHORITY_OK	102	
	USBAUTHORITY_NG	103	
NFirmwareResult	N_SUCCESS	0	
	N_ERR_OFFLINE	-5	
	N_ERR_ARGUMENT_01	-91	
	Status value	Positive values	