

Advanced Printer Driver 6

Status API Manual

Before Use

Describes the information that users need to know before using this product.

APD6 Overview

Provides an overview of APD6.

Using Status API

Explains how to establish a development environment, get ASB statuses, and how to handle the ASB statuses.

Reference for Win32

Describes information of Status API used in Win32 environment.

Reference for .NET

Describes information of Status API used in .NET environment.



Before Use

This chapter describes the information that users need to know before using the EPSON Advanced Printer Driver 6 ("APD6" below).

APD6 Packages

APD6 consists of the following packages.

- **Printer driver package**
These packages are prepared for each TM printer model. Installing the printer driver enables easy printing from software applications. The following manuals are provided.
 - **Install Manual**
This describes APD6 installation, TM printer registration, and how to automatically install the printer driver.
 - **Printer Manual**
This describes the APD6 setting procedures and functions.
 - **Printer Specification**
This describes the printer driver specifications for each TM printer model.
- **Status API package**
This is a special package in APD6 for all TM printers. This must be installed when developing applications that control TM printers using Status API, when monitoring a printer's status in real time, and when APD6 coexists with other Epson drivers. The following manuals are provided.
 - **Status API Manual (this manual)**
This describes how to use Status API to obtain the status of a TM printer from a software application. For the specifications of the APIs available for each TM printer model, see the "Printer Specification" manual contained in the printer driver package.
- **Sample program package**
This is a special package in APD6 for all TM printers. This contains sample programs and sample code for developing applications for printing and control of TM printers. Although no manual is provided, it contains HTML files that describe the programs.

Download

For customers in North America, go to the following web site:

<https://www.epson.com/support/>

For customers in other countries, go to the following web site:

<https://epson.sn>

Meaning of Symbols

**NOTE**

Indicates supplementary explanations and information you should know.

Contents

Before Use	2
■ APD6 Packages	2
Download	2
■ Meaning of Symbols	3
■ Contents	4

APD6 Overview	7
■ APD6 Features	7
Glossary	7
■ Operating Environment	8
Software Requiring Installation of Status API Package	8
Supported TM Printers	8
■ Development Language	9
■ Installing the Status API Package	9
Uninstalling the Package.....	9

Using Status API	10
■ Architecture of the Development Environment	10
■ Types of Status API Functions	13
■ Programming Flow for ASB Status Acquisition	15
Visual C++	15
Visual C# / Visual Basic .NET	17
■ Programming Flow for MICR Control	18
Visual C++	18
Visual C# / Visual Basic .NET	19
■ Status API Errors and Response	20
ASB Status	20
Status API Execution Error.....	23

Reference for Win32	25
■ BiOpenMonPrinter	25
■ BiCloseMonPrinter	27

■ BiLockPrinter	28
■ BiUnlockPrinter	30
■ BiDirectIOEx	31
■ BiResetPrinter	34
■ BiForceResetPrinter	35
■ BiGetType	36
■ BiGetStatus.....	37
■ BiSetStatusBackFunction	38
■ BiSetStatusBackFunctionEx	39
■ BiCancelStatusBack	40
■ BiGetBatteryStatus	41
■ BiSetBatteryStatusBackFunctionEx.....	42
■ BiCancelBatteryStatusBack.....	43
■ BiPowerOff	44
■ BiAutoPowerOffTime.....	45
■ BiGetPrnCapability	46
■ BiOpenDrawer	47
■ BiMICRCleaning.....	49
■ BiMICRGetStatus	50
■ BiMICRLoadCheck.....	51
■ BiMICRReadCheck.....	52
■ BiMICREjectCheck.....	53
■ BiMICRCancelWaitCheckInsertion	54
■ BiMICRRetransmissionCheckData	55
■ BiMICRSelectDataHandling	57
■ BiMICRSetReadBackFunction	59
■ BiMICRSetReadBackFunctionEX	60
■ BiMICRCancelReadBack	62

Reference for .NET.....63

■ Properties	63
IsValid	63
LastError.....	63
Status.....	63
AutoPowerOffTime.....	64

■ Methods	65
OpenMonPrinter.....	65
CloseMonPrinter	65
LockPrinter.....	65
UnlockPrinter	66
DirectIOEx	66
ResetPrinter	67
ForceResetPrinter	67
GetType.....	67
SetStatusBack	68
CancelStatusBack	68
GetBatteryStatus	68
SetBatteryStatusBack.....	68
CancelBatteryStatusBack.....	69
PowerOff.....	69
GetPrnCapability	69
OpenDrawer.....	70
MICRCleaning	70
MICRGetStatus	70
MICRLoadCheck.....	70
MICRReadCheck.....	71
MICREjectCheck.....	71
MICRCancelWaitCheckInsertion	71
MICRRetransmissionCheckData	72
MICRSelectDataHandling	72
MICRSetReadBack	72
MICRCancelReadBack	73
■ Events	74
StatusCallback	74
StatusCallbackEx	74
BatteryStatusCallbackEx	74
MICRReadCallback	75
MICRReadCallbackEx	75

Appendix.....76

■ Acknowledgements	76
■ Cautions	77
■ Trademarks	77

APD6 Overview

APD6 Features

The EPSON Advanced Printer Driver Version 6 ("APD6" below) is a Windows printer driver for Epson TM printers. By installing the Status API package, you can utilize the following functions.

- You can share your TM printer with applications that run other Epson drivers (such as OPOS). Exclusive control of the applications is automatically performed by the driver, which is why the applications do not need to carry it out.
- The TM printer Status API allows you to acquire information about the TM printer model from a particular application, control the TM printer, and acquire information about the status of the TM printer.
- You can also use the TM printer's device fonts to print in .NET environment applications.

**NOTE**

For APD6 and later, the Status API package is distributed individually.

Glossary

Term	Explanation
ASB Status	Auto Status Back: This is a function of the TM printer. This is a status automatically sent from the printer when the printer status changes (opening or closing the cover, out of paper, print completed, etc.).

Operating Environment

See "Install Manual" for the APD6.

Software Requiring Installation of Status API Package

Install the Status API package in the following cases.

- When using the Status API to develop an application that controls the TM printer
- When using one of the following Epson software applications with APD6 on the same computer

Software name	Compatible version
EPSON Advanced Printer Driver 4	Ver.4.56 or later
EPSON Advanced Printer Driver 5	Ver.5.09 or later
EPSON OPOS ADK	Ver.2.68 or later
EPSON OPOS ADK for .NET	Ver.1.11.20 or later
EPSON JavaPOS ADK	Ver.1.11.20 or later
EpsonNet SimpleViewer	Ver.2.30 or later
TM Virtual Port Driver	Ver7.10a or later

Supported TM Printers

These are the TM printers supported by the currently installed APD6. For details, see "TM Printer Specification" included in the currently installed printer driver package.

Development Language

Win32

- Visual C++

.NET

- Visual Basic .NET
- Visual C#

Installing the Status API Package

Follow the procedure described below to install the Status API package.

- 1** Double-click the Status API package installer (APD6_StatusAPI_x.exe) to start the installation process.
- 2** Follow the on-screen instructions to proceed with the installation process.
- 3** If the Status API package has been installed after installation of the printer driver package and registration of the printer, execute printing in order to switch the communication port.

This completes the installation process for the Status API package.

Uninstalling the Package

See "Install Manual" for instructions on how to uninstall the Status API. A Status API package is never uninstalled individually.

Using Status API

In this chapter, the application development environment architecture using Status API and programming method are explained.

Architecture of the Development Environment

The architecture of the application development environment using Status API differs according to the development tool.

Visual C++: [page 10](#)
 Visual Basic .NET: [page 11](#)
 Visual C#: [page 12](#)



NOTE

This describes the methods for Visual Studio 2017.

Visual C++

The following are examples of the development environment architecture using C++.

- 1 Start Microsoft Visual C++ to open Solution Explorer.**
- 2 Copy EpsStmApi.h from the folder installed with APD and paste the file into the operating folder used when developing applications (folder created by the project).**
 Save as file
 32 bit OS: "C:\Programfiles\Epson\Advanced Printer Tool\StatusAPI"
 64 bit OS: "C:\Program Files(x86)\Epson\Advanced Printer Tool\StatusAPI"
 ARM64 OS: "C:\Program Files(x86)\Epson\Advanced Printer Tool\StatusAPI"
- 3 Open the Source File. Define EpsStmApi.h using the #include directive.**
 Definition Methods: ***#include "EpsStmApi.h"***
- 4 The Visual C++ environment is ready for developing an application using Status API.**

Visual Basic .NET

The following is an example for creating the development environment using Visual Basic .NET.

- 1 Start Microsoft Visual Basic .NET to open Solution Explorer.
- 2 Right-click on [References] in Solution Explorer, and select [Add References].



NOTE

If the [References] item does not appear, click the [Show All Files] icon in Solution Explorer.

- 3 The “Add References” screen appears. Click the [Browse] tab.
- 4 Specify followings in [Look in].
 32 bit OS: "C:\Programfiles\Epson\Advanced Printer Tool\StatusAPI"
 64 bit OS: "C:\Program Files(x86)\Epson\Advanced Printer Tool\StatusAPI"
 ARM64 OS: "C:\Program Files(x86)\Epson\Advanced Printer Tool\StatusAPI"
- 5 Type the file name “EpsonStatusAPI.dll”, and click [OK].
- 6 Select [References] - [EpsonStatusAPI] in Solution Explorer, and select “False” for [Specific Version] in Properties.
- 7 Using the Imports statement at the very start of the source code, describe as follows.
Imports com.epson.pos.driver
- 8 The Visual Basic .NET environment is ready for developing an application using Status API.

Visual C#

The following is an example for creating the development environment using Visual C#.

- 1 Start Microsoft Visual C# to open Solution Explorer.
- 2 Right-click on [References] in Solution Explorer, and select [Add References].



NOTE

If the [References] item does not appear, click the [Show All Files] icon in Solution Explorer.

- 3 The “Add References” screen appears. Click the [Browse] tab.
- 4 Specify followings in [Look in].
 32 bit OS: "C:\Programfiles\Epson\Advanced Printer Tool\StatusAPI"
 64 bit OS: "C:\Program Files(x86)\Epson\Advanced Printer Tool\StatusAPI"
 ARM64 OS: "C:\Program Files(x86)\Epson\Advanced Printer Tool\StatusAPI"
- 5 Type the file name “EpsonStatusAPI.dll”, and click [OK].
- 6 Select [References] - [EpsonStatusAPI] in Solution Explorer, and select “False” for [Specific Version] in Properties.
- 7 Using the using keyword at the very start of the source code, describe as follows.
using com.epson.pos.driver
- 8 The Visual C# environment is ready for developing an application using Status API.

Types of Status API Functions

Status API has the following functions. Refer to ["Reference for Win32" on page 25](#) for details regarding the functions.

 NOTE	The supported functions differ according to the printer model. Refer to "Printer Specification" for details regarding each model.
---	--

Application	Function	Description
Starting/Closing Status API	BiOpenMonPrinter	Calls the specified printer to use Status API.
	BiCloseMonPrinter	Closes Status API.
Occupying TM printer	BiLockPrinter	Occupies TM printer. While occupied, the printer accepts no API from other processes.
	BiUnlockPrinter	Cancels BiLockPrinter.
Acquiring ASB Status	BiGetStatus	Acquires the ASB status when required by the application.
	BiSetStatusBackFunction	Provides notification regarding the call of the callback function notifying the application when the ASB status of Status API changes.
	BiSetStatusBackFunctionEx	Provides notification regarding the call of the callback function notifying the application when the ASB status of Status API changes. Also acquires the port number.
	BiCancelStatusBack	Cancels the auto status notification function. This function is applicable to BiSetStatusBackFunction and BiSetStatusBackFunctionEx.
Acquiring Battery Status	BiGetBatteryStatus	Acquires the printer's current battery status when required by the application.
	BiSetBatteryStatusBackFunctionEx	Automatically acquires the printer status (ASB status) using the callback function when the printer status changes. Identifies the printer port originating the callback, in addition to the functions of BiSetStatusBackFunction.
	BiCancelBatteryStatusBack	Cancels the automatic status notification request process called using the BiSetBatteryStatusBackFunctionEx.
Acquiring the printer information	BiGetType	Acquires the TM printer information, such as presence of BM sensor and customer display connection.
	BiGetPrnCapability	Acquires printer information, i.e. firmware, etc.
Drawer control	BiOpenDrawer	Opens the drawer.
Printer reset	BiResetPrinter	Resets the parallel / USB / ethernet interface printers. Cannot reset serial interface printers.
	BiForceResetPrinter	Can reset also the TM printers occupied with BiLockPrinter.

Application	Function	Description
Power off preprocess	BiPowerOff	Sets power-off or standby mode. The following processes are performed: • Sets the interface to BUSY. • Sets the printer to standby mode.
Mobile printer power management	BiAutoPowerOffTime	Acquires and sets the time remaining until the battery-powered printer runs out of power. This is not an API to turn off the printer.
Sends the ESC/POS command	BiDirectIOEx	Can send and receive the ESC/POS commands. Does not add the ASB suppress command.
MICR function control	BiMICRCleaning	Cleans the MICR mechanism.
	BiMICRGetStatus	Acquires the MICR status when required by the application.
	BiMICRLoadCheck	Loads the check to the check print start position.
	BiMICRReadCheck	Executes check reading.
	BiMICREjectCheck	Ejects the check.
	BiMICRCancelWaitCheckInsertion	Cancels check insertion wait.
	BiMICRRetransmissionCheckData	Resends the check reading results.
	BiMICRSelectDataHandling	Selects the check reading operation.
	BiMICRSetReadBackFunction	Executes reading of checks by BiMICRReadCheck and registers the address of the callback function.
	BiMICRSetReadBackFunctionEx	Registers the memory addresses where information read is set.
	BiMICRCancelReadBack	Cancels a reading information notification request that had been registered.

Programming Flow for ASB Status Acquisition

In this section, the obtaining method of ASB status using Status API is explained using the sequence chart.



NOTE

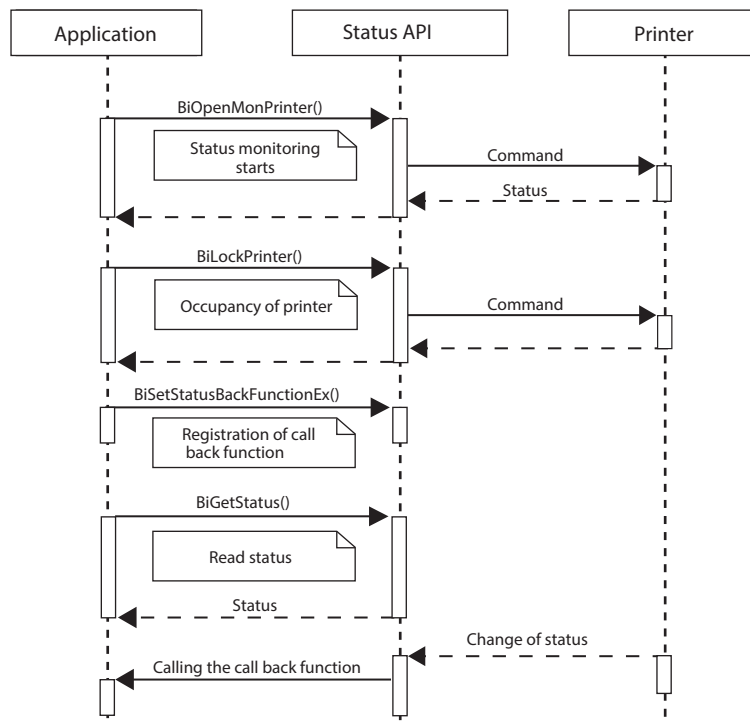
- The DLL load is not included in the sequence chart.
- Acquire the ASB status and confirm that the printer can print before printing.
- Details of the printing flow are not listed.

When Status API (BiOpenMonPrinter) is started, the TM printer automatically sends ASB status as the status changes to Status API. Following API can be used to obtain ASB status that was sent.

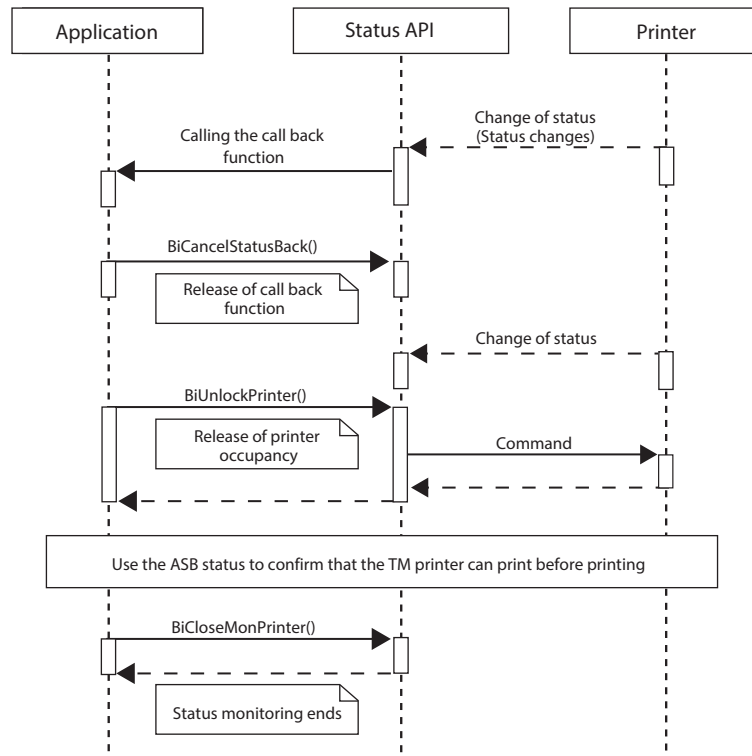
Status API	Description
BiGetStatus	API that obtains ASB status when user (or application) needs.
BiSetStatusBackFunction	API that calls the call back function to send the latest ASB status to the application.
BiSetStatusBackFunctionEx	API that calls the call back function to send the latest ASB status to the application. Called back printer port is also reported.

Visual C++

(1/2)



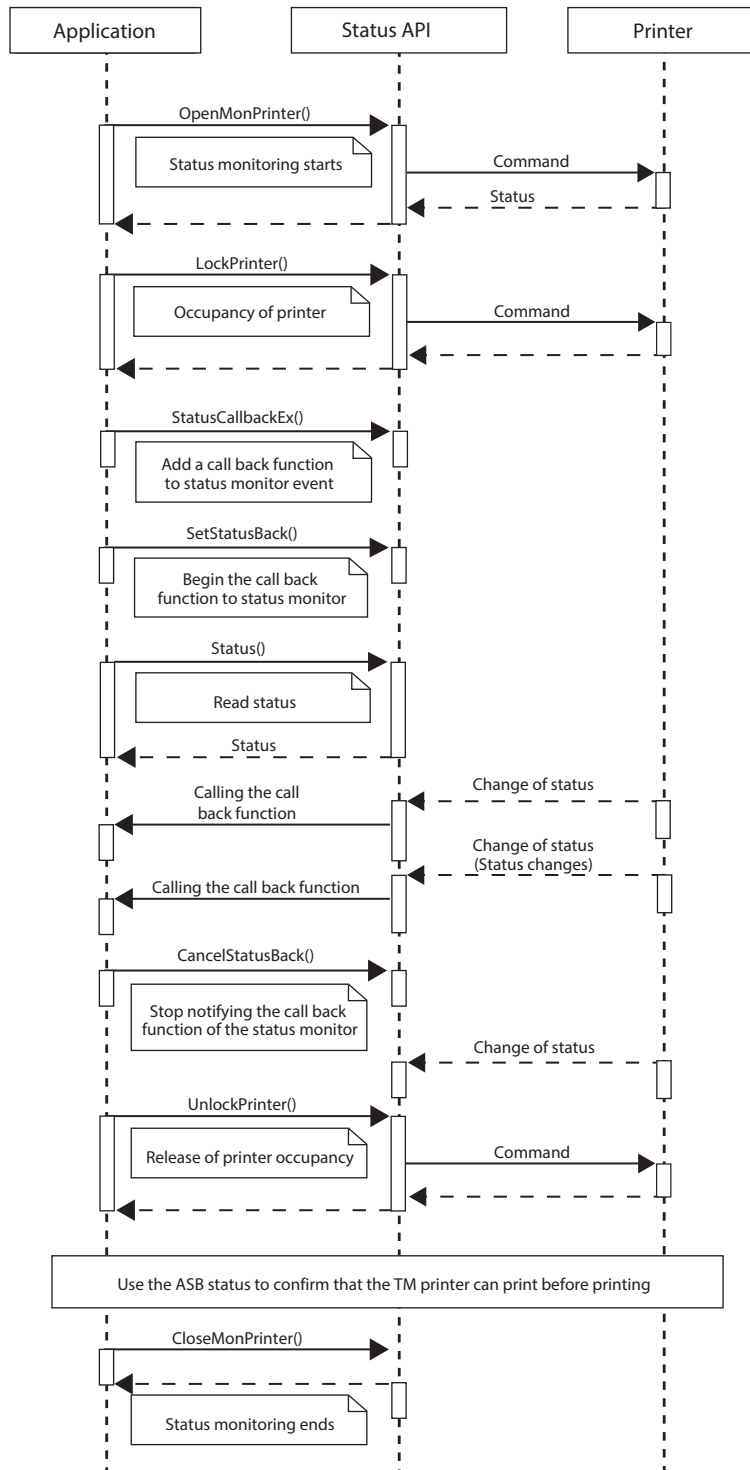
(2/2)



Visual C# / Visual Basic .NET

**NOTE**

When the development environment is Visual C# or Visual Basic .Net, .NET API is used. Refer to ["Reference for .NET" on page 63](#) for .NET API's detail.



Programming Flow for MICR Control

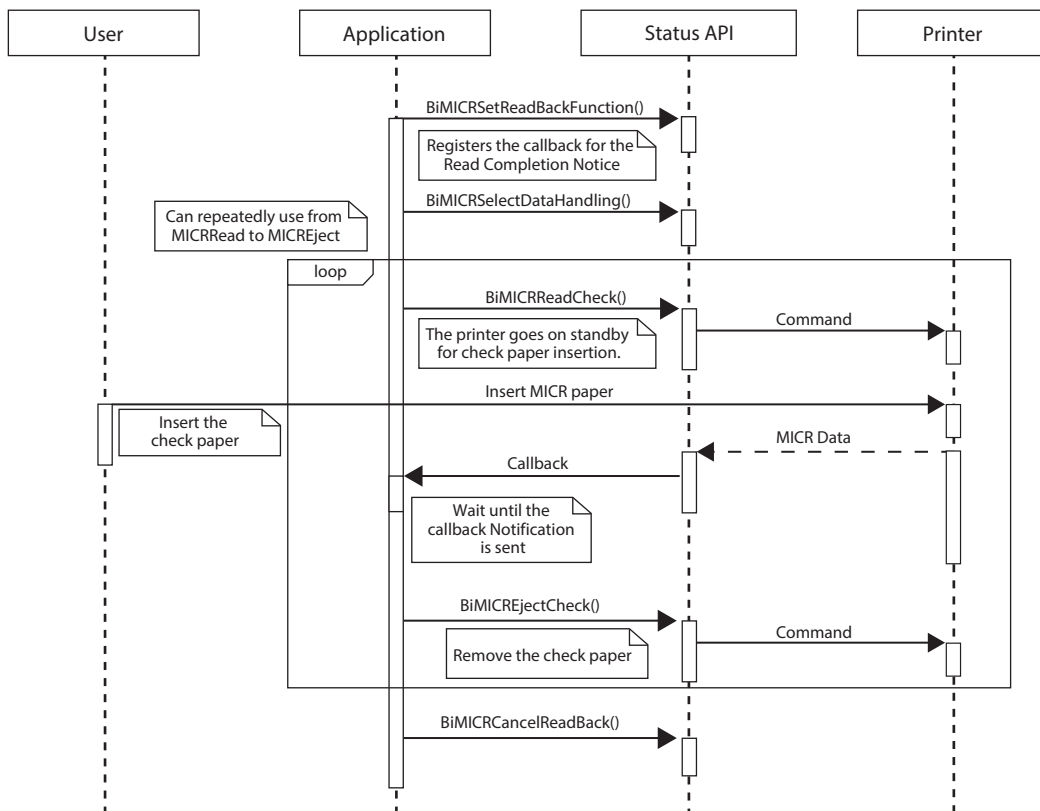
The following explains the control method for MICR control using a sequence figure.



NOTE

The DLL load is not included in the sequence chart.

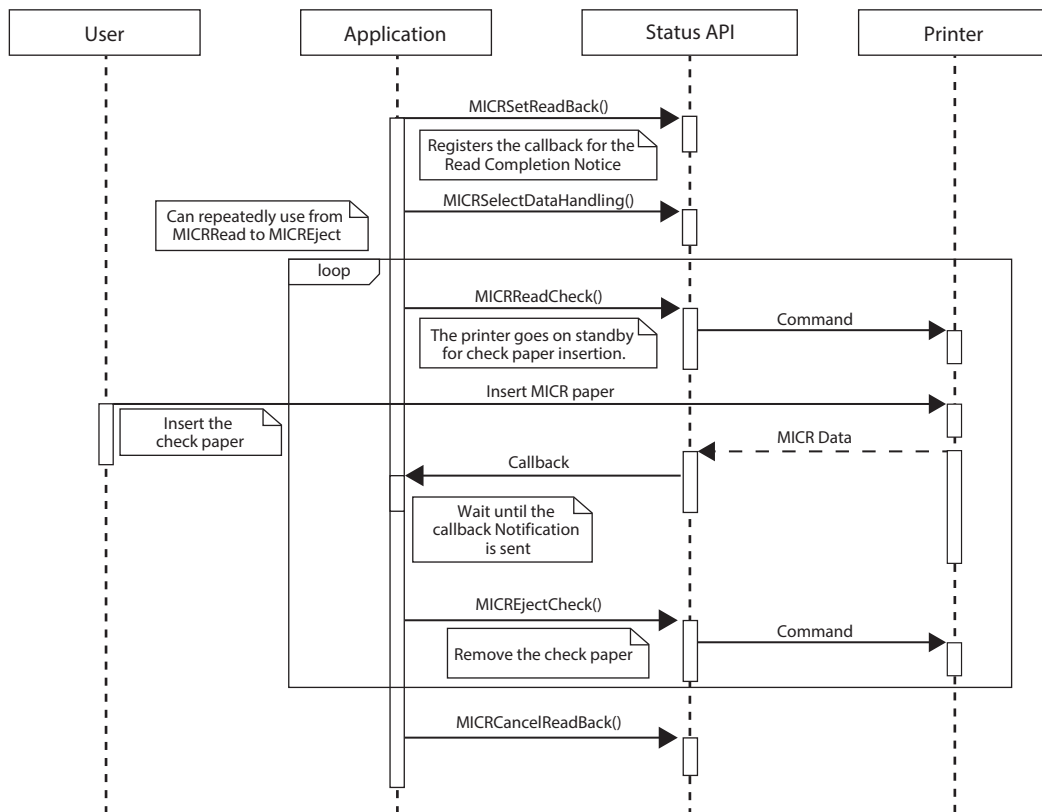
Visual C++



Visual C# / Visual Basic .NET

**NOTE**

When the development environment is Visual C# or Visual Basic .Net, .NET API is used. Refer to ["Reference for .NET" on page 63](#) for .NET API's detail.



Status API Errors and Response

Status API errors are errors when notifying ASB status and errors generated when calling Status API. The following explains the details of the errors and responses. Refer to the following and respond to the application errors.

ASB Status

The following are errors returned when ASB status is acquired. The details differ according to the printer model. Refer to "Printer Specification" for details.

Macro Definition (Constant)	Cause	Response
ASB_NO_RESPONSE	- The power to the printer is not turned ON. - The communication cable is not connected. - The specified printer name/port is different.	Confirm the status and ports of the printer, i.e. cables, etc.
ASB_PRINT_SUCCESS	Notifies that printing has completed successfully. There is nothing else that is notified. Notifies that displaying on the customer display has been completed successfully. There is nothing else to be notified.	-
ASB_DRAWER_KICK	The drawer is open.	There is no problem if the drawer has been left open intentionally.
ASB_BATTERY_OFFLINE	Offline due to low battery level.	Check the battery level.
ASB_OFF_LINE	An error causing the printer to go offline was generated.	Eliminate the cause of the printer to go offline.
ASB_COVER_OPEN	The cover is open.	Close the printer's cover.
ASB_PAPER_FEED	Paper is being fed.	There is no problem if the paper is being fed.
ASB_WAIT_ON_LINE	Waiting for getting back into online.	Check and address the cause of the "Waiting for online recovery" status.
ASB_PANEL_SWITCH	The printer's panel switch is being pressed.	- There is no problem if a switch on the printer's panel is being pressed. - By setting the TM printer's switch or button to not effective (Ex. FEED button), user can use the feed button control as a trigger which can conduct event generation/handling from the application side.
ASB_MECHANICAL_ERR	A mechanical error was generated, i.e. home position detection error, etc.	Eliminate the cause of the error, and turn the TM printer back on or send a TM printer reset command (BiResetPrinter or BiForceResetPrinter).*

Macro Definition (Constant)	Cause	Response
ASB_RECOVER_ERR	A recoverable error has occurred.	Eliminate the cause of the error, and turn the TM printer back on or send a TM printer reset command (BiResetPrinter or BiForceResetPrinter). *
ASB_AUTOCUTTER_ERR	An auto cutter error was generated.	Eliminate the cause of the error, and turn the TM printer back on or send a TM printer reset command (BiResetPrinter or BiForceResetPrinter). *
ASB_UNRECOVER_ERR	A non-recoverable error has occurred in the TM printer.	Immediately turn off the power to the printer. *
ASB_AUTORECOVER_ERR	The temperature of the head has increased.	If the temperature of the head decreases with time, the error automatically cancels. *
ASB_SLIP_INSERT_WAIT	The printer is waiting for slip paper to be inserted.	Place slip paper in the printer.
ASB_RECEIPT_NEAR_END	There is only a limited amount of paper remaining.	Place paper in the printer.
ASB_PAPER_END	There is no roll paper.	Place paper in the printer.
ASB_SLIP_REMOVE_WAIT	The printer is waiting for slip paper to be removed.	Remove the slip paper from the printer.
ASB_RECEIPT_END	No paper.	Place paper in the printer.
ASB_SLIP_TOF	No paper is detected. This is not an error but indicating the printer status.	-
ASB_SLIP_BOF	No paper is detected. This is not an error but indicating the printer status.	-
ASB_SLIP_SELECTED	This indicates whether the slip has been selected.	-
ASB_WAIT_REMOVE_LABEL	Waiting for label removal.	Remove the label from the TM printer.
ASB_PRINT_SLIP	The printer is waiting for the slip to be inserted or the slip is being copied/discharged.	-
ASB_VALIDATION_SELECTED	This indicates whether the validation has been selected.	-
ASB_NO_LABEL	Label/paper has been detected by the label peeling detector/paper removal detector.	Remove the label/paper from the TM printer.
ASB_PRINT_VALIDATION	The printer is waiting for the slip to be inserted or the validation is being copied/discharged.	-
ASB_VALIDATION_TOF	No paper is detected. This is not an error but indicating the printer status.	-
ASB_EJECT_DETECT_NO_PAPER	No paper is detected by the ejection sensor. This is not an error but indicating the printer status.	-
ASB_VALIDATION_NO_PAPER	No paper is detected by the validation detector. This is not an error but indicating the printer status.	-

Macro Definition (Constant)	Cause	Response
ASB_REMOVAL_DELECT_NO_PAPER	Paper remains in the paper ejection slot.	Remove the paper from the TM printer.
ASB_REMOVAL_DETECT_UNKNOWN	Paper is not detected correctly.	Do not expose the paper ejection slot on the TM printer to strong light.
ASB_SPOOLER_IS_STOPPED	The spooler has stopped.	Restart the computer.

* Refer to the detailed operating manuals of the various printers.

 NOTE	The macro definition is defined using the EpsStmApi.h or Module1.bas file when the development environment is constructed.
---	--

Status API Execution Error

The following are errors generated when Status API functions are called. The contents differ according to Status API function. Refer to ["Reference for Win32" on page 25](#) for details.

Macro Definition (Constant)	Cause	Response
ERR_TYPE	The parameters of nType differ.	Specify the correct value.
ERR_OPENED	The specified printer is already opened.	As the printer is already opened, use the handle value or specify a different printer.
ERR_NO_PRINTER	The specified printer driver does not exist.	Confirm the name of the printer driver.
ERR_NO_TARGET	The specified printer cannot be found. An unspecified printer is connected.	Connect to the correct printer.
ERR_NO_MEMORY	There is not enough memory.	Add available memory.
ERR_HANDLE	The handle value specified by the printer is incorrect.	Confirm the handle value.
ERR_TIMEOUT	This is a timeout error.	If the error is continuously generated, confirm whether the printer is properly connected.
ERR_ACCESS	Access cannot be performed on the printer. (The power to the printer is not turned on or the cable is not properly connected, etc.)	Confirm the printer. (Printer power, cable connection, etc.)
ERR_BUFFER_OVER_FLOW	Buffer overflow error.	-
ERR_PARAM	This is a parameter error.	Review the syntax as the specified parameter is incorrect.
ERR_NOT_SUPPORT	This is an unsupported model.	Unsupported models cannot be used.
ERR_EXIST	The specified data already exists.	Delete the already existing data. Example: When this error occurs during executing BiSet-StatusBackXXX, retry it after executing BiCancel-StatusBack.
ERR_EXEC_FUNCTION	This function is unavailable as Status API is used by other applications.	Close the Status API used by other applications.
ERR_PH_NOT_EXIST	The PortHandler is not running, or a communication error between the client of PortHandler and the server.	Verify the connection between them, then restart the computer.
ERR_SPL_NOT_EXIST	The spooler service is not operating.	Confirm if the Print Spooler service has started. (Control Panel - Management Tools - Service)
ERR_RESET	This function is unavailable as the printer is being reset.	Recall after waiting a moment.
ERR_LOCKED	The printer is locked.	Wait until the printer becomes unlocked, or execute BiUnlockPrinter in the program that is locking the printer.
ERR_EXEC_MICR	Cannot call as the MICR is reading	Execute after MICR processing is complete.
ERR_EXEC_SCAN	Cannot call as the scanner is scanning.	Execute after scanning is complete.

**NOTE**

The macro definition is defined using the EPSStmApi.h or StatusAPI.bas file when the development environment is constructed.

Reference for Win32

This chapter explains Status API and the syntax used in Win32 environment.



NOTE

- The data type is described in C++.
- See "Printer Specification" for details regarding the APIs available for each type of TM printer, ASB status, and reasons for the printer to go offline.

BiOpenMonPrinter

Makes Status API available for the printer and returns the handle.

You can open one printer from multiple processes at the same time.

When you open the opened printer from the same process again, a new different handle will return. In such a case, both handles are valid.

Syntax

```
int BiOpenMonPrinter (int nType, LPSTR pName)
```

Parameter

nType: Specifies the pName type.

Macro Definition (Constant)	Value	Description
TYPE_PORT	1	Specify the port name in <i>pName</i> .
TYPE_PRINTER	2	Specify the printer name in <i>pName</i> .

pName: If 1 is specified in nType, specify the port name (example: "ESDPRT001").

If 2 is specified, specify the printer name (example: "EPSON TM-T88V Receipt").

Example)

- Specifying a printer with the port name (ESDPRT001)
nHandle=BiOpenMonPrinter(1,"ESDPRT001");
- Specifying a printer with the printer name (EPSON TM-T88V Receipt)
nHandle=BiOpenMonPrinter(2,"EPSON TM-T88V Receipt");
- Specifying the shared printer(EPSON TM-T88V Receipt) with the host name "SERVER"
nHandle=BiOpenMonPrinter(2,\\SERVER\\”EPSON TM-T88V Receipt”);

Return value

Returns the variable defined in INT type. If Status API is successfully used, the handle identifying the printer is returned. The handle is returned even if the printer is offline. The following Status API execution errors (value) are returned.

Macro Definition (Constant)	Value	Description
ERR_TYPE	-10	Parameter error of <i>nType</i>
ERR_OPENED	-20	The specified printer is already opened.
ERR_NO_PRINTER	-30	The specified printer driver does not exist
ERR_NO_TARGET	-40	Printer unavailable
ERR_NO_MEMORY	-50	Not enough memory
ERR_TIMEOUT	-70	Timeout error
ERR_ACCESS	-80	Access cannot be performed on the printer
ERR_PARAM	-90	Parameter error
ERR_PH_NOT_EXIST	-340	The PortHandler is not running or a communication error between the client of PortHandler and the server.
ERR_SPL_NOT_EXIST	-350	The spooler service is not operating.



NOTE

For information on remedies for the Status API execution errors, refer to ["Status API Execution Error" on page 23](#).

Comment

Call this function before using other Status API functions. The handle of the return value is used as the Parameter by other Status API functions.

The maximum number of printers that can be started at one time is 32.

The following operations are executed according to the printer status when this function is called.

Printer Status	Operation
Online	Returns the handle.
Offline	Returns the handle. However, switch to online as the printer cannot print offline.
Cable Removed/Power Off	Returns "ERR_ACCESS".

BiCloseMonPrinter

Cancels the status monitoring printer.

When a BiOpenMonPrinter function is called, always cancel the status monitoring printer using the BiCloseMonPrinter function.

Syntax

int **BiCloseMonPrinter** (int nHandle)

Parameter

nHandle: Specifies the handle.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Specified handle is invalid
ERR_PH_NOT_EXIST	-340	The PortHandler is not running or a communication error between the client of PortHandler and the server.



NOTE

For information on remedies for the Status API execution errors, refer to "[Status API Execution Error](#)" on page 23.

BiLockPrinter

Locks the printer.



NOTE

- This API is used for a shared printer.
- When using a local printer, this API is used to control multiple processes.

Syntax

int **BiLockPrinter** (int nHandle, DWORD timeout)

Parameter

nHandle: Specifies the handle.

timeout: Specifies the timeout time in ms (milliseconds). Specify it with a positive value.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Specified handle is invalid
ERR_EXEC_FUNCTION	-310	This function is unavailable as Status API is used by other applications.
ERR_EXEC_MICR	-320	Cannot call as the MICR is reading
ERR_PH_NOT_EXIST	-340	The PortHandler is not running or a communication error between the client of PortHandler and the server.
ERR_RESET	-400	Cannot call as the printer is restarting
ERR_LOCKED	-1000	The printer is locked.



NOTE

For information on remedies for the Status API execution errors, refer to ["Status API Execution Error" on page 23](#).

Comment

This API allows you to access the TM printer exclusively. The BiUnlockPrinter API is provided for canceling the exclusive access. While the TM printer is exclusively accessed, the printer does not accept any other API requests of direct access to the printer. The printer will return `ERR_LOCKED` to those other API requests.

The exclusive access right to the TM printer is given to a process. Therefore, the exclusive API access is available from other threads in the same process that is locking the printer.

Executing of API in the same process can be repeated. In this case, the printer is locked with multiple accesses. To unlock the printer, execute BiUnlockPrinter the number of times the API has been executed.

When executing exclusive access from a client to a shared printer or to a local printer via Ethernet, the access status is interrupted if the connection is lost, and restored upon recovery of the connection.

However, while the exclusive access status is being interrupted, another process can lock the TM printer for exclusive access. Once the printer is locked by another process, the printer returns `ERR_LOCKED` to API of the previous process. When another process is finished unlocking the printer, the exclusive access status of the previous process is restored.

Possible causes of the connection failure are as follows.

[Failure during exclusive access to a printer connected via Ethernet]

- The printer is turned Off, or the Ethernet connection between the computer and the printer is disconnected.
- The computer has entered Standby or Hibernate mode.

[Failure during exclusive access to a shared printer from a client]

- The connection between the client and the server is disconnected.
- The client computer has entered Standby or Hibernate mode.

BiUnlockPrinter

Unlocks the lock of the printer.



NOTE

- This API is used for a shared printer.
- When using a local printer, this API is used to control multiple processes.

Syntax

int **BiUnlockPrinter** (int nHandle)

Parameter

nHandle: Specifies the handle.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Specified handle is invalid
ERR_EXEC_FUNCTION	-310	This function is unavailable as Status API is used by other applications.
ERR_EXEC_MICR	-320	Cannot call as the MICR is reading
ERR_PH_NOT_EXIST	-340	The PortHandler is not running or a communication error between the client of PortHandler and the server.
ERR_RESET	-400	Cannot call as the printer is restarting
ERR_LOCKED	-1000	The printer is locked.



NOTE

For information on remedies for the Status API execution errors, refer to "[Status API Execution Error](#)" on page 23.

Comment

This API unlocks the printer locked by "BiLockPrinter". After the lock is canceled, the printer can accept the API from other processes.

If you execute this API when the printer is not locked, "ERR_LOCKED" will be returned to Return value.

BiDirectIOEx

Sends special commands (ESC/POS command) to the printer. Can also acquire command execution results from the printer. The ASB suppress command can be added for differences with BiDirectIO. When the ASB suppress command is added, separate data (ASB status, etc.) is not sent from the printer until this function is complete, therefore, this is recommended when receiving execution results from the printer.



NOTE

- Refer to the following URL for details of the ESC/POS command.
https://support.epson.net/publist/reference_en/

Syntax

int **BiDirectIOEx** (int nHandle, DWORD writeLen, LPBYTE writeCmd, LPDWORD readLen, LPBYTE readBuff, DWORD timeout, BOOL nullTerminate, BYTE option)

Parameter

nHandle:	Specifies the handle.
writeLen:	Specifies the data length to write to the printer. Does not write to the printer when "0".
writeCmd:	Specifies the data (ESC/POS command) to write to the printer.
readLen:	Specifies the data length read from the printer. Specify when the command execution results are required from the printer. Specify as "0" when not required.
readBuff:	Specifies the buffer saving the data read from the printer.
timeout:	Specifies the timeout time in ms (milliseconds).
nullTerminate:	In the case of "True", reading is complete when NULL is received from the printer. At this time, specify the readBuff size to readLen. In the case of "FALSE", the length of data specified in readLen is read or data is read from the printer until a timeout error is generated.
option:	Controls the ASB suppression command.

Value	Description
0	Do not acquire ASB status.
1	Acquire ASB status after acquiring data.



NOTE

- Although the maximum data length that can be specified for readLen/writeLen is 2GB, specify the required minimum data length.
- Ensure that the size of readBuff is the same length specified in readLen or longer.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_NO_MEMORY	-50	Not enough memory
ERR_HANDLE	-60	Specified handle is invalid
ERR_TIMEOUT	-70	Timeout error
ERR_ACCESS	-80	Access cannot be performed on the printer
ERR_PARAM	-90	Parameter error
ERR_BUFFER_OVER_FLOW	-140	Lack of buffer capacity.
ERR_EXEC_FUNCTION	-310	This function is unavailable as Status API is used by other applications.
ERR_EXEC_MICR	-320	Cannot call as the MICR is reading
ERR_PH_NOT_EXIST	-340	The PortHandler is not running or a communication error between the client of PortHandler and the server.
ERR_RESET	-400	Cannot call as the printer is restarting
ERR_LOCKED	-1000	The printer is locked.



NOTE

For information on remedies for the Status API execution errors, refer to "[Status API Execution Error](#)" on page 23.

Comment

Confirm proper execution of the function by confirming the return value of return value or proper command execution by confirming printer operation. If execution results are acquired from the printer (specify readLen), confirm the execution results.

The following operations are executed according to the printer status when this function is called.

Printer Status	Operation
Online	Returns "SUCCESS" to the return value. Executes the command.
Offline	• Returns "SUCCESS" when communication with the printer completed successfully within the time specified with the Timeout. • Returns "ERR_TIMEOUT" when failed to communicate with the printer within the time specified with the Timeout.
Cable Removed/Power Off	Returns "ERR_ACCESS" to the return value.
Printing	Returns "ERR_LOCKED" to the return value.

Caution

- Do not send invalid commands of ASB status transmissions using this function while monitoring the status of the printer. Subsequent status cannot be acquired.
- The ASB (automatic status notification) suppression command ensures that unintended data is not received when sending commands requesting a response from the printer.
If you do not use the ASB suppression, ensure that the programming considers the reception of unintended data.
- Specifying the receiving buffer processes the data received from the printer using this function or processes the data using the same process as the monitoring sled (BiGetStatus function, etc.). Refer to the following.

Transmission Command	Receiving Buffer Specified	Receiving Buffer	Operation of the Monitoring Sled
Acquiring status command	Yes	Saves the ASB status to the receiving buffer	• Does not callback • Does not renew the status
	No	-	• Does not callback • Does not renew the status
Command with responses from other printers	Yes	Enters the printer response into the receiving buffer	Does not effect the monitoring sled
	No	-	Abnormal callbacks may be generated
Command without responses from other printers	Yes	Generates timeout error	Does not effect the monitoring sled
	No	-	Does not effect the monitoring sled

BiResetPrinter

Resets status monitoring printers.



NOTE

- Cancels print jobs when this is called while printing.
- The TM printer with the serial interface cannot be reset.

Syntax

int **BiResetPrinter** (int nHandle)

Parameter

nHandle: Specifies the handle.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Specified handle is invalid
ERR_NOT_SUPPORT	-100	Not supported
ERR_EXEC_FUNCTION	-310	Cannot call as another Status API is in use
ERR_PH_NOT_EXIST	-340	The PortHandler is not running or a communication error between the client of PortHandler and the server.
ERR_RESET	-400	Cannot call as the printer is restarting
ERR_LOCKED	-1000	The printer is locked.



NOTE

For information on remedies for the Status API execution errors, refer to "[Status API Execution Error](#)" on page 23.

Comment

Confirm proper execution of the function by confirming the return value of *return value* or by resetting the printer and confirming that the printer is online (confirming the ASB status).



NOTE

After this function is executed, the printer cannot receive a print command for 15 seconds. If print is executed during this time, the job is sent to the spooler and the print processes is executed after the passage of the aforementioned time.

The following operations are executed according to the printer status when this function is called.

Printer Status	Operation
Online	Returns "SUCCESS" to the return value and resets.
Offline	Returns "SUCCESS" to the return value and resets.
Cable Removed/Power Off	Returns "ASB_NO_RESPONSE" to ASB status and does not reset.
Printing	Cancels the print job and resets.

BiForceResetPrinter

Forces to reset the TM printer whose status is being monitored.

The TM printer can be reset even in multi-thread/multi-process/multi-user environments.

The TM printer can be reset even if BiLockPrinter is accessed by a different program.

If the connection to a network printer is lost and then re-established, some time will be required before printing is possible again. Use of this API allows that time to be shortened.



NOTE

- This API will force the printer to reset even if printing is in progress, and the data being printed will be deleted.
- The TM printer with the serial interface cannot be reset.

Syntax

```
int BiForceResetPrinter (int nHandle)
```

Parameter

nHandle: Specifies the handle.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Specified handle is invalid
ERR_ACCESS	-80	Access cannot be performed on the printer
ERR_NOT_SUPPORT	-100	Not supported
ERR_EXEC_FUNCTION	-310	Cannot call as another Status API is in use
ERR_PH_NOT_EXIST	-340	The PortHandler is not running or a communication error between the client of PortHandler and the server.



NOTE

For information on remedies for the Status API execution errors, refer to "[Status API Execution Error](#)" on page 23.

BiGetType

Acquires the type ID of the printer.



NOTE

For information on the type ID that can be acquired, ask your dealer.

Syntax

int **BiGetType** (int nHandle, LPBYTE typeID, LPBYTE font, LPBYTE exrom, LPBYTE special)

Parameter

nHandle: Specifies the handle.
 typeID: A type ID of the printer will be set.
 font: Device font will be set.
 exrom: This is not applicable.
 special: A special ID of the printer will be set.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Specified handle is invalid
ERR_TIMEOUT	-70	Timeout error
ERR_ACCESS	-80	Access cannot be performed on the printer
ERR_PARAM	-90	Parameter error
ERR_NOT_SUPPORT	-100	Not supported
ERR_EXEC_FUNCTION	-310	This function is unavailable as Status API is used by other applications.
ERR_EXEC_MICR	-320	Cannot call as the MICR is reading
ERR_PH_NOT_EXIST	-340	The PortHandler is not running or a communication error between the client of PortHandler and the server.
ERR_RESET	-400	Cannot call as the printer is restarting
ERR_LOCKED	-1000	The printer is locked.



NOTE

For information on remedies for the Status API execution errors, refer to ["Status API Execution Error" on page 23](#).

Comment

Some information may not be obtained depending on models. In the case, 0 is returned to typeID.

BiGetStatus

Acquires the current printer status (ASB status).

Syntax

int **BiGetStatus** (int nHandle, LPDWORD lpStatus)

Parameter

nHandle: Specifies the handle.
 lpStatus: Returns the ASB status saved to Status API.
 The ASB status is a 4 byte configuration.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Specified handle is invalid
ERR_PARAM	-90	Parameter error
ERR_EXEC_FUNCTION	-310	Cannot call as another Status API is in use



NOTE

For information on remedies for the Status API execution errors, refer to "[Status API Execution Error](#)" on page 23.

Comment

Refer to "Printer Specification" regarding the ASB status that can be acquired by each printer.

BiSetStatusBackFunction

Automatically acquires the printer status (ASB status) using the callback function when the printer status changes.

Syntax

```
int BiSetStatusBackFunction (int nHandle, int (CALLBACK EXPORT *pStatusCB)
                             (DWORD dwStatus))
```

Parameter

nHandle: Specifies the handle.
 *pStatusCB: Specifies the definition address of the callback function.

Parameter of call back function

dwStatus: Returns the ASB status saved to Status API.
 The ASB status is a 4 byte configuration.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Specified handle is invalid
ERR_PARAM	-90	Parameter error
ERR_EXIST	-210	The specified data already exists.
ERR_EXEC_FUNCTION	-310	Cannot call as another Status API is in use



NOTE

For information on remedies for the Status API execution errors, refer to "[Status API Execution Error](#)" on page 23.

Comment

Call this function to set the printer status to *dwStatus* and call the callback function. When the printer status changes, new information is automatically set to *dwStatus* and calls the callback function. Cancel this function using *BiCancelStatusBack*.

Refer to "Printer Specification" regarding the ASB status that can be acquired by each printer.



NOTE

Status API cannot be used within the registered callback function.

BiSetStatusBackFunctionEx

Automatically acquires the printer status (ASB status) using the callback function when the printer status changes.

Identifies the printer port originating the callback, in addition to the functions of BiSetStatusBackFunction.

Syntax

```
int BiSetStatusBackFunctionEx (int nHandle, int (CALLBACK EXPORT *pStatusCB)
                                (DWORD dwStatus, LPSTR lpcPortName))
```

Parameter

nHandle: Specifies the handle.

*pStatusCB: Specifies the definition address of the callback function.

Parameter of call back function

dwStatus: Returns the ASB status saved to Status API. The ASB status is a 4 byte configuration.

lpcPortName: Returns the printer port name originating the callback.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Specified handle is invalid
ERR_PARAM	-90	Parameter error
ERR_EXIST	-210	The specified data already exists.
ERR_EXEC_FUNCTION	-310	Cannot call as another Status API is in use



NOTE

For information on remedies for the Status API execution errors, refer to "[Status API Execution Error](#)" on page 23.

Comment

Call this function to set the printer status to dwStatus and call the callback function. When the printer status changes, new information is automatically set to dwStatus and calls the callback function. Cancel this function using BiCancelStatusBack.

Refer to "Printer Specification" regarding the ASB status that can be acquired by each printer.



NOTE

Status API cannot be used within the registered callback function.

BiCancelStatusBack

Cancels the automatic status notification request process called using the BiSetStatusBackFunction/ BiSetStatusBackFunctionEx function.

Syntax

```
int BiCancelStatusBack (int nHandle)
```

Parameter

nHandle: Specifies the handle.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Specified handle is invalid
ERR_EXEC_FUNCTION	-310	Cannot call as another Status API is in use



NOTE

- Returns "SUCCESS" even when executed when the automatic status notification request process is not registered.
- For information on remedies for the Status API execution errors, refer to "[Status API Execution Error](#)" on page 23.

BiGetBatteryStatus

Acquires the printer's current battery status.

Syntax

```
int BiGetStatus (int nHandle, LPBYTE lpbPowerStatus, LPBYTE lpbBatteryStatus)
```

Parameter

nHandle: Specifies the handle.
 lpbPowerStatus: Specifies the memory address where the power status is set.
 lpbBatteryStatus: Specifies the memory address where the remaining amount of the battery is set.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Specified handle is invalid
ERR_PARAM	-90	Parameter error
ERR_EXEC_FUNCTION	-310	Cannot call as another Status API is in use



NOTE

For information on remedies for the Status API execution errors, refer to ["Status API Execution Error" on page 23](#).

Comment

lpbPowerStatus specifies whether the printer power is supplied from the AC adapter or battery. lpbBatteryStatus specifies the battery level.

Refer to "Printer Specification" regarding the battery status that can be acquired by each printer.

BiSetBatteryStatusBackFunctionEx

Automatically acquires the printer status (ASB status) using the callback function when the printer status changes. Identifies the printer port originating the callback, in addition to the functions of BiSetStatusBackFunction.

Syntax

```
int BiSetBatteryStatusBackFunctionEx
    (int nHandle, int (CALLBACK EXPORT *pBatteryCB)
    (BYTE bPowerStatus, BYTE bBatteryStatus, LPSTR lpcPortName))
```

Parameter

nHandle: Specifies the handle.
 *pBatteryCB: Specifies the definition address of the callback function.

Parameter of call back function

bPowerStatus: Sets the value of the power status for lpbPowerStatus, and then sends the event.
 bBatteryStatus: Sets the battery status value for lpbBatteryStatus, and then sends the event.
 lpcPortName: Returns the printer port name originating the callback.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Specified handle is invalid
ERR_PARAM	-90	Parameter error
ERR_EXIST	-210	The specified data already exists.
ERR_EXEC_FUNCTION	-310	Cannot call as another Status API is in use



NOTE

For information on remedies for the Status API execution errors, refer to ["Status API Execution Error" on page 23](#).

Comment

Call this function to set the printer status to bPowerStatus / bBatteryStatus and call the callback function. When the printer status changes, new information is automatically set to bPowerStatus / bBatteryStatus and calls the callback function. Cancel this function using BiCancelBatteryStatusBack. Refer to "Printer Specification" regarding the battery status that can be acquired by each printer.



NOTE

Status API cannot be used within the registered callback function.

BiCancelBatteryStatusBack

Cancels the automatic status notification request process called using the `BiSetBatteryStatusBackFunctionEx`.

Syntax

int ***BiCancelBatteryStatusBack*** (int nHandle)

Parameter

nHandle: Specifies the handle.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Specified handle is invalid
ERR_EXEC_FUNCTION	-310	Cannot call as another Status API is in use



NOTE

- Returns "SUCCESS" even when executed when the automatic status notification request process is not registered.
- For information on remedies for the Status API execution errors, refer to ["Status API Execution Error" on page 23](#).

BiPowerOff

Turns the TM printer to power-off or standby mode.



NOTE

This API cannot be used when the TM printer is waiting to recover from an offline state. Also, if the TM printer goes offline while this API is active, the API can no longer be used.

Syntax

```
int BiPowerOff(int nHandle)
```

Parameter

nHandle: Specifies the handle.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_NO_MEMORY	-50	Not enough memory
ERR_HANDLE	-60	Specified handle is invalid
ERR_TIMEOUT	-70	Timeout error
ERR_ACCESS	-80	Access cannot be performed on the printer
ERR_NOT_SUPPORT	-100	Not supported
ERR_EXEC_FUNCTION	-310	Cannot call as another Status API is in use
ERR_PH_NOT_EXIST	-340	The PortHandler is not running or a communication error between the client of PortHandler and the server.
ERR_LOCKED	-1000	The printer is locked.



NOTE

For information on remedies for the Status API execution errors, refer to ["Status API Execution Error" on page 23](#).

BiAutoPowerOffTime

Acquires and sets the time until the mobile printer is turned off. Cannot turn off the power to the printer.

Syntax

int **BiAutoPowerOffTime** (int nHandle, BYTE bMode, LPBYTE lpbTime)

Parameter

nHandle: Specifies the handle.

bMode: Specifies the process mode.

Macro Definition (Constant)	Value	Description
EPS_BI_GET	0	Acquires the auto power off time.
EPS_BI_SET	1	Sets the auto power off time.

lpbTime: Specifies the memory address to set the auto power off time.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_NO_MEMORY	-50	Not enough memory
ERR_HANDLE	-60	Specified handle is invalid
ERR_TIMEOUT	-70	Timeout error
ERR_ACCESS	-80	Access cannot be performed on the printer
ERR_NOT_SUPPORT	-100	Not supported
ERR_EXEC_FUNCTION	-310	Cannot call as another Status API is in use
ERR_PH_NOT_EXIST	-340	The PortHandler is not running or a communication error between the client of PortHandler and the server.
ERR_LOCKED	-1000	The printer is locked.



NOTE

For information on remedies for the Status API execution errors, refer to "[Status API Execution Error](#)" on page 23.

Comment

The following items are set in the address specified by lpbTime.

Value	Description
0	Auto power off: disabled
1 - 60	1 to 60 minutes

BiGetPrnCapability

Acquires the specified printer information in printer ID.



NOTE

- For information on the Printer Capability that can be acquired, ask your dealer.
- There are printer ID's that are not supported by printers. If unsupported printer ID's are specified, a timeout error is generated.

Syntax

int **BiGetPrnCapability** (int nHandle, BYTE prnID, LPBYTE pBuffSize, LPBYTE pBuff)

Parameter

- nHandle: Specifies the handle.
- prnID: Specifies the acquiring printer information.
- pBuffSize: Specifies the memory size to set the printer information (1 to 80). Returns the actual read data size after calling this function. In the case of insufficient buffer capacity, the required byte size is returned.
- pBuff: Specifies the memory address to set the printer information.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Specified handle is invalid
ERR_TIMEOUT	-70	Timeout error
ERR_ACCESS	-80	Access cannot be performed on the printer
ERR_PARAM	-90	Parameter error
ERR_BUFFER_OVER_FLOW	-140	Lack of buffer capacity.
ERR_EXEC_FUNCTION	-310	This function is unavailable as Status API is used by other applications.
ERR_EXEC_MICR	-320	Cannot call as the MICR is reading
ERR_PH_NOT_EXIST	-340	The PortHandler is not running or a communication error between the client of PortHandler and the server.
ERR_RESET	-400	Cannot call as the printer is restarting
ERR_LOCKED	-1000	The printer is locked.



NOTE

For information on remedies for the Status API execution errors, refer to ["Status API Execution Error" on page 23](#).

BiOpenDrawer

Opens the drawer.

 NOTE	Opens the drawer even when the TM printer is offline.
---	---

Syntax

int **BiOpenDrawer** (int nHandle, BYTE drawer, BYTE pulse)

Parameter

nHandle: Specifies the handle.

drawer: Specifies the drawer to open.

Macro Definition (Constant)	Value	Description
EPS_BI_DRAWER_1	1	Opens drawer 1
EPS_BI_DRAWER_2	2	Opens drawer 2

pulse: Specifies the time when the drawer kick signal is on.

Macro Definition (Constant)	Value	Description
EPS_BI_PLUSE_100	1	Signal for 100 milliseconds
EPS_BI_PLUSE_200	2	Signal for 200 milliseconds
EPS_BI_PLUSE_300	3	Signal for 300 milliseconds
EPS_BI_PLUSE_400	4	Signal for 400 milliseconds
EPS_BI_PLUSE_500	5	Signal for 500 milliseconds
EPS_BI_PLUSE_600	6	Signal for 600 milliseconds
EPS_BI_PLUSE_700	7	Signal for 700 milliseconds
EPS_BI_PLUSE_800	8	Signal for 800 milliseconds

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Specified handle is invalid
ERR_ACCESS	-80	Access cannot be performed on the printer
ERR_PARAM	-90	Parameter error
ERR_NOT_SUPPORT	-100	Not supported
ERR_EXEC_FUNCTION	-310	This function is unavailable as Status API is used by other applications.
ERR_EXEC_MICR	-320	Cannot call as the MICR is reading
ERR_PH_NOT_EXIST	-340	The PortHandler is not running or a communication error between the client of PortHandler and the server.
ERR_RESET	-400	Cannot call as the printer is restarting
ERR_LOCKED	-1000	The printer is locked.



NOTE

For information on remedies for the Status API execution errors, refer to ["Status API Execution Error" on page 23](#).

Comment

The following operations are executed according to the printer status when this function is called.

Printer Status	Operation
Online	Returns "SUCCESS" to the return value. Opens the drawer.
Offline	Returns "SUCCESS" to the return value. Opens the drawer.
Cable Removed/Power Off	Returns "ERR_ACCESS" to the return value. Does not open the drawer.

BiMICRCleaning

Cleans the MICR mechanism.

Syntax

```
int BiMICRCleaning (int nHandle)
```

Parameter

nHandle: Specifies the handle.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Specified handle is invalid
ERR_TIMEOUT	-70	Timeout error
ERR_ACCESS	-80	Access cannot be performed on the printer
ERR_NOT_SUPPORT	-100	Not supported
ERR_EXEC_FUNCTION	-310	Cannot call as another Status API is in use
ERR_EXEC_MICR	-320	Cannot call as the MICR is reading
ERR_EXEC_SCAN	-330	Cannot call as the scanner is scanning
ERR_PH_NOT_EXIST	-340	The PortHandler is not running or a communication error between the client of PortHandler and the server.
ERR_RESET	-400	Cannot call as the printer is restarting
ERR_LOCKED	-1000	The printer is locked.



NOTE

For information on remedies for the Status API execution errors, refer to ["Status API Execution Error" on page 23](#).

Comment

If this function is called, the mechanism waits for the cleaning sheet to be inserted. Insert the cleaning sheet and carry out cleaning of the mechanism. After cleaning is finished, the MICR function is deselected.

BiMICRGetStatus

Specifies the memory address where the MICR status is set.

Syntax

```
int BiMICRGetStatus (int nHandle, LPBYTE pStatus)
```

Parameter

nHandle: Specifies the handle.

pStatus: Specifies the memory address where the MICR status is set.
See the following MICR states concerning the types of MICR status that are returned.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Specified handle is invalid
ERR_TIMEOUT	-70	Timeout error
ERR_ACCESS	-80	Access cannot be performed on the printer
ERR_PARAM	-90	Parameter error
ERR_NOT_SUPPORT	-100	Not supported
ERR_EXEC_FUNCTION	-310	Cannot call as another Status API is in use
ERR_EXEC_SCAN	-330	Cannot call as the scanner is scanning
ERR_PH_NOT_EXIST	-340	The PortHandler is not running or a communication error between the client of PortHandler and the server.
ERR_RESET	-400	Cannot call as the printer is restarting
ERR_LOCKED	-1000	The printer is locked.



NOTE

For information on remedies for the Status API execution errors, refer to ["Status API Execution Error" on page 23](#).

Comment

For the MICR statuses that can be acquired from each printer, refer to "Printer Specifications".

BiMICRLoadCheck

Loads the check to the check print start position.

Syntax

```
int BiMICRLoadCheck (int nHandle)
```

Parameter

nHandle: Specifies the handle.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Specified handle is invalid
ERR_TIMEOUT	-70	Timeout error
ERR_ACCESS	-80	Access cannot be performed on the printer
ERR_NOT_SUPPORT	-100	Not supported
ERR_EXEC_FUNCTION	-310	Cannot call as another Status API is in use
ERR_EXEC_MICR	-320	Cannot call as the MICR is reading
ERR_EXEC_SCAN	-330	Cannot call as the scanner is scanning
ERR_PH_NOT_EXIST	-340	The PortHandler is not running or a communication error between the client of PortHandler and the server.
ERR_RESET	-400	Cannot call as the printer is restarting
ERR_LOCKED	-1000	The printer is locked.



NOTE

For information on remedies for the Status API execution errors, refer to ["Status API Execution Error" on page 23](#).

BiMICRReadCheck

Executes the MICR reading for checks.

Syntax

int **BiMICRReadCheck** (int nHandle, BYTE readFont, BYTE waitInsertionTime)

Parameter

nHandle: Specifies the handle.
readFont: Specifies the reading font.

Value	Description
0	E13B
1	CMC7

waitInsertionTime: Specifies the check insertion wait time. The printer's default is 0.

Value	Description
0	waitInsertionTime: disabled. Continues the check paper insertion standby status.
1 to 15	1 to 15 minutes

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Specified handle is invalid
ERR_TIMEOUT	-70	Timeout error
ERR_ACCESS	-80	Access cannot be performed on the printer
ERR_PARAM	-90	Parameter error
ERR_NOT_SUPPORT	-100	Not supported
ERR_EXEC_FUNCTION	-310	Cannot call as another Status API is in use
ERR_EXEC_MICR	-320	Cannot call as the MICR is reading
ERR_EXEC_SCAN	-330	Cannot call as the scanner is scanning
ERR_PH_NOT_EXIST	-340	The PortHandler is not running or a communication error between the client of PortHandler and the server.
ERR_RESET	-400	Cannot call as the printer is restarting
ERR_LOCKED	-1000	The printer is locked.



NOTE

For information on remedies for the Status API execution errors, refer to "[Status API Execution Error](#)" on page 23.

Comment

The standby time for check paper insertion after MICR reading is completed is reset to 0, the printer default.

Be sure to call either BiMICRSetReadBackFunction/ BiMICRSetReadBackFunctionEx before calling this function.

BiMICREjectCheck

Ejects the check.

Syntax

```
int BiMICREjectCheck (int nHandle)
```

Parameter

nHandle: Specifies the handle.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Specified handle is invalid
ERR_TIMEOUT	-70	Timeout error
ERR_ACCESS	-80	Access cannot be performed on the printer
ERR_NOT_SUPPORT	-100	Not supported
ERR_EXEC_FUNCTION	-310	Cannot call as another Status API is in use
ERR_EXEC_MICR	-320	Cannot call as the MICR is reading
ERR_EXEC_SCAN	-330	Cannot call as the scanner is scanning
ERR_PH_NOT_EXIST	-340	The PortHandler is not running or a communication error between the client of PortHandler and the server.
ERR_RESET	-400	Cannot call as the printer is restarting
ERR_LOCKED	-1000	The printer is locked.



NOTE

For information on remedies for the Status API execution errors, refer to ["Status API Execution Error" on page 23](#).

BiMICRCancelWaitCheckInsertion

Cancels check insertion wait.

Syntax

int ***BiMICRCancelWaitCheckInsertion*** (int nHandle)

Parameter

nHandle: Specifies the handle.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Specified handle is invalid
ERR_TIMEOUT	-70	Timeout error
ERR_ACCESS	-80	Access cannot be performed on the printer
ERR_NOT_SUPPORT	-100	Not supported
ERR_EXEC_FUNCTION	-310	Cannot call as another Status API is in use
ERR_EXEC_MICR	-320	Cannot call as the MICR is reading
ERR_PH_NOT_EXIST	-340	The PortHandler is not running or a communication error between the client of PortHandler and the server.
ERR_RESET	-400	Cannot call as the printer is restarting
ERR_LOCKED	-1000	The printer is locked.



NOTE

For information on remedies for the Status API execution errors, refer to "[Status API Execution Error](#)" on page 23.

BiMICRRetransmissionCheckData

Resends the check reading results.

Syntax

```
int BiMICRRetransmissionCheckData(int nHandle, LPBYTE pReadBuffSize,
    LPBYTE readCharBuff, LPBYTE pStatus, LPBYTE pDetail, DWORD timeout)
```

Parameter

- nHandle:** Specifies the handle.
- pReadBuffSize:** Specifies the size of memory where the reading data are set. After this function is executed, the size of the actual reading data is set.
- readCharBuff:** Specifies the memory addresses where the reading data are set.
- pStatus:** Specifies a 1-byte memory address where the reading status is set.
- pDetail:** Specifies a 1-byte memory address that sets detailed information after a returned reading error in the case that addition of detailed information is specified by the BiMICRSelectDataHandling function.
- timeout:** Specifies the timeout time for data reading in msec units.
As this parameter is a DWORD type, any negative values specified are handled as positive values.
Example: If you specified -200, in DWORD, it is FFFFFFF38h, and it is handled as 4294967096 msec.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Specified handle is invalid
ERR_TIMEOUT	-70	Timeout error
ERR_ACCESS	-80	Access cannot be performed on the printer
ERR_PARAM	-90	Parameter error
ERR_NOT_SUPPORT	-100	Not supported
ERR_EXEC_FUNCTION	-310	Cannot call as another Status API is in use
ERR_EXEC_MICR	-320	Cannot call as the MICR is reading
ERR_EXEC_SCAN	-330	Cannot call as the scanner is scanning
ERR_PH_NOT_EXIST	-340	The PortHandler is not running or a communication error between the client of PortHandler and the server.
ERR_RESET	-400	Cannot call as the printer is restarting
ERR_LOCKED	-1000	The printer is locked.



NOTE

For information on remedies for the Status API execution errors, refer to "[Status API Execution Error](#)" on page 23.

Comment

If the reading data overflows, the 6th bit of the MICR reading status turns ON (reading data overflow).

If this function is called when check reading is not executed by the BiMICRReadCheck function, the reading data is not set, and the reading status ends with an error.

BiMICRSelectDataHandling

Selects the check reading operation.

Syntax

int **BiMICRSelectDataHandling** (int nHandle, BYTE charSelect, BYTE detailSelect, BYTE errorSelect)

Parameter

nHandle: Specifies the handle.

charSelect: Specifies handling of characters that cannot be analyzed.

Value	Description
0	Interrupts analysis processing at the point when characters that cannot be analyzed are detected and doesn't add the reading data.
1	Replaces characters which cannot be analyzed with a '?' and continues analysis processing, then if the reading data size is at or less than the reading data size specified in BiMICRSetReadBackFunction the reading data are added.

detailSelect: Specifies whether or not to add detailed information after a reading error.

Value	Description
0	Detailed information is not added.
1	Detailed information is added.

errorSelect: Specifies whether or not to end the MICR function or continue it after an error. Furthermore, the MICR function continues regardless of this setting if reading ends normally or when the function ends with an error with detailSelect(1) set.

Value	Description
0	MICR function ends only when the function ends with an error with detailSelect(1) set.
1	If reading ends due to an error caused by any of the following causes, the MICR function continues even after notification of the reading results. • A check with a non-standard length is inserted. • The magnetic waveform cannot be detected. • Characters which cannot be analyzed are detected in analysis processing. • Errors were detected in the noise measurements.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Specified handle is invalid
ERR_TIMEOUT	-70	Timeout error
ERR_ACCESS	-80	Access cannot be performed on the printer
ERR_PARAM	-90	Parameter error
ERR_NOT_SUPPORT	-100	Not supported
ERR_EXEC_FUNCTION	-310	Cannot call as another Status API is in use
ERR_EXEC_MICR	-320	Cannot call as the MICR is reading
ERR_EXEC_SCAN	-330	Cannot call as the scanner is scanning
ERR_PH_NOT_EXIST	-340	The PortHandler is not running or a communication error between the client of PortHandler and the server.
ERR_RESET	-400	Cannot call as the printer is restarting
ERR_LOCKED	-1000	The printer is locked.



NOTE

For information on remedies for the Status API execution errors, refer to ["Status API Execution Error" on page 23](#).

BiMICRSetReadBackFunction

Executes reading of checks by BiMICRReadCheck and registers the address of the callback function when the results are notified as well as the memory addresses where each type of information read from the check is set.

Syntax

```
int BiMICRSetReadBackFunction
(int nHandle, INT (CALLBACK EXPORT *pMicrCB)(VOID),
LPBYTE pReadBuffSize, LPBYTE readCharBuff, LPBYTE pStatus,
LPBYTE pDetail)
```

Parameter

- nHandle: Specifies the handle.
- (CALLBACK EXPORT * pMicrCB)(VOID): Specifies the address of the callback function for notifying the results from reading of a check.
- pReadBuffSize: Specifies the size of the memory where the reading data are set. After execution of this function, the size of the data which were actually read is set.
- readCharBuff: Specifies the memory address where the check reading data are set.
- pStatus: Specifies a 1-byte memory address where the MICR reading status is stored.
- pDetail: Specifies a 1-byte memory address in which reading of a check ends in an error, which is returned in cases where detailed information is added in accordance with the BiMICRSelect-DataHandling function.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Specified handle is invalid
ERR_PARAM	-90	Parameter error
ERR_NOT_SUPPORT	-100	Not supported
ERR_EXIST	-210	The specified data already exists.
ERR_EXEC_FUNCTION	-310	Cannot call as another Status API is in use



NOTE

For information on remedies for the Status API execution errors, refer to ["Status API Execution Error" on page 23](#).

Comment

For the MICR information and MICR reading status of each TM printer, refer to "Printer Specifications".

BiMICRSetReadBackFunctionEX

Executes reading of checks with the BiMICRReadCheck function and registers the address of the callback function to be called when the results are notified, as well as the memory address where each type of the information read is to be set. Also returns the pointer to the port name to recognize from which port the callback is.

Syntax

```
int BiMICRSetReadBackFunctionEx
    (int nHandle, INT (CALLBACK EXPORT *pMicrCB)(LPSTR lpcPortName),
    LPBYTE pReadBuffSize, LPBYTE readCharBuff, LPBYTE pStatus,
    LPBYTE pDetail)
```

Parameter

- nHandle:** Specifies the handle.
- (CALLBACK EXPORT * pMicrCB)(LPSTR lpcPortName):**
Specifies the address of the callback function for notifying the results from reading of a check.
Sets the port name in lpcPortName and calls the callback function.
- pReadBuffSize:** Specifies the size of the memory where the reading data are set. After execution of this function, the size of the data which were actually read is set.
- readCharBuff:** Specifies the memory address where the check reading data are set.
- pStatus:** Specifies a 1-byte memory address where the MICR reading status is stored.
- pDetail:** Specifies a 1-byte memory address in which reading of a check ends in an error, which is returned in cases where detailed information is added in accordance with the BiMICRSelectDataHandling function.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Specified handle is invalid
ERR_PARAM	-90	Parameter error
ERR_NOT_SUPPORT	-100	Not supported
ERR_EXIST	-210	The specified data already exists.
ERR_EXEC_FUNCTION	-310	Cannot call as another Status API is in use



NOTE

For information on remedies for the Status API execution errors, refer to "[Status API Execution Error](#)" on page 23.

Comment

Be sure to call this function before calling the BiMICRReadCheck function. Calling the BiMICRReadCheck function before calling this function returns an error.

If the reading data overflows, the 6th bit of the MICR reading status turns ON (reading data overflow).

For the MICR information and MICR reading status of each TM printer, refer to "Printer Specifications".

BiMICRCancelReadBack

Cancels a reading information notification request registered using `BiMICRSetReadBackFunction` or `BiMICRSetReadBackFunctionEx`.

Syntax

int ***BiMICRCancelReadBack*** (int nHandle)

Parameter

nHandle: Specifies the handle.

Return value

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Specified handle is invalid
ERR_NOT_SUPPORT	-100	Not supported
ERR_EXEC_FUNCTION	-310	Cannot call as another Status API is in use
ERR_EXEC_MICR	-320	Cannot call as the MICR is reading



NOTE

For information on remedies for the Status API execution errors, refer to "[Status API Execution Error](#)" on page 23.

Reference for .NET

This chapter explains Status API and the syntax used in .NET environment.

Properties

IsValid

Acquires the open status of the printer.

Access: Read only

Data type: System.Boolean

Explanation

Returns either of the following values.

true: Successfully opened.

false: Not opened or failed to be opened.

LastError

Acquires the error code of the last executed API.

Access: Read only

Data type: com.epson.pos.driver.ErrorCode

Explanation

Can acquire an error code at any time because this module retains the last executed API.

This method is used to judge success or failure because APIs implemented in properties cannot return error codes.

Error codes for errors that may occur in all APIs. For details, see ["Status API Execution Error" on page 23](#).

Status

Acquires the current printer status.

Access: Read only

Data type: com.epson.pos.driver.ASB

Explanation

Constants defined in com.epson.pos.driver.ASB shall be used for the value. For details, see ["BiGetStatus" on page 37](#).

AutoPowerOffTime

Acquires and sets the time until the mobile printer is turned off. For details, see ["BiAutoPowerOffTime" on page 45](#).

Access: Read/Write

Data type: int

Methods

OpenMonPrinter

Starts controlling the specified printer. For details, see ["BiOpenMonPrinter" on page 25](#).

Syntax

ErrorCode **OpenMonPrinter** (OpenType type, String name)

Parameters

OpenType type:	Type of name to be specified for name. Constants defined in <code>com.epson.pos.driver.OpenType</code> shall be used for the value.
String name:	Starts controlling the specified printer.

CloseMonPrinter

Stops controlling the specified printer. For details, see ["BiCloseMonPrinter" on page 27](#).

Syntax

ErrorCode **CloseMonPrinter** ()

LockPrinter

Occupies the printer. For details, see ["BiLockPrinter" on page 28](#).

Syntax

ErrorCode **LockPrinter** (int timeout)

Parameters

int timeout:	Timeout time (in ms units).
--------------	-----------------------------

UnlockPrinter

Stops occupying the printer. For details, see ["BiUnlockPrinter" on page 30](#).

Syntax

ErrorCode **UnlockPrinter** ()

DirectIOEx

After sending the specified data to the printer, receives data of the specified length from the printer.

For details, see ["BiDirectIOEx" on page 31](#).

Syntax

- ErrorCode **DirectIOEx** (byte[] writeCmd, ref byte[] readBuff, int timeout, bool nullTerminate, byte option)

Description: Sends the ESC/POS commands to the TM printer and receives the execution result (binary data) from the printer.

- ErrorCode **DirectIOEx** (byte[] writeCmd, out String response, int timeout, byte option)

Description: Sends the ESC/POS commands to the TM printer and receives the execution result (character string data) from the printer.

- ErrorCode **DirectIOEx** (byte[] writeCmd, int timeout)

Description: Only sends the ESC/POS commands to the TM printer. Receives neither the execution result nor ASB statuses from the printer.

Parameters

byte[] writeCmd:	Data to be sent to the printer
ref byte[] readBuff:	Data received from the printer
int timeout:	Timeout time for data transmission and reception (in ms units)
bool nullTerminate:	In the case of "True", reading is complete when NULL is received from the printer. At this time, specify the readBuff size to readLen. In the case of "False", the length of data specified in readLen is read or data is read from the printer until a timeout error is generated.
byte option:	Controls the ASB suppression command.
out String response:	Data received from the printer (to be converted into strings)

ResetPrinter

Resets the printer. When resetting the printer during printing, cancels print jobs and performs printer resetting. For details, see ["BiResetPrinter" on page 34](#).

Syntax

ErrorCode **ResetPrinter**()

ForceResetPrinter

Forces to reset the TM printer whose status is being monitored.

Can reset also the TM printers occupied with LockPrinter. This also resets TM printers during printing. Be careful in using this API. For details, see ["BiForceResetPrinter" on page 35](#).

Syntax

ErrorCode **ForceResetPrinter**()

GetType

Acquires the type ID of the printer. For details, see ["BiGetType" on page 36](#).

Syntax

ErrorCode **GetType** (out byte typeid, out byte font,
out byte exrom, out byte euspecial)

Parameters

out byte typeid:	Type ID of the printer
out byte font:	Fonts installed in the printer
out byte exrom:	Capacity of the printer's extended Flash ROM.
out byte euspecial:	Special ID of the printer

SetStatusBack

Starts status notification through StatusCallback/StatusCallbackEx events.

For details, see ["BiSetStatusBackFunctionEx" on page 39](#).

Syntax

ErrorCode ***SetStatusBack*** ()

CancelStatusBack

Stops status notification through StatusCallback/StatusCallbackEx events.

For details, see ["BiCancelStatusBack" on page 40](#).

Syntax

ErrorCode ***CancelStatusBack*** ()

GetBatteryStatus

Acquires the current printer's battery status.

For details, see ["BiGetBatteryStatus" on page 41](#).

Syntax

ErrorCode ***GetBatteryStatus*** (ref PowerStatus lpbPowerStatus,
ref BatteryStatus lpbBatteryStatus)

SetBatteryStatusBack

Starts battery status notification through a BatteryStatusCallbackEx event.

For details, see ["BiSetBatteryStatusBackFunctionEx" on page 42](#).

Syntax

ErrorCode ***SetBatteryStatusBack*** ()

CancelBatteryStatusBack

Stops battery status notification through a BatteryStatusCallbackEx event.

For details, see ["BiCancelBatteryStatusBack" on page 43](#).

Syntax

ErrorCode **CancelBatteryStatusBack** ()

PowerOff

Executes the power-off process of the printer. For details, see ["BiPowerOff" on page 44](#).

Syntax

ErrorCode **PowerOff** ()

GetPrnCapability

Acquires information about the printer specified by the printer ID.

For details, see ["BiGetPrnCapability" on page 46](#).

Syntax

- ErrorCode **GetPrnCapability** (byte printerID, out byte[] data)
Description: Acquires information (binary data) of the TM printer specified with printer ID.
- ErrorCode **GetPrnCapability** (byte printerID, out String data)
Description: Acquires information (character string data) of the TM printer specified with printer ID.

Parameters

byte printerID:	ID of the printer from which information is acquired.
out byte[] data:	Printer information
out String data:	Printer information
	The data type differs depending on the TM printer information (printer ID) to be acquired.

OpenDrawer

Activates the drawer. Can be used also when the printer is offline. For details, see ["BiOpenDrawer" on page 47](#).

Syntax

ErrorCode **OpenDrawer** (Drawer drawer, Pulse pulse)

Parameters

Drawer drawer: Drawer to be opened Constants defined in com.epson.pos.driver.Drawer shall be used for the value.

Pulse pulse: Specifies the time when the drawer kick signal is on.
Constants defined in com.epson.pos.driver.Pulse shall be used for the value.

MICRCleaning

Cleans the MICR mechanism. After cleaning is finished, the MICR function is deselected.

For details, see ["BiMICRCleaning" on page 49](#).

Syntax

ErrorCode **MICRCleaning** ()

MICRGetStatus

Acquires the MICR status.

For details, see ["BiMICRGetStatus" on page 50](#).

Syntax

ErrorCode **MICRGetStatus** (out byte status)

Parameters

out byte status: MICR status

MICRLoadCheck

Loads the check slip into the print start position.

For details, see ["BiMICRLoadCheck" on page 51](#).

Syntax

ErrorCode **MICRLoadCheck** ()

MICRReadCheck

Reads the check slip.

For details, see ["BiMICRReadCheck" on page 52](#).

Syntax

ErrorCode ***MICRReadCheck*** (MicrFont readFont, int waitInsertionTime)

Parameters

MicrFont readFont: Read font

int waitInsertionTime: Check slip insertion wait time

MICREjectCheck

Ejects the check slip.

For details, see ["BiMICREjectCheck" on page 53](#).

Syntax

ErrorCode ***MICREjectCheck*** ()

MICRCancelWaitCheckInsertion

Cancels the check slip insertion wait status.

For details, see ["BiMICRCancelWaitCheckInsertion" on page 54](#).

Syntax

ErrorCode ***MICRCancelWaitCheckInsertion*** ()

MICRRetransmissionCheckData

Executes retransmission of the results of check slip reading.

For details, see ["BiMICRRetransmissionCheckData" on page 55](#).

Syntax

ErrorCode ***MICRRetransmissionCheckData***

(out String response, out MICRStatus status, out MICRDetail detail, int timeout)

Parameters

out String response:	Reading data
out MICRStatus status:	Reading status
out MICRDetail detail:	Reading status detailed information
int timeout:	Timeout time (in ms units)

MICRSelectDataHandling

Configures the operation of check slip reading.

For details, see ["BiMICRSelectDataHandling" on page 57](#).

Syntax

ErrorCode ***MICRSelectDataHandling***

(CharSelect charSel, DetailSelect detailSel, ErrorSelect errorSel)

Parameters

CharSelect charSel:	How to handle characters that cannot be analyzed Constants defined in com.epson.pos.driver.CharSelect shall be used for the value.
DetailSelect detailSel:	Availability of detailed information at the time of abnormal termination of reading Constants defined in com.epson.pos.driver.DetailSelect shall be used for the value.
ErrorSelect errorSel:	Whether or not to end the MICR function at the time of abnormal termination. Regardless of this parameter setting, the MICR function shall continue at the time of normal termination or at the time of abnormal termination adding reading results. Constants defined in com.epson.pos.driver.ErrorSelect shall be used for the value.

MICRSetReadBack

Starts MICR data reception notification through MICRReadCallback/MICRReadCallbackEx events.

For details, see ["BiMICRSetReadBackFunction" on page 59](#), ["BiMICRSetReadBackFunctionEX" on page 60](#).

Syntax

ErrorCode ***MICRSetReadBack*** ()

MICRCancelReadBack

Stops MICR data reception notification through MICRReadCallback/MICRReadCallbackEx events.

For details, see ["BiMICRCancelReadBack" on page 62](#).

Syntax

ErrorCode ***MICRCancelReadBack*** ()

Events

StatusCallback

Event that handles ASB status notification.

For details, see ["BiSetStatusBackFunction" on page 38](#).

Syntax

StatusCallbackHandler (ASB asb)

Parameters

ASB asb: ASB status. For details, see "Printer Specification".

StatusCallbackEx

Event that handles ASB status notification. For details, see ["BiSetStatusBackFunctionEx" on page 39](#).

Syntax

StatusCallbackHandlerEx (ASB asb, String portName)

Parameters

ASB asb: ASB status. For details, see "Printer Specification".

String portName: Port name

BatteryStatusCallbackEx

Event that handles power status notification. For details, see ["BiSetBatteryStatusBackFunctionEx" on page 42](#).

Syntax

BatteryStatusCallbackHandlerEx (byte bPowerStatus, byte bBatteryStatus, String lpcPortName)

Parameters

bPowerStatus: Power status

bBatteryStatus: Battery status

portName: Port name

MICRReadCallback

Event that handles MICR data notification.

For details, see ["BiMICRSetReadBackFunction" on page 59](#).

Syntax

MICRReadCallbackHandler (String data, MICRStatus status, MICRDetail detail)

Parameters

String data:	MICR data
MICRStatus status:	Reading status. Constants defined in com.epson.pos.driver. MICRStatus shall be used for the value.
MICRDetail detail:	Reading status detailed information Constants defined in com.epson.pos.driver.MICRDetail shall be used for the value.

MICRReadCallbackEx

Event that handles MICR data notification.

Syntax

MICRReadCallbackHandlerEx
(String data, MICRStatus status, MICRDetail detail, String portName)

Parameters

String data:	MICR data
MICRStatus status:	Reading status Constants defined in com.epson.pos.driver. MICRStatus shall be used for the value.
MICRDetail detail:	Reading status detailed information Constants defined in com.epson.pos.driver. MICRDetail shall be used for the value.
String portName:	Port name

Appendix

Acknowledgements

Info-ZIP

"Advanced Printer Driver" incorporate compression code from the Info-ZIP group.

This is version 2009-Jan-02 of the Info-ZIP license. The definitive version of this document should be available at <ftp://ftp.info-zip.org/pub/infozip/license.html> indefinitely and a copy at <http://www.info-zip.org/pub/infozip/license.html>.

Copyright (c) 1990-2009 Info-ZIP. All rights reserved.

For the purposes of this copyright and license, "Info-ZIP" is defined as the following set of individuals:

Mark Adler, John Bush, Karl Davis, Harald Denker, Jean-Michel Dubois, Jean-loup Gailly, Hunter Goatley, Ed Gordon, Ian Gorman, Chris Herborth, Dirk Haase, Greg Hartwig, Robert Heath, Jonathan Hudson, Paul Kienitz, David Kirschbaum, Johnny Lee, Onno van der Linden, Igor Mandrichenko, Steve P. Miller, Sergio Monesi, Keith Owens, George Petrov, Greg Roelofs, Kai Uwe Rommel, Steve Salisbury, Dave Smith, Steven M. Schweda, Christian Spieler, Cosmin Truta, Antoine Verheijen, Paul von Behren, Rich Wales, Mike White.

This software is provided "as is," without warranty of any kind, express or implied. In no event shall Info-ZIP or its contributors be held liable for any direct, indirect, incidental, special or consequential damages arising out of the use of or inability to use this software.

Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the above disclaimer and the following restrictions:

1. Redistributions of source code (in whole or in part) must retain the above copyright notice, definition, disclaimer, and this list of conditions.
2. Redistributions in binary form (compiled executables and libraries) must reproduce the above copyright notice, definition, disclaimer, and this list of conditions in documentation and/or other materials provided with the distribution. Additional documentation is not needed for executables where a command line license option provides these and a note regarding this option is in the executable's startup banner. The sole exception to this condition is redistribution of a standard UnZipSFX binary (including SFXWiz) as part of a self-extracting archive; that is permitted without inclusion of this license, as long as the normal SFX banner has not been removed from the binary or disabled.
3. Altered versions—including, but not limited to, ports to new operating systems, existing ports with new graphical interfaces, versions with modified or added functionality, and dynamic, shared, or static library versions not from Info-ZIP—must be plainly marked as such and must not be misrepresented as being the original source or, if binaries, compiled from the original source. Such altered versions also must not be misrepresented as being Info-ZIP releases—including, but not limited to, labeling of the altered versions with the names "Info-ZIP" (or any variation thereof, including, but not limited to, different capitalizations), "Pocket UnZip," "WiZ" or "MacZip" without the explicit permission of Info-ZIP. Such altered versions are further prohibited from misrepresentative use of the Zip-Bugs or Info-ZIP e-mail addresses or the Info-ZIP URL(s), such as to imply Info-ZIP will provide support for the altered versions.
4. Info-ZIP retains the right to use the names "Info-ZIP," "Zip," "UnZip," "UnZipSFX," "WiZ," "Pocket UnZip," "Pocket Zip," and "MacZip" for its own source and binary releases.

Windows Template Library

This application uses the Microsoft Windows Template Library (WTL).

Cautions

- No part of this document may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of Seiko Epson Corporation.
- The contents of this document are subject to change without notice. Please contact us for the latest information.
- While every precaution has been taken in the preparation of this document, Seiko Epson Corporation assumes no responsibility for errors or omissions.
- Neither is any liability assumed for damages resulting from the use of the information contained herein.
- Neither Seiko Epson Corporation nor its affiliates shall be liable to the purchaser of this product or third parties for damages, losses, costs, or expenses incurred by the purchaser or third parties as a result of: accident, misuse, or abuse of this product or unauthorized modifications, repairs, or alterations to this product, or (excluding the U.S.) failure to strictly comply with Seiko Epson Corporation's operating and maintenance instructions.
- Seiko Epson Corporation shall not be liable against any damages or problems arising from the use of any options or any consumable products other than those designated as Original EPSON Products or EPSON Approved Products by Seiko Epson Corporation.

Trademarks

Microsoft[®], Windows[®], Visual Studio[®], Visual C++[®], and Visual C#[®] are trademarks of the Microsoft group of companies.

All other trademarks are the property of their respective owners and used for identification purpose only.

ESC/POS Command System

EPSON ESC/POS is a proprietary POS printer command system that includes patented or patent-pending commands. ESC/POS is compatible with most EPSON POS printers and displays.

ESC/POS is designed to reduce the processing load on the host computer in POS environments. It comprises a set of highly functional and efficient commands and also offers the flexibility to easily make future upgrades.

©Seiko Epson Corporation 2019–2025