EPSON
EXCEED YOUR VISION

Advanced Printer Driver for TM-T81 Ver.4

Status API Manual

Overview

Explains about Status API.

Using Status API

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

Reference for Win32

Describes available Status API and the syntax.

Reference for .NET

Describes information of Status API used in .NET environment.

Generating Log Files

Describes the log function.

Appendix

Describes information of TM-T81 acquired by Status API.

M00003809
Rev.J

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 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

EPSON® and ESC/POS® are registered trademarks of Seiko Epson Corporation in the U.S. and other countries.

MS-DOS®, Microsoft®, Win32®, Windows®, Windows Vista®, Windows Server®, Visual Studio®, Visual Basic®, Visual C++®, and Visual C#® are either registered trademarks or trademarks of Microsoft Corporation in the United States and other countries.

ESC/POS® Command System

EPSON has been taking industry's initiatives with its own POS printer command system (ESC/POS). ESC/POS has a large number of commands including patented ones. Its high scalability enables users to build versatile POS systems. The system is compatible with all types of EPSON POS printers (excluding the TM-C100) and displays. Moreover, its flexibility makes it easy to upgrade the future. The functionality and the user-friendliness is valued around the world.

For Safety

Key to Symbols

The symbols in this manual are identified by their level of importance, as defined below. Read the following carefully before handling the product.

CAUTION	Provides information that must be observed to avoid damage to your equipment or a malfunction.
NOTE	Provides important information and useful tips.

Restriction of Use

When this product is used for applications requiring high reliability/safety such as transportation devices related to aviation, rail, marine, automotive etc.; disaster prevention devices; various safety devices etc; or functional/precision devices etc, you should use this product only after giving consideration to including fail-safes and redundancies into your design to maintain safety and total system reliability. Because this product was not intended for use in applications requiring extremely high reliability/safety such as aerospace equipment, main communication equipment, nuclear power control equipment, or medical equipment related to direct medical care etc, please make your own judgment on this product's suitability after a full evaluation.

About this Manual

Aim of the Manual

This manual is aimed at development engineer and provides necessary information for developing an application using the Status API.

Manual Content

The manual is made up of the following sections:

Chapter 1	Overview
Chapter 2	Using Status API
Chapter 3	Reference for Win32
Chapter 4	Reference for .NET
Chapter 5	Generating Log Files
Appendix	Model Information

Contents

■ **For Safety** **3**
Key to Symbols 3
■ **Restriction of Use** **3**
■ **About this Manual** **4**
Aim of the Manual 4
Manual Content 4
■ **Contents** **5**

Overview **9**
Manual organization 9
■ **Status API Summary** **10**
Status API System 10
Glossary 10
■ **Information that can be Acquired from the TM Printer** **11**
■ **Development Language** **12**

Using Status API **13**
■ **Install and Uninstall** **13**
■ **Architecture of the Development Environment** **13**
■ **Types of Status API Functions** **17**
■ **Acquiring ASB Status** **19**
BiGetStatus 20
BiSetStatusBackFunction 21
BiSetStatusBackWnd 22
■ **Status API Errors and Response** **23**
ASB Status 23
Status API Execution Error 25
■ **How to Use Shared Printers** **27**
Constructing the Exclusive Access 27
When Using APD3.xx Application 28

Reference for Win32.....29

■ Status API used for TM-T81 29
■ BiOpenMonPrinter 30
■ BiCloseMonPrinter 32
■ BiLockPrinter 33
■ BiUnlockPrinter 35
■ BiSetMonInterval 36
■ BiSetMonEtherInterval 37
■ BiDirectIO 38
■ BiDirectIOEx 40
■ BiResetPrinter 44
■ BiForceResetPrinter 46
■ BiCancelError 47
■ BiGetType 49
■ BiGetStatus 50
■ BiGetRealStatus 51
■ BiSetStatusBackFunction 52
■ BiSetStatusBackFunctionEx 54
■ BiSetStatusBackWnd 56
■ BiCancelStatusBack 57
■ BiPowerOff 58
■ BiGetCounter 59
■ BiResetCounter 61
■ BiGetPrnCapability 63
■ BiOpenDrawer 65
■ BiSendDataFile 67
■ BiDirectSendRead 69
■ BiSetDefaultEchoTime 72
■ BiSetEtherEchoTime 73
■ BiSetReadWaitTimeOut 74

Reference for .NET.....75

■ **Properties.....75**

IsValid 75
LastError 75
Status 76

■ **Methods.....77**

OpenMonPrinter 77
CloseMonPrinter 77
LockPrinter 78
UnlockPrinter 78
SetMonInterval 79
SetMonEtherInterval 79
DirectIOEx 80
ResetPrinter 81
ForceResetPrinter 81
CancelError 81
GetType 82
GetRealStatus 82
SetStatusBack 83
CancelStatusBack 83
PowerOff 83
GetCounter 84
ResetCounter 85
GetPrnCapability 85
OpenDrawer 86
SendDataFile 86
DirectSendRead 87
SetDefaultEchoTime 88
SetEtherEchoTime 88

■ **Events.....89**

StatusCallback 89
StatusCallbackEx 89

Generating Log Files.....91

■ **Log Files Settings.....92**

■ **Viewing Log Files 93**

Appendix.....95

■ **Model Information.....95**

TM-T81 95

Overview

Status API is a status monitor API for Epson's TM printers. Advanced functions for monitoring TM printers can be embedded in applications with print functions.

Manual organization

Install Manual

Descriptions of the procedures from installing the APD to performing test print, adding printer drivers, and the silent install which is an automated APD installation.

TM Printer Manual

Descriptions of how to use the APD and its functions.
Descriptions of the specifications.

Status API Manual

This manual. Descriptions of how to get the status of the TM printer from the user application by using the Status API.

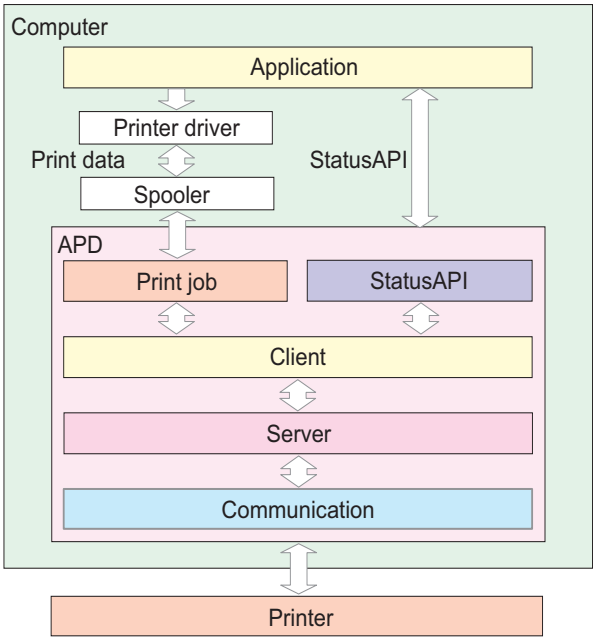
Devmode API / PRINTERINFO Manual

Descriptions of how to configure some printer functions on your application using the Devmode API. Descriptions of the PRINTERINFO Function of Windows.

Status API Summary

Status API System

Status API receives signals from the TM printer and always maintains the most recent status of the printer. The application can acquire the most recent information when necessary.



CAUTION In Terminal Service / Citrix XenApp environment, Status API cannot be used.

Glossary

Term	Explanation
APD	Advanced Printer Driver: Windows printer driver for TM printers. Unlike general Windows printer drivers, Status API is simultaneously installed to monitor the printer status.
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.).
Maintenance Counter	TM printer internal counter recording the operating status of the printer, i.e. operating count of the auto cutter, printer running time, etc.

Information that can be Acquired from the TM Printer

Term	Explanation
ASB Status	Information required for the print application, i.e. print completed, offline, out of paper, cover open, power off, error generated, etc. This information is automatically sent to the Status API.
Maintenance Counter	Acquires information, i.e number of feed paper lines, operating count of the auto cutter, running time, etc. This is used for printer management applications. There are counters that can be reset from Status API and there are integral counters that cannot be reset.

Development Language

Win32

- Visual Basic 6.0
- Visual C++

.NET

- Visual Basic .NET
- Visual C#

CAUTION

.NET Framework Version

Conforming to the APD environment. Refer to "Install Manual".

If you use Status API .NET Wrapper in Windows XP, install .NET Frame Work 2.0 or later before installing APD.

Using Status API

This chapter explains the architecture of the application development environment using Status API, the acquisition methods of ASB Status, and the procedures of the ASB Status. Refer to "Reference for Win32" on page 29 for other functions.

Install and Uninstall

Status API is installed/uninstalled at the same time as APD (Advanced Printer Driver) is installed/uninstalled. Refer to the "Install Manual" for details.

Architecture of the Development Environment

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

Visual Basic

The following are examples of the development environment architecture using Visual Basic.

1. Copy StatusAPI.bas in the folder where the sample program is installed (default is C:\Program Files\EPSON\EPSON Advanced Printer Driver 4 \Sample\US\Src\VB6\SingleFunction\Program09") and paste it into the operating folder used when developing applications.
2. Start Microsoft Visual Basic and open the project screen.
3. Select (Add a standard module) from (Project) in the menu bar.
4. The add standard module screen appears. Select the (Existing file) tab and specify "StatusAPI.bas" copied from the sample program in Procedure 1. Click the (Open) button.
   "StatusAPI.bas" is added the project explorer.
5. Select (Reference settings) from (Project) in the menu bar.
6. The reference settings screen appears. Place a check by "Microsoft DAO 3.6 Object Library" from the (Reference library file) and click the (OK).

Visual C++

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

1. Start Microsoft Visual C++ and open the project screen.

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).

3. Open the Source File. Define EpsStmApi.h using the #include directive.
   Definition Methods: *#include "EpsStmApi.h"*

Visual Basic .NET

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

1. Start Microsoft Visual Studio 2005 and open the Visual Basic .NET project screen.

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 "C:\WINDOWS\assembly" in (Look in).

5. Type the file name following the naming rule shown below, and click (OK).
   "GAC_MSIL\EpsonStatusAPI\(version of EpsonStatusAPI)_(Public Key Token of EpsonStatusAPI)\EpsonStatusAPI.dll"

   Example: GAC_MSIL\EpsonStatusAPI\4.0.9.0__46bb02e1480038cb\EpsonStatusAPI.dll

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 Studio 2005 and open the Visual C# project screen.
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 "C:\WINDOWS\assembly" in (Look in).
5. Type the file name following the naming rule shown below, and click (OK).
   "GAC_MSIL\EpsonStatusAPI\(version of EpsonStatusAPI)_(Public Key Token of EpsonStatusAPI)\EpsonStatusAPI.dll"

   Example: GAC_MSIL\EpsonStatusAPI\4.0.9.0__46bb02e1480038cb\EpsonStatusAPI.dll
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 29 for details regarding the functions. The supported functions differ according to the printer model. Refer to "Reference for Win32" on page 29 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. Occupies TM printer that is used as a shared printer. While occupied, the printer accepts no API from other processes.
	BiUnlockPrinter	Cancels BiLockPrinter.
Acquiring ASB Status	BiGetStatus	Acquires the ASB status from Status API 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.
	BiSetStatusBackWnd	Generates a button click event when the ASB status of Status API changes.
	BiCancelStatusBack	Cancels the auto status notification function. This function is applicable to BiSetStatusBackFunction, BiSetStatusBackFunctionEx, and BiSetStatusBackWnd.
Acquiring and resetting the maintenance counter	BiGetCounter	Acquires the maintenance counter value of the printer.
	BiResetCounter	Resets the maintenance counter of the printer.
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.

Application	Function	Description
Recovery from a recoverable error	BiCancelError	After the cause of the error, i.e. paper jam, etc., is removed, the printer's auto cutter is recovered from the error status using this function. The status is recovered to print standby without turning the printer's power off and on.
Printer reset	BiResetPrinter	Resets the parallel, USB, and ethernet I/F printers. Cannot reset serial I/F printers.
	BiForceResetPrinter	Can reset also the TM printers occupied with BiLockPrinter.
Power off preprocess	BiPowerOff	The printer perform as follows. • Stores the maintenance counter value. • Places the interface in BUSY state. • Places the TM printer in standby mode with power off.
Command definition file	BiSendDataFile	Defines the command definition file created separately to the printer. The command is not executed.
	BiDirectSendRead	Sends the command of the defined command definition file to the printer and executes the command.
Sends the ESC/POS command	BiDirectIO	Can transmit ESC/POS commands to the printer and receive data from the printer.
	BiDirectIOEx	Can send and receive the ESC/POS commands. Does not add the ASB suppress command.

Acquiring ASB Status

The method and function to acquire ASB status from the application are as follows.

Timing	Status API
Acquires when required by the application	BiGetStatus
The ASB status is acquired as follows	BiSetStatusBackFunction
	BiSetStatusBackFunctionEx
	BiSetStatusBackWnd

The ASB status is acquired as follows.

- Confirms whether the printer can print in advance.
- Confirms that printing has completed successfully. Confirms with “ASB_PRINT_SUCCESS” (constant) of the macro definition.
- ASB status monitors the main printer conditions, i.e. out-of-paper, cover open, printer connection status, etc. Therefore, it is recommended that the printer is continually monitored, even when not printing.

Refer to "ASB Status" on page 23 regarding acquired ASB status.

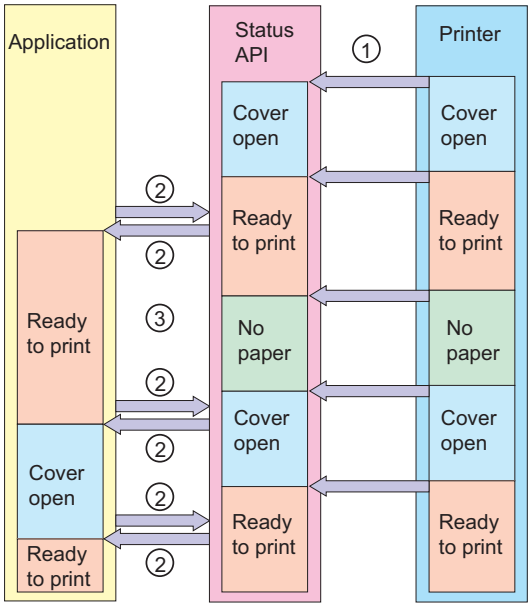
NOTE

Status API has to be started using BiOpenMonPrinter and the printer needs to be opened when acquiring ASB status.

BiGetStatus

BiGetStatus acquires ASB status when required by the user (or application).

Example: The following diagram explains the flow of Status API and ASB status using BiGetStatus.



[1] The printer automatically sends the ASB status to Status API using the ASB function each time the status changes. Status API stores the most recent ASB status.

[2] The application calls BiGetStatus when required by ASB status. Status API sends the stored ASB status to the application.

[3] Status API does not send the ASB status even if the ASB status of the printer changes when there is no request from the application.

Refer to "BiGetStatus" on page 50 regarding the syntax of BiGetStatus.

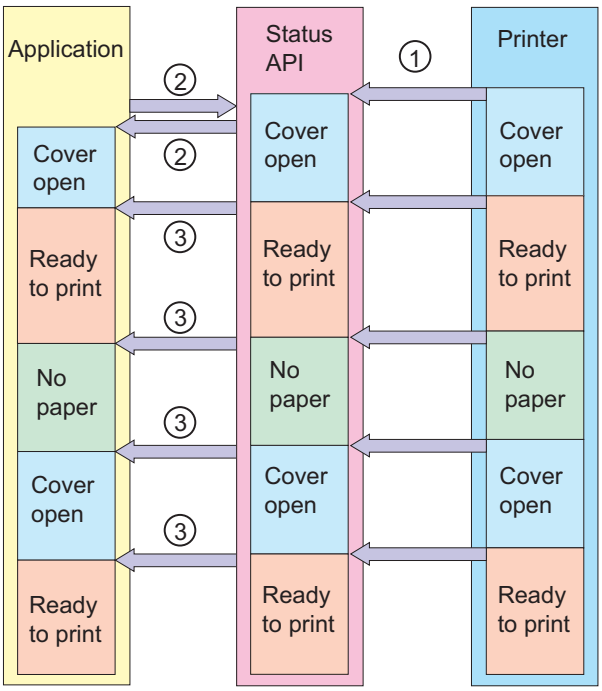
BiSetStatusBackFunction

BiSetStatusBackFunction is an API that automatically allows the application to acquire the most recent ASB status by using the callback function.
Calling BiCancelStatusBack cancels the ASB status notification from Status API using the callback function. Refer to "BiCancelStatusBack" on page 57 for details.

CAUTION This is unavailable when the development environment is Visual Basic.

NOTE BiSetStatusBackFunctionEx can recognize from which printer the callback is, in addition to the function of BiSetStatusBackFunction.

Example: The following diagram explains the flow of Status API and ASB status using BiSetStatusBackFunction.



[1] The printer automatically sends the ASB status to Status API using the ASB function each time the status changes.

[2] BiSetStatusBuckFunction registers the callback function.

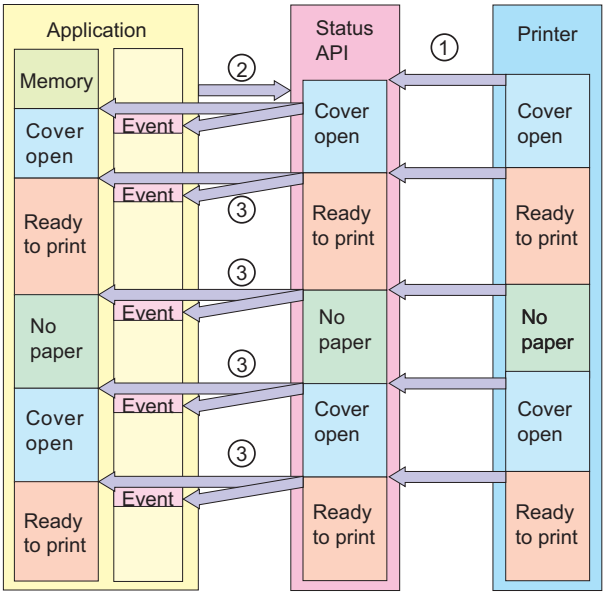
[3] Status API calls the callback function and notifies the application of the most recent ASB status each time the ASB status of the printer changes.

Refer to "BiSetStatusBackFunction" on page 52 regarding the syntax of BiSetStatusBackFunction.

BiSetStatusBackWnd

BiSetStatusBackWnd is an API that acquires the most recent ASB status by registering the window handle of the application screen button and memory address storing the status. Calling BiCancelStatusBack cancels the ASB status notification from Status API using the button click event. Refer to "BiCancelStatusBack" on page 57 for details.

Example: The following diagram explains the flow of Status API and ASB status using BiSetStatusBackWnd.



[1] The printer automatically sends the ASB status to Status API using the ASB function each time the status changes. Status API stores the most recent ASB status.

[2] When BiSetStatusBackWnd registers the window handle of the application screen button and memory address storing the status, Status API sets the data to the specified address and sends a button click event.

[3] Status API sets the most recent ASB status of the printer to the specified memory and sends a button click event each time the ASB status of the printer changes.

Refer to "BiSetStatusBackWnd" on page 56 regarding the syntax of BiSetStatusBackWnd.

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 "Model Information" on page 95 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.	-
ASB_DRAWER_KICK	The drawer is open.	There is no problem if the drawer has been left open intentionally.
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_AUTOCUTTER_ERR	An auto cutter error was generated.	Eliminate the cause of the error and restart the printer or send an error recovery command(BiCancelError).*
ASB_UNRECOVER_ERR	A print error was generated to the printer.	Immediately turn off the power to the printer.*

Macro Definition (Constant)	Cause	Response
ASB_AUTORECOVER_ERR	The temperature of the head has increased.	If the temperature of the head decreases with time, the error automatically cancels.*
ASB_RECEIPT_NEAR_END	There is only a limited amount of paper remaining.	Place paper in the printer.
ASB_RECEIPT_END	No paper.	Place paper in the printer.

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

NOTE

The macro definition is defined using the EPSStmApi.h or StatusAPI.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.

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	R/W 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_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 BiSetStatusBackXXX, retry it after executing BiCancelStatusBack.
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.

Macro Definition (Constant)	Cause	Response
ERR_SPL_NOT_EXIST	The spooler service is not operating.	Confirm whether the Print Spooler is properly operating. (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.

NOTE

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

How to Use Shared Printers

When using shared printers, note the followings when developing an application.

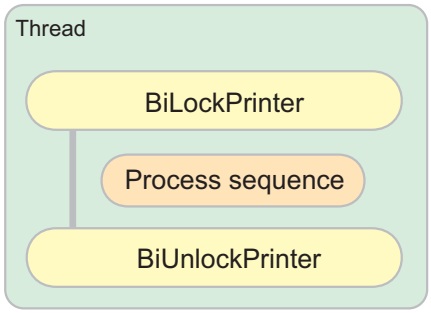
- A whole process of accessing to the device shall be performed exclusively.
- Handling of BiLockPrinter error is necessary.
- Set a time of exclusive access to the printer as short as possible.

> **CAUTION** When you set a firewall, add the port number 2291 to [Exception].

Constructing the Exclusive Access

Exclusive control of a process sequence

Construct an application putting each of process sequences between BiLockPrinter and BiUnlockPrinter.



> **NOTE** When you use only one API in your application, the exclusive access is not necessary.

Handling of BiLockPrinter error

When the printer is already accessed from another process, the BiLockPrinter returns an error (ERR_LOCKED). Construct the application so that the error is handled and the BiLockPrinter is executed again after handling the error.

Shortening exclusive access time to the printer

While the printer is accessed exclusively, other processes cannot execute exclusive API and printing is disabled.
Therefore, set the exclusive access time as short as possible to improve the system performance.

Program example

```
int nRet = BiLockPrinter(1, 1000);
if (nRet == SUCCESS) {
 //Locks the printer to allow exclusive access of the following API. Unlocks the printer when finished.
BiSCNSetImageFormat(1, EPS_BI_SCN_JPEGNORMAL);
BiSCNReadImage(1, 1, EPS_BI_SCN_CHECKPAPER, 0, 0, NULL, EPS_BI_SCN_NVMEMORY_NOTSAVE);
 //Unlocks the printer
BiUnlockPrinter(1);
} else {
 //Error handling in case that the exclusive access is failed.

}
```

When Using APD3.xx Application

The APD3.xx does not support the exclusive access to the printer and consequently your application that supports the APD3.xx do not have the function for the exclusive access . Therefore, when you use the existing application and do not use any other one, the exclusive access to the printer is available without modifying the application.

If any access from other processes is expected, modify the application to establish the exclusive access referring to "Constructing the Exclusive Access" on page 27.

Reference for Win32

This chapter describes the Status API and syntax used for TM-T81. Refer to "Model Information" on page 95 regarding the ASB status for TM-T81, issues causing the printer to go maintenance counters.

> NOTE
> The data type is described in C++.

Status API used for TM-T81

Status API	Page	Status API	Page
BiOpenMonPrinter	30	BiCloseMonPrinter	32
BiLockPrinter	33	BiUnlockPrinter	35
BiSetMonInterval	36	BiSetMonEtherInterval	37
BiDirectIO	38	BiDirectIOEx	40
BiResetPrinter	44	BiForceResetPrinter	46
BiCancelError	47	BiGetType	49
BiGetStatus	50	BiGetRealStatus	51
BiSetStatusBackFunction	52	BiSetStatusBackFunctionEx	54
BiSetStatusBackWnd	56	BiCancelStatusBack	57
BiPowerOff	58	BiGetCounter	59
BiResetCounter	61	BiGetPrnCapability	63
BiOpenDrawer	65	BiSendDataFile	67
BiDirectSendRead	69	BiSetDefaultEchoTime	72
BiSetEtherEchoTime	73	BiSetReadWaitTimeOut	74

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

nErr = **BiOpenMonPrinter** (int nType, LPSTR pName)

Example)

- Make Status API available from the port.
  nHandle = BiOpenMonPrinter(1,"ESDPRT001");
- Make Status API available from the printer.
  nHandle = BiOpenMonPrinter(2, "EPSON TM-T81 Receipt");

Argument

nType: Specifies the pName type. This is an INT type.

Macro Definition (Constant)	Value	Description
TYPE_PORT	1	Specify the port name in *pName*.
TYPE_PRINTER	2	Specify the printer name in *pName*.

pName: If 1 is specified in nType, specify the port name (example: "ESDPRT001").
If 2 is specified, specify the printer name (example: "EPSON TM-T81 Receipt").
This is a LPSTR type.

Return value

Returns the variable (*nHandle*) defined in INT type. If Status API is successfully used, the handle identifying the printer is returned to *nHandle* (correct value). 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 *nType*
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	R/W cannot be performed on the printer

Macro Definition (Constant)	Value	Description
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 25.

Comment

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

NOTE

When this function is called, the specified printer is exclusively available until BiCloseMonPrinter is called. Status API functions from other applications are unavailable during this time.
Acquired handles are only valid within the same application.
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 to *nHandle*.
Offline	Returns the handle to *nHandle*. However, switch to online as the printer cannot print offline.
Cable Removed/Power Off	Returns "ERR_ACCESS" to *nHandle*.

BiCloseMonPrinter

Cancels the status monitoring printer.

> **NOTE** When a BiOpenMonPrinter function is called, always cancel the status monitoring printer using the BiCloseMonPrinter function. An error is generated if a BiOpenMonPrinter function is called again without canceling.

Syntax

nErr = **BiCloseMonPrinter** (*nHandle*)

Argument

nHandle: Specifies the handle. This is an INT type.

Return value

Returns the following Status API execution errors (value) to the variable (nErr) defined by the INT type. Returns "SUCCESS" (macro definition) when this function is successfully called.

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 25.

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

```
nErr = BiLockPrinter (nHandle, timeout)
```

Argument

nHandle: Specifies the handle. This is an INT type.

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

Return value

Returns the following Status API execution errors (value) to the variable (nErr) defined by the INT type. Returns "SUCCESS" (macro definition) when this function is successfully called.

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_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 25.

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

nErr = **BiUnlockPrinter** (*nHandle*)

Argument

nHandle: Specifies the handle. This is an INT type.

Return value

Returns the following Status API execution errors (value) to the variable (nErr) defined by the INT type. Returns "SUCCESS" (macro definition) when this function is successfully called.

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_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 25.

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, “SUCCESS” will be returned to *Return value*.

BiSetMonInterval

Configures the time interval for Status API to read printer status.

> **CAUTION** Setting the interval to a long time overflows the serial receiving buffer and an accurate ASB status cannot be acquired.

Syntax

```
nErr = BiSetMonInterval (nHandle, wNoPrnInterval,
                                  wPrnInterval)
```

Argument

nHandle: Specifies the handle. This is an INT type.

wNoPrnInterval: Not used.

wPrnInterval: Specifies the interval to monitor the status of Status API in milliseconds. This is a WORD type.

Return value

Returns the following Status API execution errors (value) to the variable (nErr) defined by the INT type. Returns "SUCCESS" (macro definition) when this function is successfully called.

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Specified handle is invalid
ERR_PARAM	-90	Parameter error
ERR_EXEC_FUNCTION	-310	This function is unavailable as Status API is used by other applications.
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 25.

Comment

The default value is 100 milliseconds if the interval to monitor the status of Status API is not specified by this API.

> **CAUTION**
> - For interfaces supporting Ethernet, this is done by BiSetMonEtherInterval. See "BiSetMonEtherInterval" on page 37.
> - Before execute this API, execute BiUnlockPrinter.

BiSetMonEtherInterval

Status API configures the read interval of the network printer status.

> **CAUTION** If the interface of your printer is a parallel I/F, serial I/F or USB I/F, this is done by BiSetMonInterval. See "BiSetMonInterval" on page 36.

Syntax

```
nErr = BiSetMonEtherInterval (nHandle, wEtherInterval)
```

Argument

nHandle: Specifies the handle. This is an INT type.

wEtherInterval: Specifies the interval to monitor the status of Status API (1 to 65) in second. This is a WORD type. Even if the value more than 65 seconds is configured, 65 seconds will be actually configured.

Return value

Returns the following Status API execution errors (value) to the variable (nErr) defined by the INT type. Returns "SUCCESS" (macro definition) when this function is successfully called.

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Specified handle is invalid
ERR_ACCESS	-80	R/W 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_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 25.

Comment

The default value is 3 seconds if the interval to monitor the status of Status API is not specified by this API.

> **CAUTION** Before execute this API, execute BiUnlockPrinter.

BiDirectIO

Sends special commands (ESC/POS command) to the printer. Can also acquire command execution results from the printer. It is recommended that BiDirectIOEx is used to acquire execution results.

> NOTE
> Contact the dealer regarding ESC/POS commands.

Syntax

```
nErr = BiDirectIO (nHandle, writeLen, writeCmd, readLen,
                   readBuff, Timeout, nullTerminate)
```

Refer to the next argument.

Argument

nHandle: Specifies the handle. This is an INT type.

writeLen: Specifies the data length to write to the printer. Does not write to the printer when "0".
This is a BYTE type.

writeCmd: Specifies the data (ESC/POS command) to write to the printer. This is a LPBYTE type.

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. This is a LPBYTE type.

readBuff: Specifies the buffer saving the data read from the printer.
This is a LPBYTE type.

Timeout: Specifies the timeout time in ms (milliseconds). This is a DWORD type.

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.

> NOTE
> Ensure that the size of readBuff is the same length specified in readLen or longer.

Return value

Returns the following Status API execution errors (value) to the variable (nErr) defined by the INT type. Returns "SUCCESS" (macro definition) when this function is successfully called.

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Specified handle is invalid
ERR_TIMEOUT	-70	Timeout error
ERR_ACCESS	-80	R/W 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_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 25.

Comment

Confirm proper execution of the function by confirming the return value of *nErr* 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 *nErr*. 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 *nErr*.
Printing	Returns "ERR_LOCKED" to *nErr*.

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**
>
> When considering expandability and versatility, it is recommended to use BiDirectIOEx rather than using the BiDirectIO function.
> Contact the dealer regarding ESC/POS commands.

Syntax

```
nErr = BiDirectIOEx (nHandle, writeLen, writeCmd,
                     readLen, readBuff, Timeout,
                     nullTerminate, option )
```

Refer to the next argument.

Argument

nHandle: Specifies the handle. This is an INT type.

writeLen: Specifies the data length to write to the printer. Does not write to the printer when "0". This is a DWORD type.

writeCmd: Specifies the data (ESC/POS command) to write to the printer. This is a LPBYTE type.

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. This is a LPDWORD type.

readBuff: Specifies the buffer saving the data read from the printer.
This is a LPBYTE type.

Timeout: Specifies the timeout time in ms (milliseconds). This is a DWORD type.

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. This is a BYTE type.

Value	Description
0	Send an ASB suppression command before writing data and enable ASB after reading the data.
1	Does not send the ASB suppression command or ASB enable command.

NOTE

Ensure that the size of readBuff is the same length specified in readLen or longer.

Return value

Returns the following Status API execution errors (value) to the variable (nErr) defined by the INT type. Returns "SUCCESS" (macro definition) when this function is successfully called.

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	R/W 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_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 25.

Comment

Confirm proper execution of the function by confirming the return value of *nErr* 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 *nErr*. Executes the command.
Offline	• Returns "SUCCESS" when communication with the printer completed successfully within the time specified with the Time-out. • 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 *nErr*.
Printing	Returns "ERR_LOCKED" to *nErr*.

Caution

- Although the maximum data length that can be specified for Read/Write is 2GB, specify the required minimum data length.
- 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
Presenter Command	Yes	Saves the presenter response to the receiving buffer	Does not callback
	No	-	Calls back the presenter
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.

CAUTION

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

Syntax

nErr = **BiResetPrinter** (*nHandle*)

Specify the handle in *nHandle*.

Argument

nHandle: Specifies the handle. This is an INT type.

Return value

Returns the following Status API execution errors (value) to the variable (nErr) defined by the INT type. Returns "SUCCESS" (macro definition) when this function is successfully called.

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 25.

Comment

Confirm proper execution of the function by confirming the return value of *nErr* 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 *nErr* and resets.
Offline	Returns "SUCCESS" to *nErr* 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/process/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.

CAUTION

- 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

nErr = **BiForceResetPrinter** (*nHandle*)

Specify the handle in *nHandle*.

Argument

nHandle: Specifies the handle. This is an INT type.

Return value

Returns the following Status API execution errors (value) to the variable (nErr) defined by the INT type. Returns "SUCCESS" (macro definition) when this function is successfully called.

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Specified handle is invalid
ERR_ACCESS	-80	R/W 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 25.

BiCancelError

If a printer recoverable error is generated, execute this function after removing the error cause, and the TM printer recovers from the error.

> **NOTE** If a printer recoverable error is generated while transmitting data, recovery may not be possible with this function. In this case, use BiResetPrinter after resolving the error cause and recover from the error.

Syntax

nErr = **BiCancelError** (*nHandle*)

Specify the handle in *nHandle*.

Argument

nHandle: Specifies the handle. This is an INT type.

Return value

Returns the following Status API execution errors (value) to the variable (nErr) defined by the INT type. Returns "SUCCESS" (macro definition) when this function is successfully called.

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Specified handle is invalid
ERR_ACCESS	-80	R/W cannot be performed on the printer
ERR_NOT_SUPPORT	-100	Not supported
ERR_EXEC_FUNCTION	-310	This function is unavailable as Status API is used by other applications.
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 25.

Comment

Use BiCancelError as follows.

Error	Response
Cover open error	Call this function after closing the cover.
Auto cutter error	Call this function after removing any paper around the cutter and closing the cover.

BiGetType

Acquires the type ID of the printer.

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

Syntax

```
nErr = BiGetType (nHandle, typeID, font, exrom, special)
```

If you specify a handle to *nHandle*, a type ID is set to *typeID*, Device font is set to *font*. A special ID of the printer returns to *special*.

Argument

nHandle: Specifies the handle. This is an INT type.

typeID: A type ID of the printer will be set. This is a LPBYTE type.

font: Device font will be set. This is a LPBYTE type.

exrom: This is not applicable. This is a LPBYTE type.

special: A special ID of the printer will be set. This is a LPBYTE type.

Return value

Returns the following Status API execution errors (value) to the variable (nErr) defined by the INT type. Returns "SUCCESS" (macro definition) when this function is successfully called.

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Specified handle is invalid
ERR_TIMEOUT	-70	Timeout error
ERR_ACCESS	-80	R/W 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_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 25.

BiGetStatus

Acquires the current printer status (ASB status).

Syntax

nErr = **BiGetStatus** (*nHandle, lpStatus*)

Specify the handle in *nHandle*. Returns ASB status to *lpStatus*.

Argument

nHandle: Specifies the handle. This is an INT type.

lpStatus: Returns the ASB status saved to Status API. This is a LPDWORD type. The ASB status is a 4 byte configuration.

Return value

Returns the following Status API execution errors (value) to the variable (nErr) defined by the INT type. Returns "SUCCESS" (macro definition) when this function is successfully called.

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 25.

Comment

Refer to "Model Information" on page 95 regarding the ASB status that can be acquired by TM-T81.

BiGetRealStatus

Acquires the current printer status (ASB status).

Syntax

nErr = **BiGetRealStatus** (*nHandle*, *lpStatus*)

Argument

nHandle: Specifies the handle. This is an INT type.

lpStatus: Returns the ASB status saved to Status API. This is a LPDWORD type. The ASB status is a 4 byte configuration.

Return value

Returns the following Status API execution errors (value) to the variable (nErr) defined by the INT type. Returns "SUCCESS" (macro definition) when this function is successfully called.

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Specified handle is invalid
ERR_ACCESS	-80	R/W cannot be performed on the printer
ERR_PARAM	-90	Parameter error
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 25.

Comment

This function sends the command to acquire the ASB status to the printer and receives the acquired status after the function is called. That is why, even when printing has been completed, ASB_PRINTSUCCESS is not acquired. Also, when the power is turned off, ASB_NO_RESPONSE is not acquired because ERR_ACCESS is returned.

Refer to "Model Information" on page 95 regarding the ASB status that can be acquired by TM-T81.

BiSetStatusBackFunction

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

> **NOTE** This is unavailable when the development environment is VB.

Syntax

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

Argument

nHandle: Specifies the handle. This is an INT type.

int (CALLBACK EXPORT *pStatusCB)(DWORD dwStatus):
Specifies the definition address of the callback function.

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

Return value

Returns the following Status API execution errors (value) to the variable (nErr) defined by the INT type. Returns "SUCCESS" (macro definition) when this function is successfully called.

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 25.

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 "Model Information" on page 95 regarding the ASB status that can be acquired by TM-T81.

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.

> NOTE
> This is unavailable when the development environment is VB.

Syntax

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

Argument

nHandle: Specifies the handle. This is an INT type.

int (CALLBACK EXPORT *pStatusCB)(DWORD dwStatus, LPSTR lpcPortName):
Specifies the definition address of the callback function.

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

lpcPortName: Returns the printer port name originating the callback. This is a LPSTR type.

Return value

Returns the following Status API execution errors (value) to the variable (nErr) defined by the INT type. Returns "SUCCESS" (macro definition) when this function is successfully called.

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 25.

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 "Model Information" on page 95 regarding the ASB status that can be acquired by TM-T81.

NOTE	Status API cannot be used within the registered callback function.

BiSetStatusBackWnd

Automatically generates a click event and acquires the printer status (ASB status) when the printer status changes.

Syntax

nErr = **BiSetStatusBackWnd** (*nHandle*, *hWnd*, *lpStatus*)

Specify the handle in *nHandle*. Returns ASB status to *lpStatus*.

Argument

nHandle: Specifies the handle. This is an INT type.

hWnd: Specifies the window handle of the button generating the click event. This is a Long type.

lpStatus: Returns the ASB status saved to Status API. This is a LPDWORD type. The ASB status is a 4 byte configuration.

Return value

Returns the following Status API execution errors (value) to the variable (nErr) defined by the INT type. Returns "SUCCESS" (macro definition) when this function is successfully called.

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 25.

Comment

Refer to "Model Information" on page 95 regarding the ASB status that can be acquired by TM-T81.

NOTE

Status API cannot be used from the specified window handle.

BiCancelStatusBack

Cancels the automatic status notification request process called using the BiSetStatusBackFunction, BiSetStatusBackFunctionEx, or BiSetStatusBackWnd function.

Syntax

nErr = **BiCancelStatusBack** (*nHandle*)

Specify the handle in *nHandle*.

Argument

nHandle: Specifies the handle. This is an INT type.

Return value

Returns the following Status API execution errors (value) to the variable (nErr) defined by the INT type. Returns "SUCCESS" (macro definition) when this function is successfully called.

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 25.

BiPowerOff

Updates the maintenance counter and prepares to turn off the power to the printer.
Cannot turn off the power to the printer.

> **NOTE** Cannot call when in online recovery standby.

Syntax

nErr = **BiPowerOff** (*nHandle*)

Specify the handle in *nHandle*.

Argument

nHandle: Specifies the handle. This is an INT type.

Return value

Returns the following Status API execution errors (value) to the variable (nErr) defined by the INT type. Returns "SUCCESS" (macro definition) when this function is successfully called.

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	R/W 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 25.

Comment

The TM printer perform as follows.

- Stores the maintenance counter value.
- Places the interface in BUSY state.
- Places the TM printer in standby mode with power off.

BiGetCounter

Acquires the maintenance counter value.

NOTE

- For information on the counter number and the maintenance counters that can be acquired, refer to "Model Information" on page 95.
- The maintenance counter may not be available according to the printer. In this case, a timeout error is generated.
- Confirm that the ASB status is online before calling this function.

Syntax

nErr = **BiGetCounter** (*nHandle, readno, readcounter*)

Specify the handle in *nHandle*. *readno* specifies the acquired maintenance counter number and the maintenance counter value is returned to *readcounter*.

Argument

nHandle: Specifies the handle. This is an INT type.

readno: Specifies the acquired maintenance counter number. This is a WORD type.

readcounter: Returns the maintenance counter. This is a LPDWORD type.

Return value

Returns the following Status API execution errors (value) to the variable (nErr) defined by the INT type. Returns "SUCCESS" (macro definition) when this function is successfully called.

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Specified handle is invalid
ERR_TIMEOUT	-70	Timeout error
ERR_ACCESS	-80	R/W 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_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 25.

Comment

There are two types of maintenance counters; those that can be reset by the user, and integrated counters that cannot be reset.

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

Printer Status	Operation
Online	Returns "SUCCESS" to *nErr*. Acquires the maintenance counter value.
Offline	Returns "ERR_TIMEOUT" to *nErr*. Does not acquire the maintenance counter value.
Cable Removed/ Power Off	Returns "ERR_ACCESS" to *nErr*. Does not acquire the maintenance counter value.
Printing	Returns "ERR_ACCESS" to *nErr*. Does not acquire the maintenance counter value.

BiResetCounter

Resets the maintenance counter.

NOTE

- For information on the counter number and the maintenance counters that can be reset, refer to "Model Information" on page 95.
- The maintenance counter may not be available according to the printer. In this case, a timeout error is generated.
- Confirm that the ASB status is online before calling this function.

Syntax

```
nErr = BiResetCounter (nHandle, readno)
```

Specify the handle in *nHandle*. Specify the maintenance counter number reset to *readno*.

Argument

nHandle: Specifies the handle. This is an INT type.

readno: Specifies the maintenance counter number to be reset. This is a WORD type.

Return value

Returns the following Status API execution errors (value) to the variable (nErr) defined by the INT type. Returns "SUCCESS" (macro definition) when this function is successfully called.

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Specified handle is invalid
ERR_TIMEOUT	-70	Timeout error
ERR_ACCESS	-80	R/W 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_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 25.

Comment

If "SUCCESS" is the return value of *nErr* and the value acquired by BiGetCounter after resetting the maintenance counter, this confirms normal execution.

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

Printer Status	Operation
Online	Returns "SUCCESS" to *nErr*. Resets the maintenance counter.
Offline	Returns "ERR_TIMEOUT" to *nErr*. Does not reset the maintenance counter.
Cable Removed/Power Off	Returns "ERR_ACCESS" to *nErr*. Does not reset the maintenance counter.
Printing	Returns "ERR_ACCESS" to *nErr*. Does not reset the maintenance counter.

BiGetPrnCapability

Acquires the specified printer information in printer ID.

> NOTE For information on the Printer Capability that can be acquired, ask your dealer.

Syntax

```
nErr = BiGetPrnCapability (nHandle, prnID, pBuffSize,
                                                pBuff)
```

Specify the handle in *nHandle* and specify the acquiring printer information to *prnID*. Specify the memory size to set the printer information in *pBuffSize* and specify the memory address to set the printer information in *pBuff*.

Argument

nHandle: Specifies the handle. This is an INT type.

prnID: Specifies the acquiring printer information. This is a BYTE type.

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. This is a LPBYTE type.

pBuff: Specifies the memory address to set the printer information. This is a LPBYTE type.

Return value

Returns the following Status API execution errors (value) to the variable (nErr) defined by the INT type. Returns "SUCCESS" (macro definition) when this function is successfully called.

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Specified handle is invalid
ERR_TIMEOUT	-70	Timeout error
ERR_ACCESS	-80	R/W 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_PH_NOT_EXIST	-340	The PortHandler is not running or a communication error between the client of PortHandler and the server.

Macro Definition (Constant)	Value	Description
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 25.

BiOpenDrawer

Opens the drawer.

> NOTE
> Opens the drawer even when the printer is offline.

Syntax

nErr = **BiOpenDrawer** (*nHandle, drawer, pulse*)

Specify the handle in *nHandle*. Specify the drawer to open in *drawer* and specify the time until the drawer opens in *pulse*.

Argument

nHandle: Specifies the handle. This is an INT type.

drawer: Specifies the drawer to open. This is a BYTE type.

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 until the drawer is opened. This is a BYTE type.

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

Return value

Returns the following Status API execution errors (value) to the variable (nErr) defined by the INT type. Returns "SUCCESS" (macro definition) when this function is successfully called.

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Specified handle is invalid
ERR_ACCESS	-80	R/W 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_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 25.

Comment

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

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

BiSendDataFile

Specify the command definition file to define the transmission command (ESC/POS command).

> **NOTE**
> Specify the command definition file in the specified format.
> Contact the dealer regarding ESC/POS commands.

Syntax

nErr = **BiSendDataFile** (*nHandle, lpcFileName*)

Specify the handle in *nHandle* and the command definition file name in *lpcFileName*.

Argument

nHandle: Specifies the handle. This is an INT type.

lpcFileName: Specifies the command definition file name. This is a LPCSTR type.

Return value

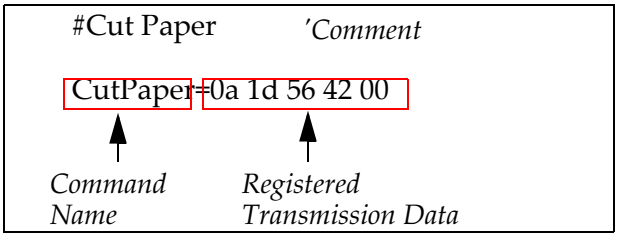
Returns the following Status API execution errors (value) to the variable (nErr) defined by the INT type. Returns "SUCCESS" (macro definition) when this function is successfully called.

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_NO_MEMORY	-50	Not enough memory
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 25.

Comment

Describe the command definition file using the following format.



Caution

- Handle character strings following "#" as comments.
- The character string listed to the left of "=" is the "command name" of the data to be written to the printer and the character string listed to the right of "=" is the "registered transmission data".
- Ensure that character strings are listed using parentheses (" ").
- List binary data as two digit hexadecimals.
- The maximum size of "command names" is 33 bytes (33 characters in ANK).
- The maximum size of "registered transmission data" is 10,240 bytes. However, the size of the "registered transmission data" is not the length of an "ASCIIZ character string" but the size after converting the data to binary. Refer to the following examples.

  Example) ABC="ABC": The maximum size of "registered transmission data" is 3 bytes.
  ABC="ABC" 0D 0A: The maximum size of "registered transmission data" is 5 bytes.
  ABC=41 42 43 0D 0A: The maximum size of "registered transmission data" is 5 bytes.
- If a command name is already registered, stop the command registration process and return an error.
- The number of commands that can be registered is limited to the usable memory of the system.
- Call the BiCloseMonPrinter function to cancel the registered command data.

BiDirectSendRead

Executes the command defined in BiSendDataFile(ESC/POS commands).

> NOTE Contact the dealer regarding ESC/POS commands.

Syntax

```
nErr = BiDirectSendRead (nHandle, lpcCmdName,
                         lpcReadName, readLen, pReadBuf,
                         Timeout, nullTerminate )
```

Refer to the next argument.

Argument

nHandle: Specifies the handle. This is an INT type.

lpcCmdName: Specifies the "command name" of the command definition file name. This is a LPCSTR type.

lpcReadName: Specifies the "receiving data type name" of the data read from the printer. Refer to Page "BiSendDataFile" on page 67. This is a LPCSTR type.

readLen: Specifies the data length read from the printer. Does not write to the printer when "0". Returns the data length when reading. This is a LPDWORD type.

preadBuff: Specifies the buffer saving the data read from the printer.
This is a LPBYTE type.

Timeout: Specifies the timeout time in ms (milliseconds). This is a DWORD type.

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.

> NOTE Ensure that the size of preadBuff is the same length specified in readLen or longer.

Return value

Returns the following Status API execution errors (value) to the variable (nErr) defined by the INT type. Returns "SUCCESS" (macro definition) when this function is successfully called.

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_NO_MEMORY	-50	Not enough memory

Macro Definition (Constant)	Value	Description
ERR_HANDLE	-60	Specified handle is invalid
ERR_TIMEOUT	-70	Timeout error
ERR_ACCESS	-80	R/W cannot be performed on the printer
ERR_PARAM	-90	Parameter error
ERR_EXEC_FUNCTION	-310	This function is unavailable as Status API is used by other applications.
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 25.

Comment

This function specifies the name (macro name) previously specified in the command definition file. The following data types can be specified as responses from the printer.

Data Types	Description
ASB	Automatic status transmission
ASB Extended	Automatic status transmission regarding extended status
Ptr Info Byte	Printer ID information
Ptr Info String	Printer information B
Power OFF	Power OFF notification
Power ON	Power ON notification
Realtime	Realtime transmission of status
Buffer Clear	Buffer clear
Slip Remaining	Dot count transmission of remaining print area of a single-cut sheet
NVM Image Size	Transmission of full capacity of NV graphics area
NVM Image Free	Transmission of remaining capacity of NV graphics area
NVM Image Keys	Transmission of the key code list of defined NV graphics
NVM Image List	Transmission of the data ID list of image reading results saved to NV memory for storage
NVM User Used	Transmission of usage capacity (byte count of used area)
NVM User Free	Transmission of remaining capacity (byte count of unused area)
NVM User Get	Transmission of the storage data of a specified record
NVM User Keys	Transmission of the key code list of a storage record
NVM Set Mode	Transmission of transfer notification to user settings mode
NVM Get Mswitch	Transmission of memory switch values
NVM Set Size	Transmission of customized values

Data Types	Description
Ptr Info Type A	Printer information A
Test Print	Execution of test print
RAM Image Free	Transmission of remaining area of download graphics area
RAM Image Keys	Transmission of the key code list of defined download graphics
OfflineCode Bit	Transmission of offline response (bit format)
OfflineCode Data	Transmission of offline response (data format)
ProcessID	Transmission of process ID response
Data Types	Description
Buffer Clear24	Buffer clear 24
Other	Data not applicable to the aforementioned

BiSetDefaultEchoTime

Configures the response confirmation frequency of the network printer and the initial time value for a single timeout.

> **NOTE** Can only be used when connected by Ethernet.

Syntax

nErr = **BiSetDefaultEchoTime** (*Count*, *Timeout*)

Configures the response confirmation frequency to *Count* and the single timeout time to *Timeout*.

Argument

Count: Configures the response confirmation frequency (1 to 255). This is a BYTE type.

Timeout: Configures the single timeout time (1 to 65535) in ms (millisecond) units. This is a WORD type.

Return value

Returns the following Status API execution errors (value) to the variable (nErr) defined by the INT type. Returns "SUCCESS" (macro definition) when this function is successfully called.

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_PARAM	-90	Parameter error
ERR_EXEC_FUNCTION	-310	This function is unavailable as Status API is used by other applications.
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 25.
> - When a shred printer is accessed from a client that does not have PortHandler, the printer returns ERR_PH_NOT_EXIST.

Comment

The response confirmation frequency is three times and the timeout time is 1 second immediately after installing Status API. The configuration with this API will be valid after restart the computer. The configuration with this API will be valid to the all of the TM printers (w/ Ethernet Port) which are connected to the computer.

BiSetEtherEchoTime

Configures the response confirmation frequency of the network printer and the timeout time for one time after Status API is available.

> NOTE
> Can only be used when connected by Ethernet.

Syntax

nErr = **BiSetEtherEchoTime** (*nHandle*, *Count*, *Timeout*)

Specify the handle in *nHandle*. Configures the response confirmation frequency to *Count* and the single timeout time to *Timeout*.

Argument

nHandle: Specifies the handle. This is an INT type.

Count: Configures the response confirmation frequency (1 to 255). This is a BYTE type.

Timeout: Configures the single timeout time (1 to 65535) in ms (millisecond) units. This is a WORD type.

Return value

Returns the following Status API execution errors (value) to the variable (nErr) defined by the INT type. Returns "SUCCESS" (macro definition) when this function is successfully called.

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success
ERR_HANDLE	-60	Specified handle is invalid
ERR_ACCESS	-80	R/W 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_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 25.

Comment

If a value is not configured to this function, the value set in the BiSetDefaultEchoTime function is used.

BiSetReadWaitTimeOut

This is a compatible API. This API itself has no function.

Syntax

```
nErr = BiSetReadWaitTimeOut (nHandle, wTimeOut)
```

Argument

nHandle: Specifies the handle. This is an INT type.

wTimeOut: Not used. This is a WORD type.

Return value

Returns the following Status API execution errors (value) to the variable (nErr) defined by the INT type. Returns "SUCCESS" (macro definition) when this function is successfully called.

Macro Definition (Constant)	Value	Description
SUCCESS	0	Success

NOTE For information on remedies for the Status API execution errors, refer to "Status API Execution Error" on page 25.

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 25.

Status

Calls BiGetSatus in APD4StatusAPI and 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 50.

Methods

OpenMonPrinter

Starts controlling the specified printer.

Calls BiOpenMonPrinter in APD4 Status API. For details, see "BiOpenMonPrinter" on page 30.

Prototype

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

Parameters

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

Return value other than APD4 Status API return code

This method returns only an error code defined in com.epson.pos.driver.ErrorCode. (It does not return a handle.)

CloseMonPrinter

Stops controlling the specified printer.

Calls BiCloseMonPrinter in APD4 Status API. For details, see "BiCloseMonPrinter" on page 32.

Prototype

```
ErrorCode CloseMonPrinter ()
```

LockPrinter

Occupies the printer.

Calls BiLockPrinter in APD4 Status API. For details, see "BiLockPrinter" on page 33.

Prototype

```
ErrorCode LockPrinter (int timeout)
```

Parameters

int timeout: Timeout time (in ms units) .

UnlockPrinter

Stops occupying the printer.

Calls BiUnLockPrinter in APD4 Status API. For details, see "BiUnlockPrinter" on page 35.

Prototype

```
ErrorCode UnlockPrinter ()
```

SetMonInterval

Specifies the interval for Status API to read the status of the printer.

Calls BiSetMonInterval in APD4 Status API. For details, see "BiSetMonInterval" on page 36.

Prototype

```
ErrorCode SetMonInterval( int noPrnInterval, int prnInterval )
```

Parameters

int noPrnInterval: Unused

int prnInterval: Printer monitoring interval

SetMonEtherInterval

Specifies the interval for Status API to read the status of the printer.

Calls BiSetMonEtherInterval in APD4 Status API. For details, see "BiSetMonEtherInterval" on page 37.

Prototype

```
ErrorCode SetMonEtherInterval (int EtherInterval)
```

Parameters

int EtherInterval: Network printer monitoring interval

DirectIOEx

After sending the specified data to the printer, receives data of the specified length from the printer.
Calls BiDirectIOEx in APD4 Status API. For details, see "BiDirectIOEx" on page 40.

Prototype

- `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:	Whether or not to terminate reception when NULL is received
byte option:	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.
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.

Calls BiResetPrinter in APD4 Status API. For details, see "BiResetPrinter" on page 44.

Prototype

```
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.

Calls BiForceResetPrinter in APD4 Status API. For details, see "BiForceResetPrinter" on page 46.

Prototype

```
ErrorCode ForceResetPrinter()
```

CancelError

Calls BiCancelError in APD4 Status API. Performs recovery from a printer recoverable error.

For details, see "BiCancelError" on page 47.

Prototype

```
ErrorCode CancelError ()
```

GetType

Acquires the type ID of the printer. Some information cannot be acquired depending on the model. In such a case, “0”is set.

Calls BiGetType in APD4 Status API. For details, see "BiGetType" on page 49.

Prototype

```
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

GetRealStatus

Acquires the most recent status of the printer. Individual bits in the status correspond to the contents of the ASB status and constants defined in com.epson.pos.driver. ASB shall be used.

Calls BiGetRealStatus in APD4 Status API. For details, see "BiGetRealStatus" on page 51.

Prototype

```
ErrorCode GetRealStatus (out ASB asb)
```

Parameters

out ASB asb:	Current printer status

SetStatusBack

Starts status notification through StatusCallback/StatusCallbackEx events.

Calls BiSetStatusBackFunctionEx in APD4 Status API. For details, see "BiSetStatusBackFunctionEx" on page 54.

Prototype

```
ErrorCode SetStatusBack ()
```

CancelStatusBack

Stops status notification through StatusCallback/StatusCallbackEx events.

Calls BiCancelStatusBack in APD4 Status API. For details, see "BiCancelStatusBack" on page 57.

Prototype

```
ErrorCode CancelStatusBack ()
```

PowerOff

Executes the power-off process of the printer.

Calls BiPowerOff in APD4 Status API. For details, see "BiPowerOff" on page 58.

Prototype

```
ErrorCode PowerOff ()
```

GetCounter

Reads the maintenance counter.

Calls BiGetCounter in APD4 Status API. For details, see "BiGetCounter" on page 59.

Prototype

- `ErrorCode` ***`GetCounter`*** `(CounterIndex counter, bool cumulative, out int value)`

  Description: With the combination of CounterIndex counter and bool cumulative, calculates the counter number and acquires the value of the counter.

- `ErrorCode` ***`GetCounter`*** `(byte counter, out int value)`

  Description: Acquires value of the counter specified with byte counter.

Parameters

CounterIndex counter:	Maintenance counter numberConstants defined in com.epson.pos.driver.CounterIndex shall be used for the value.
bool cumulative:	Whether or not the maintenance counter number specified by counter refers to the cumulative counter true: cumulative counter false: reset counter
out int value:	Maintenance counter value
byte counter:	Maintenance counter number

ResetCounter

Resets the maintenance counter.

Calls BiResetCounter in APD4 Status API. For details, see "BiResetCounter" on page 61.

Prototype

- `ErrorCode` ***`ResetCounter`*** `(CounterIndex counter)`

  Description: Reset the value of counter specified with the com.epson.pos.driver.CounterIndex.

- `ErrorCode` ***`ResetCounter`*** `(byte counter)`

  Description: Reset the value of counter specified with byte counter.

Parameters

CounterIndex counter: Maintenance counter numberConstants defined in com.epson.pos.driver.CounterIndex shall be used for the value.

byte counter: Maintenance counter number

GetPrnCapability

Acquires information about the printer specified by the printer ID.

Calls BiGetPrnCapability in APD4 Status API. For details, see "BiGetPrnCapability" on page 63.

Prototype

- `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

OpenDrawer

Activates the drawer. Can be used also when the printer is offline.

Calls BiOpenDrawer in APD4 Status API. For details, see "BiOpenDrawer" on page 65.

Prototype

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

Parameters

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

Pulse pulse: Interval up to activation of the drawerConstants defined in com.epson.pos.driver. Pulse shall be used for the value.

SendDataFile

Registers commands by using the command definition file. For the file format of the command definition file, see the descriptions in the subsequent sections.

Prototype

```
ErrorCode SendDataFile (String filename)
```

Parameters

String filename: Command definition fileFile in the current folder is used if no path is specified.

Explanation

The registered command data is discarded when CloseMonPriner is executed. If the same command name is found to be already registered, aborts command registration and returns an error.

The number of commands that can be registered is restricted only by the available memory space of the system.

Calls BiSendDataFile in APD4 Status API.

For details, see "BiSendDataFile" on page 67.

DirectSendRead

Transmits the commands registered through SendDataFile and receives the data specified with the data type name of data to be received.

Calls BiDirectSendRead in APD4 Status API. For details, see "BiDirectSendRead" on page 69.

Prototype

- `ErrorCode` ***`DirectSendRead`*** `(String cmdName, String readName, ref byte[] readBuf, int timeout, bool nullTerminate)`

  Description: Sends a command defined with SendDataFile to a TM printer and receives the execution result (binary data) from the printer.

- `ErrorCode` ***`DirectSendRead`*** `(String cmdName, String readName, out String response, int timeout)`

  Description: Sends a command defined with SendDataFile to a TM printer and receives the execution result (character string data) from the printer.

- `ErrorCode` ***`DirectSendRead`*** `(String cmdName, String readName, int timeout)`

  Description: Only sends a command defined with SendDataFile to a TM printer. Not receives the execution result from the printer.

Parameters

String cmdName:	Command name
String readName:	Data type name of data to be received
ref byte[] readBuf:	Received data
int timeout:	Timeout time for data transmission and data reception (in ms units)
bool nullTerminate:	Whether or not to terminate reception when NULL is received. When “False” is specified, reads as much data as specified with readBuf, or reads data from the TM printer until a time-out error occurs.
out String response:	Received data

SetDefaultEchoTime

Sets initial values for the number of response confirmation times and for the timeout time per one response confirmation, to the network printer.
When APD4StatusAPI is installed for the first time, the number of confirmation response times is one and the timeout time per three response confirmation is 1 second.
The configuration with this API will be valid after restart the computer.
Note that this API is executable only during Ethernet connection.
Calls BiSetDefaultEchoTime in APD4 Status API. For details, see "BiSetDefaultEchoTime" on page 72.

Prototype

```
ErrorCode SetDefaultEchoTime (int count, int timeout)
```

Parameters

int count: Number of response confirmation times

int timeout: Timeout time per one response confirmation (in ms units)

SetEtherEchoTime

Sets values for the number of response confirmation times and for the timeout time per one response confirmation, to the network printer. Before this API executes, the values set with SetDefaultEchoTime are used. Note that this API is enabled only during Ethernet connection.
Calls BiSetEtherEchoTime in APD4 Status API. For details, see "BiSetEtherEchoTime" on page 73.

Prototype

```
ErrorCode SetEtherEchoTime (int count, int timeout)
```

Parameters

int count: Number of response confirmation times

int timeout: Timeout time per one response confirmation

Events

StatusCallback

Event that handles ASB status notification.

Corresponds to the callback function specified by BiSetStatusBackFunction in APD4 Status API.
For details, see "BiSetStatusBackFunction" on page 52.

Prototype

```
StatusCallbackHandler (ASB asb)
```

Parameters

ASB asb:	ASB statusConstants defined in com.epson.pos.driver. ASB shall be used for the value.

StatusCallbackEx

Event that handles ASB status notification.

Corresponds to the callback function specified by BiSetStatusBackFunctionEx in APD4 Status API.
For details, see "BiSetStatusBackFunctionEx" on page 54.

Prototype

```
StatusCallbackHandlerEx (ASB asb, String portName)
```

Parameters

ASB asb:	ASB statusConstants defined in com.epson.pos.driver. ASB shall be used for the value.
String portName:	Port name

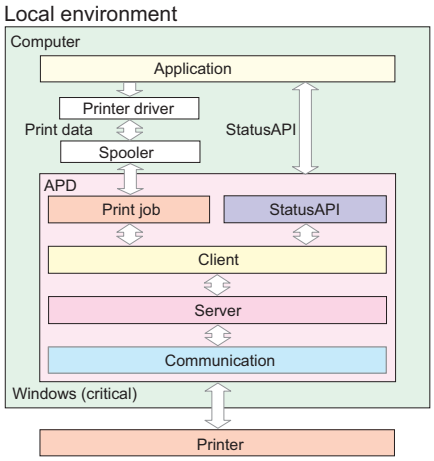
Generating Log Files

This chapter explains how to output and view log files.

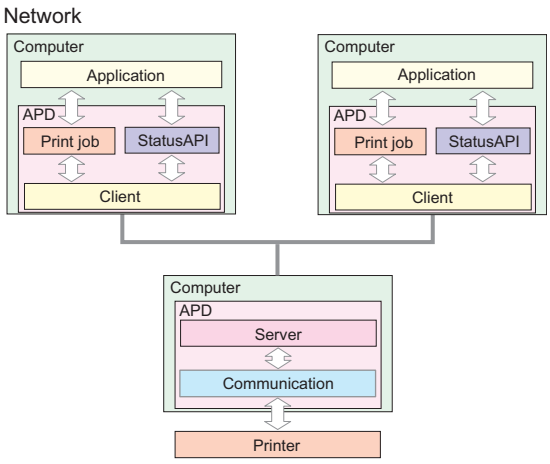
The APD allows you to create a log file which can help you to troubleshoot a problem quickly. The log file is saved under a name of EpsonPOSPort.log. The main features are as follows.

- When a trouble occurs in a printing system, you can view Windows error information in addition to the APD log.
- A process ID can be acquired. You can identify which log is for which process when multiple processes have been executed.
- Client-server system is supported. Logs of client/server module can be acquired.
- Log file can be generated for each of the following items; Status API, print job, client, server, and communication module.

The following diagram shows where the APD log is acquired. Logs for each module are output as a single log file in the order of acquisition.



The module configuration of client-server system is shown below.

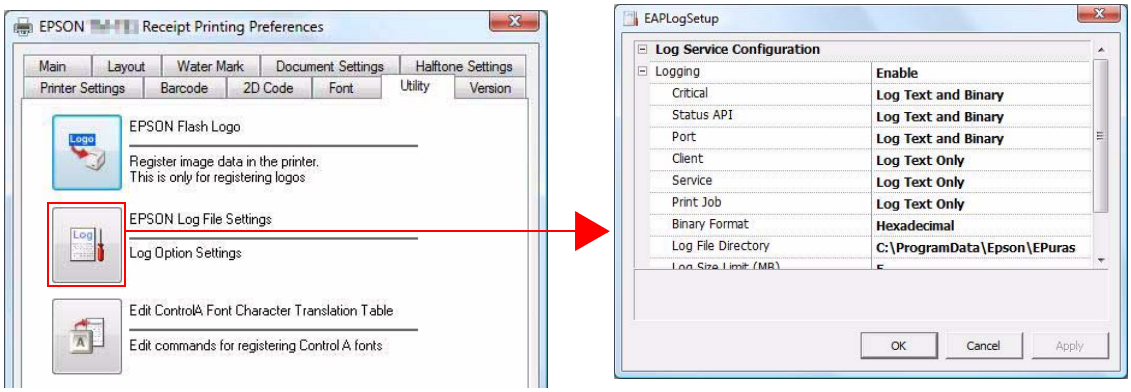


A log file is generated for each computer. The computer's date and time setting is applied to the time stamp.

Log Files Settings

You can select whether to output the log file or not, which module to be logged, and where to output the log file.

Select the Utility tab on the properties screen, and click the [EPSON Log File Setting] button.



Make the following settings on the tab.

Setting		Description
Logging	Enable (Default)	Enables log output.
	Disable	Disables log output.
Critical	Select how the Windows error information is output.	
	Log Text Only	Outputs the log as text data.
	Log Text and Binary (Default)	Outputs the log as text and binary data.
Status API	Select how the Status API log is output.	
	Do Not log	A Status API log is not output.
	Log Text Only	Outputs the log as text data.
	Log Text and Binary (Default)	Outputs the log as text and binary data.
Port	Select how the log of the communication port is output.	
	Do Not log	A communications port log is not output.
	Log Text Only	Outputs the log as text data.
	Log Text and Binary (Default)	Outputs the log as text and binary data.
Client	Select how the log of the application on the client-server system is output.	
	Do Not log	A client log is not output.
	Log Text Only (Default)	Outputs the log as text data.
	Log Text and Binary	Outputs the log as text and binary data.
Service	Select how the log of the server on the client-server system is output.	
	Do Not log	A service log is not output.
	Log Text Only (Default)	Outputs the log as text data.
	Log Text and Binary	Outputs the log as text and binary data.

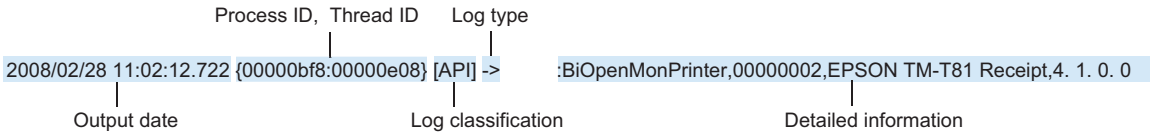
Setting		Description
Print Job	Select how the log of print jobs is output.	
	Do Not log	A print job information log is not output.
	Log Text Only (Default)	Outputs the log as text data.
	Log Text and Binary	Outputs the log as text and binary data.
Binary Format	Configures the binary data format.	
	Hexadecimal (Default)	Outputs the log to a hexadecimal log file.
	Base64	Outputs the log to a Base64 log file.
Log File Directory	Specify where to output the log file. (Default) Windows XP: C:\Documents and Settings\All Users\Application Data\Epson\EPuras Windows 7 / Vista: C:\ProgramData\Epson\EPuras	
Log Size Limit (MB)	Specify the upper limit of the log file size. When the upper limit is exceeded, the log file is compressed using zip format and saved as a BAK file. The subsequent log information is saved as a new log file. A sequential number is added to the name of the BAK files. (example: EpsonPOSPort1.bak). Specify the number of log files to be backed up. (Range: 1 to 1024 , Default: 5)	
Backup File Count	Specify the number of log files to be backed up. (Range 1 to 9 , Default: 1)	

NOTE

Output function of hexadecimal dumping list is not supported.

Viewing Log Files

A log file is viewed as follows.



Log Data	Explanation
Output Date	YYYY/MM/DD hh:mm:ss.sss
Process ID, Thread ID	{Process ID: Thread ID}
Log classification	Shows of which module the log file is. (See "Log classification" on page 94.)
Log type	-> Call function <- Return function -- Execute function (call, return) ** Occurrence of an event

Log Data	Explanation
Detailed information	Information for each module and log type When a function is executed: Function name (parameter 1, ---, parameter n) <return> (execution time (ms))

Log classification

Module	Log classification	Contents of detailed information
Critical	!!!	Important events and errors on Windows
Status API	API	Status API call function and its parameter information, or called Status API function and its parameter and return information
Port	PRT	Port controls, events specific to the interface, and input/output data information
Client	CLI	Process information of the application on the client-server system.
Server	SVR	Process information of the server on the client-server system.
Print Job	SPL	Port open/close information and input/output to/from the port information

Log output example

```
2008/02/28 11:02:12.722 {00000bf8:00000e08} [API] ->    :BiOpenMonPrinter,00000001,EPSON TM-T81 Receipt,4. 1. 2. 0
2008/02/28 11:02:12.722 {00000bf8:00000e08} [CLI] -> Open('pipe://TM/ESDPRT001', 0x01473460)
2008/02/28 11:02:12.722 {000006dc:00000cd0} [SVR] -> 0036d4e8::Open(0, TM/ESDPRT001)
2008/02/28 11:02:12.722 {000006dc:00000cd0} [SVR] <- 0036d4e8::Open(8, TM/ESDPRT004) <00000000>
2008/02/28 11:02:12.722 {00000bf8:00000e08} [CLI] ** (TM/ESDPRT001)Event(0x00010003) 4:
2008/02/28 11:02:12.722 {000006dc:00000cdc} [SVR] -- 0036d4e8::RegisterCallback(8, 00010001) <PHR_SUCCESS>
2008/02/28 11:02:12.722 {00000bf8:00000e08} [CLI] <- Open('pipe://TM/ESDPRT001', 1) <00000000>
2008/02/28 11:02:12.722 {00000bf8:00000e08} [CLI] -> RegisterCallback(1, 0x00040002, 0x01396e00, 0x0177f2f0)
2008/02/28 11:02:12.722 {000006dc:00000c7c} [SVR] -- 0036d4e8::RegisterCallback(8, 00040002) <PHR_SUCCESS>
2008/02/28 11:02:12.722 {00000bf8:00000e08} [CLI] <- RegisterCallback(1, 0x00040002, 0x01396e00, 0x0177f2f0)
                    <00000000>
```

Appendix

Model Information

This document explains the information acquired by Status API for the TM-T81.

TM-T81

ASB Status

Macro Definitions	ON/ OFF	Value	Status
ASB_NO_RESPONSE	ON	0x00000001	No printer response
	OFF	0x00000000	Printer response
ASB_PRINT_SUCCESS	ON	0x00000002	Print complete
	OFF	0x00000000	-
ASB_DRAWER_KICK	ON	0x00000004	Status of the drawer kick number 3 connector pin = "H"
	OFF	0x00000000	Status of the drawer kick number 3 connector pin = "L"
ASB_OFF_LINE	ON	0x00000008	Offline status
	OFF	0x00000000	Online status
ASB_COVER_OPEN	ON	0x00000020	Cover is open
	OFF	0x00000000	Cover is closed
ASB_PAPER_FEED	ON	0x00000040	Paper feed switch is feeding paper
	OFF	0x00000000	Paper feed switch is not feeding paper
ASB_AUTOCUTTER_ERR	ON	0x00000800	Auto cutter error generated
	OFF	0x00000000	Auto cutter error not generated
ASB_UNRECOVER_ERR	ON	0x00002000	Unrecoverable error generated
	OFF	0x00000000	Unrecoverable error not generated
ASB_AUTORECOVER_ERR	ON	0x00004000	Auto recovery error generated
	OFF	0x00000000	Auto recovery error not generated
ASB_RECEIPT_NEAR_END	ON	0x00020000	No paper in the roll paper near end detector
	OFF	0x00000000	Paper in the roll paper near end detector

Macro Definitions	ON/ OFF	Value	Status
ASB_RECEIPT_END	ON	0x00080000	No paper in the roll paper end detector
	OFF	0x00000000	Paper in the roll paper end detector
ASB_SPOOLER_IS_STOPPED	ON	0x80000000	Stop the spooler
	OFF	0x00000000	Operation the spooler

Maintenance Counter

Counter Number (readno)	Resetability	Counter (readcounter)	Unit
20 (14H)	Resetable	Number of line feeds (for roll paper)	Lines
21 (15H)	Resetable	Number of times head is energized (for roll paper)	Times
50 (32H)	Resetable	Number of autocutter operations	Times
70 (46H)	Resetable	Printer operation time	Hours
148 (94H)	Cumulative	Number of line feeds (for roll paper)	Lines
149 (95H)	Cumulative	Number of times head is energized (for roll paper)	Times
178 (B2H)	Cumulative	Number of autocutter operations	Times
198 (C6H)	Cumulative	Printer operation time	Hours