Printer Utility DLL

(PADLL.DLL & PADLL.H)

Practical Automation Inc.

Version 2.0.0.9 Rev A

PRACTICAL AUTOMATION, INC.
The Alinabal Group of Companies
45 Woodmont Road, P.O. Box 3028, Milford, CT 06460 ▪ Phone: (203) 882-5640 ▪ FAX: (203) 882-5648
www.practicalautomation.com

Table of Contents

Document and DLL Release History

Release	Date	Name	Comments
V1.00	3/13/01		Initial Release of this document.
V1.01	4/12/01		Updated comments to reflect the name change of "DLLsimple" to "PADLL."
V1.02	4/23/01		Changed the option on the TOC to have invisible boxes for the hyperlinked entries. No changes to content were made.
V1.03	12/02/01		This document corresponds with the DLL version V1.02. The DLL has been updated to operate under WinNT 4.0, Win 98 and Win 2K. No changes have been to this document.
V1.04	05/07/02		This document corresponds with the DLL version V1.03. Language Monitor interface functions have been added and reflected in this document.
	06/20/02		Added remote printer flash reprogram capability.
V1.05	10/23/02		Added printer driver pass-through functions. This document corresponds with the DLL version V1.05.
V1.06	12/02/02		Added block diagrams for the non-LM and LM printer status acquisition techniques and application information TOC links for this information. Added documentation regarding paSetOrientation
V1.07	01/23/03		Added : paLMGetProgress - Get firmware upload status paLMParseFlashStatus - Parse upload status paSetNewPrinter - Point DLL to new printer paLMAbortDocument - Abort document in process paGetSpoolingStatus - Get spooling status
V1.08 B2	10/25/04		Updated paGetStatus so that it properly functions with printers connected to LPT2 and LPT3 Updated version tables for paProgramFlash Added: paCreateFont - Create font for FGL printers paTextOut - Text out for FGL printers paSelectObject - Select font for FGL printers paDeleteObject - Delete font for FGL printers
V1.11.3	07/09/08		Removed obsolete functions and added set registry key functions and rotate image functions. Add notes regarding .NET compatibility. Added paPulseCashDrawerGPrinters function Added operating system mutex to manage multiple access of padll.dll
V2.0.0.0	12/15/17	WDW	Created 64bit version labeled as padllx64.dll. This can be renamed to padll.dll if required. Removed some compilation warnings Removed references to: DllRegisterServer PRIVATE DllUnregisterServer PRIVATE Upgraded development environment
V2.0.0.1	02/01/17	WDW	Added feature to create fake print job in an attempt to force spooler to load language monitor dll. Removed some compilation warnings
V2.0.0.2	02/02/17	WDW	Renamed padllx64 to padll Added forcing spooler to launch LM feature to other API functions Changed deletejob function to delete created function only.

DLL Version	Document Revision	Date	Name	Comments
V2.0.0.3	DRAFT A	2017-12-05	WDW	Added paLMGetPrinterOKStatus2 Added paLMRFIDPrinter for supported printers Added NOTE about PA_PRINTER_DISPOSE_TOGGLE Changed Revision 9 to Version V2.0.0.3 to match dll versioning Removed mentions of padll.bas as this doesn't exist. The definitions in the sample globals.bas can be used instead.
V2.0.0.3	A	2017-12-15	WDW	Adjusted box colours to match PA Updated timeout note on paLMRFIDPrinter
V2.0.0.4	A	2018-08-01	WDW	Fixed Form Length parsing issue caused signed char and unsigned char arithmetic.
V2.0.0.5	A	2018-09-20	WDW	Added paGetDLLVersion to dll
V2.0.0.6	A	2018-10-04	WDW	Added mutex to paLMIsPipe, paLMSetGeneric, paLMGetGeneric and paLMRFIDPrinter to prevent file handle corruption.
V2.0.0.7	A	2018-10-05	WDW	Changed code to request PRINTER_ALL_ACCESS when using OpenPrinter to have resolved the hang. Improved 64bit compatibility by changing some calls to use A versions of calls when passing char data around.
V2.0.0.8	A	2018-10-08	WDW	• Improved text file logging and information is now logged to temp folder (DEBUG VERSIONS ONLY). • Improved thread-safe handling. • Fixed issue where GetDefaultPrinter failed because buffer size was wrong. • Changed all OpenPrinter calls to request PRINTER_ALL_ACCESS when opening. • Changed DLL_PROCESS_ATTACH to only do first stage printer validation. • Updated DLL_PROCESS_ATTACH to use default PA printer if installed. • Removed unnecessary OpenPrinter calls from DllMain.
V2.0.0.9	A	2018-10-16	WDW	• Resolved incorrect document entries for paParseStatus and paLMGetStatus. • Added paLMPtrGetStatus, paLMPtrRFIDPrinter, paLMPtrIsPipe • Added troubleshooting entry for driver permissions on Windows 10

OVERVIEW

All Printer Utility DLL (later referred to as DLL) function calls, unless told otherwise, assume that the user is manipulating the system's default printer. An API function paSetNewPrinter is provided to point the DLL functions to a printer other than the default printer. This DLL provides an Application Programmers Interface (API) that permits the printer's parameters to be obtained, or modified, as well as function calls to permit the reading the detailed status information available from printer over its connected PC interface. This DLL can save the application programmer many man-hours of work by having much of this API detail work executed by the DLL.

This DLL is provided, to the applications programmer, on an "as is" basis. Its fitness, for any application, must be the determined by the application programmer. Practical Automation makes no warranty of its fitness to your application.

Additionally the toolkit includes the files PADLL.H, PADLL.LIB

PADLL.H, PADLL.LIB are used in C and C++ projects that perform compile-time binding of the functions exported by PADLL.DLL

Sample Applications GLOBALS.BAS can be used by VB users. The file contains VB external declarations to the functions provided by PADLL.DLL

x86\PADLL.DLL is the 32bit version and can be used on 32bit and 64bit operating systems.

x64\PADLL.DLL is the 64bit version and can only be used on 64bit operating systems (excluding Windows XP 64bit).

FUNCTION RETURN CODES

```
API functions that return a long return code will return
one of the following constants. These constants are defined in padll.h (C and
C++)

PA_ERR_OK                     = Function completed without error

PA_ERR_PRINTER_NOT_RESPONDING = Printer is not responding to status
                                This is generally due to the printer
                                being powered off, the printer
                                being disconnected or the wrong port
                                selected in the printers panel.

PA_ERR_PIPE_OPEN_FAILED       = Status connection between the application
                                And the printer could not be opened.
                                This is generally due an attempt to communicate
                                with a printer other than the default printer.
                                In this case use paSetNewPrinter to point PADLL
                                Functions to the correct non-default printer.

PA_XXX                        = Return codes specific to a given function.
                                Please refer to individual API for details.

Accessing Practical Automation functions via .NET applications.
```


The toolkit contains a .NET compiled class library (PADLL_DOT_NET.DLL) that wraps PADLL.DLL. The class library can be used with any of the .NET languages (Visual Basic, Visual C#, Visual J# and Visual C++).

The toolkit contains two sample projects (C# and VB.NET) that document the use of the class library. The projects are zipped as CSGetStatusOnlyLM.zip and VBGetStatusOnlyLM.zip.

SET NEW PRINTER

(if not Default Printer)

paSetNewPrinter

This DLL, PADLL.DLL acquires the name of the system default printer when loaded. This default printer is used for all subsequent operations. In may be necessary to communicate with a printer other than the default printer. Use this function, paSetNewPrinter to point the DLL to a new printer. Note that the printer name passed to the function must match that of a currently installed printer.

```
BOOL APIENTRY paSetNewPrinter(char * pszPrinterName);

Declare Function paSetNewPrinter Lib "PADLL" (ByVal pszNewPrinter As String) As
Boolean
```

Parameters

pszPrinterName[in]

Pointer to null terminated new printer name.

Return Values

TRUE if printer redirected, else FALSE.

.NET Support

C#, VB.NET example included in CSGetStatusOnlyLM.zip and VBGetStatusOnlyLM.zip. (Practical Automation toolkit).

STATUS COLLECTION

paLMGetStatus

Get printer status. This function will collect status based on the type of status requested. The details of each status type returned can be found in the programmer's manual for each printer. The status returned can be parsed into a structure for immediate use. Refer to description of paParseStatus that follows. Note that the function will return **Complete** status if the printer is Busy.

```
LRESULT APIENTRY paLMGetStatus(DWORD dwType, BYTE * pResultBuffer, DWORD
dwResultBufferLength, DWORD * pdwResultBufferRead);

Declare Function paLMGetStatus Lib "PADLL" (ByVal dwType As Long, ByRef
pResultBuffer As Byte, ByVal dwResultBufferLength As Long, ByRef
pdwResultBufferRead As Long) As Long
```

Parameters

dwType[in]

Type of status requested:

0 = Short
1 = Normal
2 = Extended
3 = Ticket Count
4 = Head temperature
5 = Form Length
6 = Complete

pResultBuffer[in]

Pointer to a byte buffer to receive the status.
This buffer must be at least PA_RAW_STATUS_MAX_LENGTH bytes long (128)

dwResultBufferLength[in]

The length of the result buffer.

pdwResultBufferRead[In,out]

Number of bytes returned by this function

Return Values

Refer to **FUNCTION RETURN CODES** for details

C Example Sequence

```
STATUS        statPrinter;      // Structure defined in PADLL.H
BYTE          uchStatus[PA_STATUS_MAX_LENGTH];
LRESULT       lResult;
DWORD         lStatRead;

lResult = paLMGetStatus(PA_STAT_COMPLETE, &uchStatus[0], sizeof(uchStatus), &lStatRead);

// Now parse the status
lResult = paParseStatus(&uchStatus[0], lStatRead, &statPrinter);
```

Refer to programmer's manual for details of each status field.
See paParseStatus for details of status structure.

---

paParseStatus is generally used to obtain the operational status of the printer. If the requested status type is Short, Normal, Extended or Complete, bytes 3 and 4 of the returned status encode the operational status of the printer. The bit assignments for these 2 bytes are:

```
// Byte 3
#define PA_PRINTER_NOT_READY            0x80
#define PA_PRINTER_SYS_ERROR            0x40
#define PA_PRINTER_MAN_DESELECT         0x20
#define PA_PRINTER_OUTPUT_JAM           0x10
#define PA_PRINTER_REGISTRATION         0x08
#define PA_PRINTER_CUTTER_JAM           0x04
#define PA_PRINTER_HEAD_LEVER_UP        0x02
#define PA_PRINTER_OUT_OF_PAPER         0x01

// Byte 4
#define PA_DOC_IN_PROCESS               0x80
#define PA_PRINTER_DATA_ERROR           0x40
#define PA_PRINTER_BUSY                 0x20
#define PA_PRINTER_TEMP_WAIT            0x10
#define PA_PRINTER_TICKET_NOT_TAKEN     0x08
#define PA_PRINTER_AVG_POWER_WAIT       0x04
#define PA_PRINTER_DISPOSE_TOGGLE       0x02
#define PA_PRINTER_LOW_PAPER            0x01

// Where:
// PA_PRINTER_SYS_ERROR AND PA_PRINTER_DATA_ERROR = Purge occurred
// PA_PRINTER_SYS_ERROR AND (NOT PA_PRINTER_DATA_ERROR) = Data error
```

NOTE:

PA_PRINTER_DISPOSE_TOGGLE is a toggle and will not be reset when another print is received

.NET Support

C#,VB.NET example included in CSGetStatusOnlyLM.zip and VBGetStatusOnlyLM.zip. (Practical Automation toolkit).

paParseStatus

The Parse status function converts binary status data acquired by the paStatusFunction into a convenient structure.

```
LRESULT APIENTRY paParseStatus(PUCHAR pStatus, LONG lLen, PSTATUS pStat);

Declare Function paParseStatus Lib "PADLL" (ByVal pStatus As String, ByVal lLen
As Long, ByRef pStat As Status) As Long
```

Parameters

pStatus[in]

Pointer to array of status characters acquired using paLMGetStatus

lLen[in]

Length of buffer pStatus

pStatus[in,out]

Pointer to structure of type PSTATUS as defined in padll.h, NULL on error

Return Values

PA_ERR_OK
PA_ERR_NO_PRINTERS
PA_ERR_BLOWN_POINTER

Remarks

```
See paLMGetStatus for code example.
Structure STATUS is defined as:
typedef struct{
      LONG  lStatBits;              //field 00
      LONG  lError;
      LONG  lDocCount;
      LONG  lTemp_C;
      LONG  lTemp_F;
      LONG  lFormLen;               //field 04
      CHAR  type[2];                // C or E or A etc
      CHAR  chFormLen[16];          //field 04
      CHAR  chTemperature[4]; //field 03
      CHAR  chDocCount[9];          //field 02
      CHAR  chError[4];             //field 01
      CHAR  chStatBits[8];          //field 00
      CHAR  version[64];            //field 05
      CHAR  ver[16];                // ver  part of field 05
      CHAR  ver_date[16];           // date part of field 05
      //
} STATUS, *PSTATUS;
```

Refer to programmer's manual for details of each status field.

.NET Support

C#,VB.NET example included in CSGetStatusOnlyLM.zip and VBGetStatusOnlyLM.zip. (Practical Automation toolkit).

Simple printer status functions. Note that at times the application may want to Collect a single piece of status information. The following functions perform and status request and parse data accordingly. NOTE each function call results in a dialog with the printer. If more than one status id is needed it will be more efficient to collect status via paLMGetStatus and parse the individual status bit fields.

paLMGetPrinterOKStatus

This function returns the printer okay status. Printer okay is defined as no errors and not low paper.

If you don't want low paper to affect the result returned by *paLMGetPrinterOKStatus* then use *paLMGetPrinterOKStatus2* instead.

```
LRESULT APIENTRY paLMGetPrinterOKStatus(BOOL * pbPrinterOKStatus);

Declare Function paLMGetPrinterOKStatus Lib "PADLL" (ByRef pbPrinterOKStatus As
Boolean) As Long
```

Parameters

pbPrinterOKStatus[in,out]

Pointer to bool set by function
FALSE = Printer is NOT okay. – (Check other functions for details)
TRUE = Printer is okay.

Return Values

Refer to **FUNCTION RETURN CODES** details

.NET Support

C#,VB.NET example included in CSGetStatusOnlyLM.zip and VBGetStatusOnlyLM.zip. (Practical Automation toolkit).

paLMGetPrinterOKStatus2

This function returns the printer okay status. Printer okay is defined as no errors.

Unlike *paLMGetPrinterOKStatus*, *paLMGetPrinterOKStatus2* does **not** use the low paper setting to make the printer not ready.

```
LRESULT APIENTRY paLMGetPrinterOKStatus2(BOOL * pbPrinterOKStatus);

Declare Function paLMGetPrinterOKStatus2 Lib "PADLL" (ByRef pbPrinterOKStatus
As Boolean) As Long
```


Parameters

pbPrinterOKStatus2[in,out]
Pointer to bool set by function
FALSE = Printer is NOT okay. – (Check other functions for details)
TRUE = Printer is okay.

Return Values

Refer to **FUNCTION RETURN CODES** details

.NET Support

C#,VB.NET example included in CSGetStatusOnlyLM.zip and VBGetStatusOnlyLM.zip. (Practical Automation toolkit).

paLMGetOutOfPaperStatus

This function returns the printer out of paper status.

```
LRESULT APIENTRY paLMGetOutOfPaperStatus(BOOL * pbOutOfPaperStatus);

Declare Function paLMGetOutOfPaperStatus Lib "PADLL"(ByRef pbOutOfPaperStatus
As Boolean) As Long
```

Parameters

pbOutOfPaperStatus[in,out]
Pointer to bool set by function:
FALSE = Printer not out of paper
TRUE = Printer is out of paper

Return Values

Refer to **FUNCTION RETURN CODES** for details

.NET Support

C#,VB.NET example included in CSGetStatusOnlyLM.zip and VBGetStatusOnlyLM.zip. (Practical Automation toolkit).

paLMGetLowPaperStatus

This function returns the printer low paper status.

```
LRESULT APIENTRY paLMGetLowPaperStatus(BOOL * pbLowPaperStatus);
```



```
Declare Function paLMGetLowPaperStatus Lib "PADLL"(ByRef pbLowPaperStatus As
Boolean) As Long
```

Parameters

pbLowPaperStatus[in,out]
Pointer to bool set by function:
FALSE = Printer not low paper
TRUE = Printer is low paper

Return Values

Refer to **FUNCTION RETURN CODES** for details

.NET Support

C#,VB.NET example included in CSGetStatusOnlyLM.zip and VBGetStatusOnlyLM.zip. (Practical Automation toolkit).

paLMGetHeadLeverOpenStatus

This function returns the printer head lever open status.

```
LRESULT APIENTRY paLMGetHeadLeverOpenStatus(BOOL * pbHeadLeverOpenStatus);

Declare Function paLMGetHeadLeverOpenStatus Lib "PADLL" (ByRef
pbHeadLeverOpenStatus As Boolean) As Long
```

Parameters

pbHeadLeverOpenrStatus[in,out]
Pointer to bool set by function
FALSE = Printer head lever is closed
TRUE = Printer head lever is open

Return Values

Refer to **FUNCTION RETURN CODES** for details

.NET Support

C#,VB.NET example included in CSGetStatusOnlyLM.zip and VBGetStatusOnlyLM.zip. (Practical Automation toolkit).

paLMGetManDeselectStatus

This function returns the printer manual deselect status.


```
LRESULT APIENTRY paLMGetManDeselectStatus(BOOL * pbManDeselectStatus);

Declare Function paLMGetManDeselectStatus Lib "PADLL" (ByRef
pbManDeselectStatus As Boolean) As Long
```

Parameters

pbManDeselectrStatus[in,out]
Pointer to bool set by function
FALSE = Printer is selected and online.
TRUE = Printer is manually deselected.

Return Values

Refer to **FUNCTION RETURN CODES** for details

.NET Support

C#,VB.NET example included in CSGetStatusOnlyLM.zip and VBGetStatusOnlyLM.zip. (Practical Automation toolkit).

paLMGetJamStatus

This function returns the printer jammed status.

```
LRESULT APIENTRY paLMGetJammedStatus(BOOL * pbJamStatus);

Declare Function paLMGetJammedStatus Lib "PADLL" (ByRef pbJamStatus As Boolean)
As Long
```

Parameters

pbJamStatus[in,out]
Pointer to bool set by function
FALSE = Printer is not jammed.
TRUE = Printer is jammed.

Return Values

Refer to **FUNCTION RETURN CODES** for details

.NET Support

C#,VB.NET example included in CSGetStatusOnlyLM.zip and VBGetStatusOnlyLM.zip. (Practical Automation toolkit).

paLMGetTicketNotTakenStatus

This function returns the printer ticket not taken status.

```
LRESULT APIENTRY paLMGetTicketNotTakenStatus(BOOL * pbTicketNotTakenStatus);

Declare Function paLMGetTicketNotTakenStatus Lib "PADLL" (ByRef
pbTicketNotTakenStatus As Boolean) As Long
```

Parameters

pbTicketNotTakenStatus[in,out]
Pointer to bool set by function
FALSE = Ticket has been taken.
TRUE = Ticket has not been taken. (Ticket remains in presenter)

Return Values

Refer to **FUNCTION RETURN CODES** for details

.NET Support

C#,VB.NET example included in CSGetStatusOnlyLM.zip and VBGetStatusOnlyLM.zip. (Practical Automation toolkit).

paLMGetPrintInProgress

This function allows an application to test if there is a document in progress. Normal status operations inhibited during printing however this function will return printing status even if there is a document in progress.

```
LRESULT APIENTRY paLMGetPrintInProgress(BOOL * pbPrintInProgressStatus);

Declare Function paLMGetPrintInProgress Lib "PADLL" (ByRef
pbPrintInProgressStatus As Boolean) As Long
```

Parameters

pbPrintInProgressStatus[in,out]]
Pointer to bool set by function
FALSE = Print not in progress
TRUE = Print in progress

Return Values

Refer to **FUNCTION RETURN CODES** for details

.NET Support

C#,VB.NET example included in CSGetStatusOnlyLM.zip and VBGetStatusOnlyLM.zip. (Practical Automation toolkit).

paLMGetAddStatus

Get printer status using the 'address method'. This function will send the addressed status command, collect and return status.
Please refer to the Programmer's manual for details regarding addressed status.

```
LRESULT APIENTRY paLMGetAddStatus(char * pStatusReqString, BYTE *
pResultBuffer, DWORD dwResultBufferLength, DWORD * pdwResultBufferRead);

Declare Function paLMGetAddStatus Lib "PADLL" (ByRef pStatusReqString As Byte,
ByRef pResultBuffer As Byte, ByVal dwResultBufferLength As Long, ByRef
pdwResultBufferRead As Long) As Long
```

Parameters

pStatusRequestString[in]
Pointer to status request string that will be sent to the printer before a status read occurs. (See manual)

pResultBuffer[in]
Pointer to a byte buffer to receive the status.

dwResultBufferLength[in]
The length of the result buffer.

pdwResultBufferRead[in,out]
Number of bytes returned by this function

Return Values

Refer to **FUNCTION RETURN CODES** for details

.NET Support

C#,VB.NET example included in CSGetStatusOnlyLM.zip and VBGetStatusOnlyLM.zip. (Practical Automation toolkit).

DOCUMENT SETTINGS

paSetOrientation

Sets printer orientation.

```
LRESULT APIENTRY paSetOrientation(SHORT orientationID);

Declare Function paSetOrientation Lib "PADLL" (ByVal orientationID As Long) As
long
```

Parameters

orientationID[in]

Desired printer orientation. Must be either:
DMORIENT_LANDSCAPE
DMORIENT_PORTRAIT

Return Values

PA_ERR_OK
Otherwise use API GetLastError() to acquire system error code.

C Example Sequence

```
LRESULT result;

result = paSetOrientation(DMORIENT_LANDSCAPE);

        or

result = paSetOrientation(DMORIENT_PORTRAIT);
```

VB Example Sequence

```
Dim result As Long

result = paSetOrientation(DMORIENT_LANDSCAPE)

        or

result = paSetOrientation(DMORIENT_PORTRAIT)
```

.NET Support

C#,VB.NET example included in CSGetStatusOnlyLM.zip and VBGetStatusOnlyLM.zip. (Practical Automation toolkit).

paSetSheetSize (Set Form Size)

For those printers capable of it this function changes the cut length to one of the pre-selected values as defined by the printer driver. The function will search for the paper size that most closely matches the desired values. The function will return the values actually set in width and length.

```
LRESULT APIENTRY paSetSheetSize(PFLOAT width, PFLOAT length);

Declare Function paSetSheetSize Lib "PADLL" (ByRef Width As Single, ByRef
Length As Single) As Long
```

Parameters

width[in]
Desired page width in inches
length[in]
Desired page length in inches

Return Values

PA_ERR_OK // Exact page size found
PA_ERR_SHEET_NOT_FOUND // Exact page size not found, set to closest page size

C Example Sequence

Example 1: Printer supports a paper size of 8.00 x 4.50 inches

```
Float width,length;
LRESULT lResult;

width = 8.00;
length = 4.50;
lRresult = paSetSheetSize(width, length);

// Details after function call
lResult = PA_ERR_OK
width = 8.00
length = 4.50
```


Example 2: Printer supports a paper sizes of 8.00 x 4.50 inches and 8.00 x 4.75 inches

```
Float width,length;
LRESULT lResult;

width = 8.00;
length = 4.60; (Note this length is not supported by the printer)
lRresult = paSetSheetSize(width, length);

// Details after function call (Printer page size now set to 8.00 x 4.50)
lResult = PA_ERR_SHEET_NOT_FOUND
width = 8.00
length = 4.50
```

VB Example Sequence

```
Dim Width, Length As Single

Width = 8#
Length = 4.5
lRresult = paSetSheetSize(Width, Length)
```

.NET Support

C#,VB.NET example included in CSGetStatusOnlyLM.zip and VBGetStatusOnlyLM.zip. (Practical Automation toolkit).

SET TICKET PACKETING

(TICKET PRINTERS ONLY)

paSetBinByBinName

Updates the bin ID for the current printer. This is used by some PA printers for cut control, e.g. cut each, cut every 2nd, etc.

```
LRESULT APIENTRY pSetBin(LPSTR pBinName);

Declare Function paSetBin Lib "PADLL" (ByVal pBinName As String) As Long
```

Parameters

pBinName[in]

Pointer to string (bin name) as defined in padll.h

// Defines for legal paper bins for ticket printers
#define PA_CUT_ALL "Cut on all pages"
#define PA_CUT_2ND "Cut every 2nd page"
#define PA_CUT_3RD "Cut every 3rd page"
#define PA_CUT_4TH "Cut every 4th page"
#define PA_CUT_5TH "Cut every 5th page"
#define PA_CUT_LAST "Cut on last page only"
#define PA_NO_CUT "Disable cutting"

Return Values

PA_ERR_OK // Bin Found
PA_ERR_BIN_NOT_FOUND
PA_ERR_ACCESS

C Example Sequence

```
LRESULT lResult;

lResult = paSetBinByBinName(PA_CUT_ALL);
```

Remarks

None

.NET Support

C#,VB.NET example included in CSGetStatusOnlyLM.zip and VBGetStatusOnlyLM.zip. (Practical Automation toolkit).

UPDATE PRINTER FIRMWARE

paLMProgramFlash

This function is used to update printer firmware. The following conditions must exist before attempting to reprogram the printer's firmware.

1 - The printer must be idle and ready with no errors and paper loaded.
2 - The flash file referenced must be in the current path or contain a fully qualified path.

Note 1 - The reprogramming operation may take several minutes. This function will return after the Language Monitor verifies that a valid flash file has been found and the flash programming operation has started. The function will return PA_ERR_OK and pResultString will be set to 'Flash Update Started'. The function paLMGetProgress must be called periodically and the return code evaluated. When the flash programming is complete the function will return complete status.
Please see paLMGetProgress for details.

Note 2 - The parameter ProgramMode determines if the flash update uses a 'smart' or 'dumb' algorithm. The 'smart' algorithm communicates with the printer and determines if the program file is valid for this printer. The 'dumb' algorithm is only used in a case where reprogramming was interrupted preventing the printer from communicating with the programming algorithm.

Note 3 - Please contact Practical Automation regarding an application (written in VB) that demonstrates the firmware upload process in its entirety.

```
LRESULT APIENTRY paLMProgramFlash(Lpsz pFlashFile, BOOL ProgramMode, Lpsz
pResultString, DWORD dwResultStringLength);

Declare Function paLMProgramFlash Lib "PADLL" (ByVal FlashFile As String, ByVal
ProgramMode As Boolean, ByVal ReturnString As String, ByVal ReturnStringLength
As Long) As Long
```

Parameters

pFlashFile[in]
Pointer to flash file name including extension. File must be in system path or fully qualified.

ProgramMode[in]
TRUE program using 'smart mode', FALSE program using 'dumb' mode.

pResultString[in]
Pointer to result string buffer. Upon completion this buffer will be set to 'Flash Update Started' or the reason for failure. Result string buffer should be 1024 bytes long.

dwResultStringLength[in]

Size of result buffer in bytes. The function will truncate the return message if the size of the ResultString buffer is too small to contain the return code.

Return Values

```
The combination of the function return value the result string indicate the
success or failure of the paLMProgramFlash function.

LRESULT (function return value)

PA_ERR_OK                = Successful reprogram operation-started.
                           The buffer pointed to by pResultString
                           will contain 'Flash Update Started'.

PA_ERR_PIPE_OPEN_BUSY    = Pipe was busy (After timeout)
PA_ERR_PIPE_OPEN_FAILED  = Status pipe could not be opened
PA_ERR_SYSTEM            = Flash programming did not start.
                           The string pResultString will indicate
                           reason for lack of flash programming start

pResultString possible values:

   "BIN FILE NOT FOUND"
   "BIN FILE BAD EXTENTION"
   "PRINTER VERSION < 1.20"
   "INCORRECT BIN FILE"
   "PRINTER NOT READY"
   "PRINTER CAN NOT GET MODEL"
   "PRINTER STATUS READ ERROR"
   "PRINTER DID NOT ENTER REPROGRAM MODE"
   "PRINTER PROGRAM FAILED AFTER RETRY"
```


VB Example Sequence

```
Const BUFFER_LENGTH = 1024
Dim ResultString As String * BUFFER_LENGTH
Dim txtBinFileName As String
Dim ConvertedBinName As String
Dim i As Long
Dim lRC As Long

' txtBinFileName set to name and path of PA BIN file for printer.

' All \ must be converted to  \\
ConvertedBinName = ""
For i = 1 To Len(txtBinFileName)
   ConvertedBinName = ConvertedBinName + Mid(txtBinFileName, i, 1)
   If Mid(txtBinFileName, i, 1) = "\" Then
       ConvertedBinName = ConvertedBinName + "\"
   End If
Next i

ResultString = Space(BUFFER_LENGTH)

lRC = paLMProgramFlash(ConvertedBinName, True, ResultString, BUFFER_LENGTH)

' Evaluate lRC, will be PA_ERR_OK if flash program started.
```


paLMGetProgress

This function is used in conjunction with paLMParseFlashStatus to monitor the printer firmware programming process. The call to paLMProgramFlash starts the flash programming process and returns. paLMGetProgress must be called periodically to determine when the printer firmware programming process is complete. The application can use this function to display a firmware update progress bar if desired. The helper function paParseFlashStatus can be used to parse flash programming status into an easy to use structure. The elements of the structure can be used to manage a progress bar. See paLMParseFlashStatus for usage.

```
LRESULT APIENTRY paLMGetProgress (LPSTR ProgressType, PUCHAR  ResultString,
DWORD dwResultStringLength, DWORD * pResultBufferRead);

Declare Function paLMGetProgress Lib "PADLL" (ByVal ProgressType As String,
ByRef pResultBuffer As Byte, ByVal dwResultBufferLength As Long, ByRef
pdwResultBufferRead As Long) As Long
```

Parameters

ProgressType[in]
Pointer type of progress result desired (Currently only 'O' (Flash programs status) is defined..

ResultString[in,out]
Pointer to result string.

dwResultStringLength[in]
Length of ResultString buffer.

pdwResultBufferRead[in,out]
Pointer to DWORD. Function fills in with the number of progress bytes returned in ResultString

Return Values

PA_ERR_OK

VB Example Sequence

See paLMParseFlashStatus

paLMParseFlashStatus

This function is used in conjunction with paLMGetProgress to monitor the printer firmware programming process. The call to paLMProgramFlash starts the firmware programming process and returns. paLMGetProgress must be called periodically to determine when the printer firmware programming process is complete. The application can be used this function to display a firmware progress bar if desired. The function paParseFlash progress can be used to parse flash programming status into an easy to use structure. The elements of the structure can be used to manage a progress bar.

```
LRESULT APIENTRY paLMParseFlashProgress(LPSTR pRawFlashProgress,
PFLASH_PROGRESS pFlashProgress);

Declare Function paLMParseFlashProgress Lib "PADLL" (ByVal pProgressStatus As
String, ByRef pProgStat As ProgressStatus) As Long
```

Parameters

pRawFlashProgress[in]
Pointer to ResultString returned from previous call to paLMGetProgress.

pFlashProgress [in,out]
Pointer to stucture of type PFLASH_PROGRESS (defined in PADLL.H)

```
typedef struct{
   DWORD   state;          // 0 = initial delay (in seconds)
                           // 1 = sending data (in Kilobytes)
                           // 2 = end delay (in seconds)
                           //  3 = firmware update complete
   DWORD   max;            // Maximum value for this state
   DWORD   current         // Current value for this state
   char    PassFail;       // Pass 'P' or Fail 'F'
   char    note[1024];     // Ascii string describing current state info. String will contain programmed version
on when flash programming is complete
   //
}  FLASH_PROGRESS, * PFLASH_PROGRESS;
```

Examples:

Initial Delay:
state = 0 , max = 15 , current = 5 , PassFail = 'P' , note = " " // 5 seconds of 15 second delay complete

Send Data to printer:
state =1 , max = 300 , current = 105 , PassFail = 'P' , note = " " // 105Kb of 300Kb sent to printer

End Delay:
state = 2 , max = 25 , current = 5 , PassFail = 'P' , note = " " // 5 seconds of 25 second delay complete

Firmware update complete:
state = 3 , max = 0 , current = 0 , PassFail = 'P' , note = "V1.03 (12/13/02) " // Firmware upload complete

Return Values

```
PA_ERR_OK

Note that structure members of pFlashProgress will be filled in by the
function.
```

VB Example Sequence

' Start programming sequence (see paLMProgramFlash above)

```
Dim lRC As Long

' Structure for parsing the progress status
Type ProgressStatus
    progressState As Long
    progressMax As Long
    progressCurrent As Long
    progressPassFail As Byte
    progressNote As String * 1024
End Type

Dim strStatus As String
Dim progressStat As ProgressStatus

' Get firmware upload progress to StatusBuffer every n milliseconds
'
ProgStatus = paLMGetProgress(ProgressType, StatusBuffer(LBound(StatusBuffer)),
PA_STATUS_MAX_LENGTH, lReturned)

If (ProgStatus = PA_ERR_OK) Then
  strStatus = left$(StrConv(StatusBuffer, vbUnicode), lReturned)

  ' Parse the status
  lRC = paLMParseFlashProgress(strStatus, progressStat)

  If progressStat.progressPassFail = Asc("P") Then
    If progressStat.progressPassState = 3
        ' Firmware update complete
        Exit Do
    Endif
  Else
    MsgBox ("Flash Program Failed")
      Exit Do
  End If

Else
  Exit Do ' error has occured
Endif
```

PRINTER DRIVER BYPASS FUNCTIONS

PRINTER BYPASS DESCRIPTION

There are times when an application may need to write directly to the installed printer, bypassing the printer driver. This occurs primarily when an application is using a printer that employs the FGL programming language.

Functions are provided to write data to the printer, bypassing the installed printer driver. Data is written to the spooler and the spooler subsystem operates normally. The language monitor status acquisition function is operational.

Two bypass methods are provided. The first method requires that the application open the document for printing, write data to the printer and close the document. The second method combines all 3 functions into a single call. The second method requires that the application write all document data to an array before the bypass print function is called.

Examples:

```
Method #1
bReturn = paPrintBypassOpenPrinter ("Document Name");  // Open document
bReturn = paPrintBypassWritePrinter (pBytes, lLen);    // Write data
                  .........                            // Write more data
bReturn = paPrintBypassWritePrinter (pBytes, lLen);    // Write data
bReturn = paPrintBypassClosePrinter ();                // Close and print

Method #2
bResult = paPrintBypassPrintDocument ("DocumentName", pBytes, dwCount);
```


paPrintBypassOpenPrinter

Open the document (pszDocumentName) to print

```
BOOL APIENTRY paPrintBypassOpenPrinter(char * pszDocumentName);

Declare Function paPrintBypassOpenPrinter Lib "PADLL" (ByVal DocumentName As
String) As Boolean
```

Parameters

pszDocumentName[in]

Pointer to null terminated document name.

Return Values

Successful document open returns TRUE.

paPrintBypassWritePrinter

Write data to printer. Document must be open by previous call to paPrintBypassOpenPrinter. Application can write multiple times to open document.

```
BOOL APIENTRY paPrintBypassWritePrinter(BYTE * pBytes, DWORD lLen);

Declare Function paPrintBypassWritePrinter Lib "PADLL" (ByVal DataToWrite As
String, ByVal BytesToWrite As Long) As Boolean
```

Parameters

pBytes[in]

Pointer to array of bytes to write to printer.

lLen[in]

Number of bytes to write.

Return Values

Successful write to printer returns TRUE.

paPrintBypassClosePrinter

Close the document (pszDocumentName) previously opened. Document will be sent to the printer at this time.

```
BOOL APIENTRY paPrintBypassClosePrinter(void);

Declare Function paPrintBypassClosePrinter Lib "PADLL" () As Boolean
```

Parameters

None.

Return Values

Successful document close returns TRUE.

paPrintBypassPrintDocument

This function incorporates all the functionality necessary to print a document. The application must write all the necessary print data to an array before this function is called.

```
BOOL APIENTRY paPrintBypassPrintDocument(char * pszDocumentName, BYTE * pBytes,
DWORD dwCount);

Declare Function paPrintBypassPrintDocument Lib "PADLL" (ByVal DocumentName As
String, ByVal DataToWrite As String, ByVal BytesToWrite As Long) As Boolean
```

Parameters

pszDocumentName[in]
Pointer to null terminated document name.

pBytes[in]
Pointer to array of bytes to write to printer.

dwCount[in]
Number of bytes to write.

Return Values

Successful document formation returns TRUE.

FGL ROTATED FONT GENERATION

paCreateFont

This function is used to create printer fonts when developing FGL applications in C, C++ and VB.

VB Print Object - Note that the VB print object is not aware of 'rotated fonts'. This can cause printing alignment problems when printing rotated fonts near the edge of ticket when using a Practical Automation FGL language enabled printer. For example printing rotated up characters located at the bottom edge of the ticket will result in a positioning error. The VB object, not understanding rotation believes that the image will not fit on the ticket. VB will generate a new set of X\Y positioning commands different from the intended position. In this case the VB print object must be bypassed and calls made to the Windows create font and print APIs. This DLL provides functions simplify this operation.

C,C++ - The paFont functions provided are thin wrappers around the API font calls. Note however that the paCreateFont call has imbedded font size tables relieving the programmer of the task of manually providing the height and width of the desired font.

The related funtions are:

```
paCreateFont            - Create font for FGL printers
paTextOut               - Text out for FGL printers
paSelectObject          - Select font for FGL printers
paDeleteObject          - Delete font for FGL printers
```

```
HFONT APIENTRY paCreateFont(HDC DeviceContext, char * FontName);

Private Declare Function paCreateFont Lib "PADLL.DLL" (ByVal hdc As Long, ByVal
lpString As String) As Long
```

Parameters

DeviceContext[in]
Handle to printer device context

FontName[in]
pointer to complete desired FGL font name. Ex: "F03 1:1 NR 15 cpi"

Return Values

```
Success:
returns a pointer to a font structure (HFONT)
Fail:
returns NULL
```


VB Example Sequence

```
Const TXT1 = "DLL FGL RU"
Dim NewFont As Long
Dim OldFont As Long

Dim lRet As Long
Dim printer_hdc As Long

' Initialize the printer.

Printer.Orientation = vbPRORLandscape
Printer.ScaleMode = vbPixels

' When we are using api generated fonts we must fool the VB print
' object into thinking something was printing else the end doc will not
' invoke a print.

Printer.Print " "

' Save the printer hDC.
printer_hdc = Printer.hdc

' Create a new font
NewFont = paCreateFont(printer_hdc, "F12 2:2 RU 3.2 cpi" & vbNullChar)

' Select the new font.
OldFont = paSelectObject(printer_hdc, NewFont)

' Draw the text. Note that fonts created with API CreateFont must be
' printed with TextOut (API)
lRet = paTextOut(printer_hdc, 1500, 900, TXT1)

' Restore the original font.
paSelectObject printer_hdc, OldFont

' Release the resources associated with the new font
paDeleteObject NewFont

Printer.EndDoc
```


paTextOut

See paCreateFont for description

```
LRESULT APIENTRY paTextOut(HDC deviceContext, int Xpos, int Ypos, char
*textToWrite);

Private Declare Function paTextOut Lib "PADLL.DLL" (ByVal hdc As Long, ByVal x
As Long, ByVal y As Long, ByVal lpString As String) As Long
```

Parameters

DeviceContext[in]
Handle to printer device context

Xpos[in]
X coordinate for text

Ypos[in]
Y coordinate for text

textToWrite[in]
pointer to zero terminated text to print

Return Values

Success:
Function returns non-zero
Fail:
Function returns zero

paSelectObject

See paCreateFont for description

```
HGDIOBJ APIENTRY paSelectObject(HDC PrinterHdc, HGDIOBJ NewFont);

Private Declare Function paSelectObject Lib "PADLL.DLL" (ByVal hdc As Long,
ByVal hFont As Long) As Long
```

Parameters

DeviceContext[in]
Handle to printer device context

NewFont[in]
Handle to new font created with paCreateFont

Return Values

Returns Handle to old font

paDeleteObject

See paCreateFont for description

```
BOOL APIENTRY paDeleteObject(HGDIOBJ FontToDelete);
Private Declare Function paDeleteObject Lib "PADLL.DLL" (ByVal hFont As Long)
As Boolean
```

Parameters

DeviceContext[in]
Handle to printer device context

FontToDelete[in]
pointer to Font to delete

Return Values

Success:
Function returns non-zero
Fail:
Function returns zero

PRINTER DRIVER REGISTRY SETUP FUNCTIONS

Many printer driver functions can be enabled, disabled or conditioned by adjusting the registry settings associated with the printer. This function is available for printer drivers V2.07 and higher.

paLMSetRegistryKey

Set the value of a registry key. The registry key name must be one of the following valid key names: (Contact factory for details)

```
DRWdebugMask
DRWenable
DRWpollTime
DRWMaxFileSize
DRWMinFileSize
EnableDiags
EnableHeartBeat
EOJEjectSendLastPage
ECPmode
EOJTimeout
HeartBeatRate
JAVAlowPaperDisable
JAVAticketNotTakenDisable
LPmode
ParallelPacketSize
SerialPacketSize
USBAbortOnWriteTimeout
USBPacketSize
USBprintingTimeout
USBReadIntervalTimeout
USBWriteTotalTimeoutConstant
USBReadTotalTimeoutMultiplier
USBWDGlobalEnable
USBReadTotalTimeoutConstant
USBWriteTotalTimeoutMultiplier
USBWriteTotalTimeoutConstantFlashUpdate
USBWDtimeout
USBWDResetPrinterOnCRCError
DeleteAllJobsOnStart
LPLatchEnable
LPLatched
LPLatchThreshold
LPLatchCount
```



```
LRESULT APIENTRY paLMSetRegistryKey(LPSTR RegistryKeyName, DWORD
RegistryValue, LPSTR pResultString, DWORD dwResultStringLength,
DWORD * pResultBufferRead);

Declare Function paLMSetRegistryKey Lib "PADLL" (ByVal RegistryKeyName As
String, ByVal RegistryValue As Long, ByRef pResultBuffer as Byte, ByVal
ReturnStringLength As Long, ByRef pdwResultBufferRead As Long) As Long
```

Parameters

RegistryKeyName[in]
Pointer to Name of registry key to update
RegistryValue[in]
Registry value to set (DWORD)
pResultString[in,out]
Byte pointer to result string
dwResultStringLength[in]
Result string length
pResultBufferRead[in,out]
Pointer to DWORD where number of bytes returned

Return Values

Success:
PA_ERR_OK
Fail:
PA_ERR_PIPE_OPEN_BUSY
PA_ERR_PIPE_OPEN_FAILED
PA_ERR_SYSTEM
PA_ERR_UNKNOWN_KEY

.NET Support

C#,VB.NET example included in CSGetStatusOnlyLM.zip and VBGetStatusOnlyLM.zip. (Practical Automation toolkit).

paSetPrinterKeyByName

Set the value of a printer registry key. The registry key name must be one of
The following valid key names: (Contact the factory for details)

AUTOSIZE
BACKUP_CONTROL
BEEPER_CONTROL
DO_NOT_COMPRESS
ENERGY_LEVEL
ENERGY_TABLE
EOD_CONTROL
JAM_ERROR_CONTROL
LOCKED
LOW_PAPER_ACTIVE_DETECT
LOW_PAPER_ALGORITHMN
MIN_FORM_LENGTH
ROTATE_IMAGE_180
SJO_CONFIG
TNT_CONTROL
TNT_DISPOSE
TNT_EJECT_ALWAYS
USE_IPS_LIMIT

```
LRESULT APIENTRY paSetPrinterKeyByName(LPSTR printerKeyName, DWORD
printerKeyValue);

Declare Function paSetPrinterKeyByName Lib "PADLL" (ByVal
PrinterRegistryKeyName As String, ByVal RegistryValue As Long) As Long
```

Parameters

PrinterRegistryKeyName[in]
Pointer to Name of registry key to update
RegistryValue[in]
Registry value to set (DWORD)

Return Values

Success:
PA_ERR_OK
Fail:
PA_ERR_UNKNOWN_KEY - Could not find the key specified
PA_ERR_BAD_PRINTER_NAME - Could not open printer, probably bad name
PA_ERR_PRINTER_KEY_NOT_FOUND - No such key found

.NET Support

C#,VB.NET example included in CSGetStatusOnlyLM.zip and VBGetStatusOnlyLM.zip. (Practical Automation toolkit).

MISCELLANY

paPulseCashDrawerGPrinters

This function may be called to pulse the cash drawer output on a G type ticket printer. The printer must be in a ready state (Power on with tickets loaded and no error).

```
BOOL APIENTRY paPulseCashDrawerGPrinter (void);

Declare Function paPulseCashDrawer Lib "PADLL" () As Boolean
```

Parameters

None

Return Values

True if function succeeds else False.

Printer must be on line with tickets loaded in order for function to operate properly.

paLMAbortDocument

There may be times when an application needs to abort the document that is printing. The function paLMAbortDocument will cause the Language Monitor to abort the currently printing document. No action is taken if this function is called when a document is not printing.

```
LRESULT APIENTRY paLMAbortDocument (void);

Declare Function paLMAbortDocument Lib "PADLL" () As Long
```

Parameters

None

Return Values

PA_ERR_OK

paLMGetPrinterName

Get the 'friendly' printer name of the printer connected to the currently open language monitor pipe.

```
LRESULT APIENTRY paLMGetPrinterName(char * pResultBuffer, DWORD
dwResultBufferLength, DWORD * pdwResultBufferRead);

Declare Function paLMGetPrinterName Lib "PADLL" (ByRef pResultBuffer As Byte,
ByVal ResultBufferLength As Long, ByRef ResultBufferRead As Long) As Long
```

Parameters

pResultBuffer[in]
Pointer to buffer that will receive printer name.

dwResultBufferLength[in]
Size of ResultBuffer.

pdwResultBufferRead[out]
Number of bytes returned by this function

Return Values

Refer to **FUNCTION RETURN CODES** for details

.NET Support

C#,VB.NET example included in CSGetStatusOnlyLM.zip and VBGetStatusOnlyLM.zip. (Practical Automation toolkit).

paLMGetPortName

Get the name of the port in use by the language monitor. (Example **USB003**)
The name will be returned in the buffer pointed to by pResultBuffer.

```
LRESULT APIENTRY paLMGetPortName(BYTE * pResultBuffer, DWORD
dwResultBufferLength, DWORD * pdwResultBufferRead);

Declare Function paLMGetPortName Lib "PADLL" (ByRef pResultBuffer As
Byte, ByVal dwResultBufferLength As Long, ByRef pdwResultBufferRead As
Long) As Long
```

Parameters

pResultBuffer[in]
Pointer to a byte buffer to receive the null terminated port name.

dwResultBufferLength[in]
The length of the result buffer.

pdwResultBufferRead[in,out]
Number of bytes returned by this function

Return Values

Refer to **FUNCTION RETURN CODES** for details

.NET Support

C#,VB.NET example included in CSGetStatusOnlyLM.zip and VBGetStatusOnlyLM.zip. (Practical Automation toolkit).

paLMGetDriverVersion

Used to recover the printer driver version.

```
BOOL APIENTRY paLMGetDriverVersion(char * pszPrinterName,
char * pszPrinterVersion, DWORD dwPrinterVersionBufferLength,
DWORD * pdwResultBufferRead);

Declare Function paLMGetDriverVersion Lib "PADLL" (ByVal PrinterName As String,
ByRef pResultBuffer As Byte, ByVal dwResultBufferLength As Long,
ByRef pdwResultBufferRead As Long) As Boolean
```

Parameters

pszPrinterName [in]
Pointer to a buffer containing the null terminated name of the printer
pszPrinterVersion[in,out]
Pointer to buffer that receives the null terminated driver number.
dwPrinterVersionBufferLength
Size of PrinterVersion buffer
pdwResultBufferRead[in,out]
Number of bytes returned by this function

Return Values

Success:
TRUE
Fail:
FALSE

.NET Support

C#,VB.NET example included in CSGetStatusOnlyLM.zip and VBGetStatusOnlyLM.zip. (Practical Automation toolkit).

paGetDLLVersion

Used to recover the PADLL.dll version.

```
BOOL APIENTRY paGetDLLVersion(char * pszDLLVersionBuffer, DWORD
dwDLLVersionBufferLength,
DWORD * pdwResultBufferRead);

Declare Function paGetDLLVersion Lib "PADLL" (ByRef pszDLLVersionBuffer As
Byte, ByVal dwDLLVersionBufferLength As Long,
ByRef pdwResultBufferRead As Long) As Boolean
```

Parameters

pszDLLVersionBuffer[in,out]
Pointer to buffer that receives the null terminated version.
dwDLLVersionBufferLength
Size of pszDLLVersionBuffer buffer
pdwResultBufferRead[in,out]
Number of bytes returned by this function

Return Values

Success:
TRUE
Fail:
FALSE

.NET Support

C#,VB.NET example included in CSGetStatusOnlyLM.zip and VBGetStatusOnlyLM.zip. (Practical Automation toolkit).

NEW FUNCTIONS

Check with PA before using these functions, some of which are not released yet.

Delete All Print jobs on startup

If there are print jobs pending in the Spooler when the PC is shut down, these jobs may be printed on power up. Under certain conditions it may be desirable to delete all pending jobs associated with the printer on system. This feature is defaulted off. To enable the feature set the registry key DeleteAllJobsOnStart.
This function is available for printer drivers V2.07 and higher.

```
#define BUFFER_SIZE 256

char  ResultBuffer[BUFFER_SIZE];
DWORD dwResultBufferRead;
Long  Result;

Result = paLMSetRegistryKey("DeleteAllJobsOnStart", // Name of key
                            1,                      // 1 = Enable this function
                                                    // 0 = Disable this function
                            ResultBuffer,
                            BUFFER_SIZE,
                            &dwResultBufferRead);
```

.NET Support

C#,VB.NET example included in CSGetStatusOnlyLM.zip and VBGetStatusOnlyLM.zip. (Practical Automation toolkit).

Low Paper Filter

The printer driver includes a "Low paper filter". When enabled the filter requires that the printer sense low paper for 10 consecutive documents before low paper is asserted in status. Contact Practical Automation for more details regarding the low paper filter. This feature is defaulted off. To enable the feature set the registry key LPLatchEnable. This function is available for printer drivers V2.07 and higher.

```
#define BUFFER_SIZE 256

char  ResultBuffer[BUFFER_SIZE];
DWORD dwResultBufferRead;
Long  Result;

Result = paLMSetRegistryKey("LPLatchEnable",           // Name of key
                            1,                         // 1 = Enable this function
                            ResultBuffer,
                            BUFFER_SIZE,
                            &dwResultBufferRead);
```


Rotate Print Image 180 degrees

The printer driver has the ability to rotate the printed image 180 degrees. (Bottom of document prints first). This is useful when the application desires the printed output to appear "right side up" when exiting the printer. This feature is defaulted off. To enable the feature set the printer registry key ROTATE_IMAGE_180.
This function is available for printer drivers for the ITK38, V2.07 and higher.

```
Long  Result;

Result = paSetPrinterKeyByName("ROTATE_IMAGE_180", // Name of key
                               1);                 // 1 = Enable this function
                                                   // 0 = Disable this function
```


paLMPtrGetStatus

Get printer status for a specific device. This removes the need to call paSetNewPrinter first. And aids in communicating with multiple printers from different threads. This function will collect status based on the type of status requested. The details of each status type returned can be found in the programmer's manual for each printer. The status returned can be parsed into a structure for immediate use. Refer to description of paParseStatus that follows. Note that the function will return **Complete** status if the printer is Busy.

```
LRESULT APIENTRY paLMPtrGetStatus(char *pszPrinterName, DWORD dwType, BYTE *
pResultBuffer, DWORD dwResultBufferLength, DWORD * pdwResultBufferRead);

Declare Function paLMPtrGetStatus Lib "PADLL" (ByVal pszPrinterName As String,
ByVal dwType As Long, ByRef pResultBuffer As Byte, ByVal dwResultBufferLength
As Long, ByRef pdwResultBufferRead As Long) As Long
```

Parameters

pszPrinterName[in]

Pointer to null terminated printer name.

dwType[in]

Type of status requested:

```
0 = Short
1 = Normal
2 = Extended
3 = Ticket Count
4 = Head temperature
5 = Form Length
6 = Complete
```

pResultBuffer[in]

Pointer to a byte buffer to receive the status.
This buffer must be at least PA_RAW_STATUS_MAX_LENGTH bytes long (128)

dwResultBufferLength[in]

The length of the result buffer.

pdwResultBufferRead[In,out]

Number of bytes returned by this function

Return Values

Refer to **FUNCTION RETURN CODES** for details

C Example Sequence

```
STATUS        statPrinter;      // Structure defined in PADLL.H
BYTE          uchStatus[PA_STATUS_MAX_LENGTH];
LRESULT       lResult;
DWORD         lStatRead;

lResult = paLMPtrGetStatus("PA ITK38", PA_STAT_COMPLETE, &uchStatus[0],
sizeof(uchStatus), &lStatRead);

// Now parse the status
lResult = paParseStatus(&uchStatus[0], lStatRead, &statPrinter);
```

Refer to programmer's manual for details of each status field.
See paParseStatus for details of status structure.

---

paParseStatus is generally used to obtain the operational status of the printer. If the requested status type is Short, Normal, Extended or Complete, bytes 3 and 4 of the returned status encode the operational status of the printer. The bit assignments for these 2 bytes are:

// Byte 3

#define PA_PRINTER_NOT_READY	0x80
#define PA_PRINTER_SYS_ERROR	0x40
#define PA_PRINTER_MAN_DESELECT	0x20
#define PA_PRINTER_OUTPUT_JAM	0x10
#define PA_PRINTER_REGISTRATION	0x08
#define PA_PRINTER_CUTTER_JAM	0x04
#define PA_PRINTER_HEAD_LEVER_UP	0x02
#define PA_PRINTER_OUT_OF_PAPER	0x01

// Byte 4

#define PA_DOC_IN_PROCESS	0x80
#define PA_PRINTER_DATA_ERROR	0x40
#define PA_PRINTER_BUSY	0x20
#define PA_PRINTER_TEMP_WAIT	0x10
#define PA_PRINTER_TICKET_NOT_TAKEN	0x08
#define PA_PRINTER_AVG_POWER_WAIT	0x04
#define PA_PRINTER_DISPOSE_TOGGLE	0x02
#define PA_PRINTER_LOW_PAPER	0x01

```
// Where:
// PA_PRINTER_SYS_ERROR AND PA_PRINTER_DATA_ERROR = Purge occurred
// PA_PRINTER_SYS_ERROR AND (NOT PA_PRINTER_DATA_ERROR) = Data error
```

NOTE:

PA_PRINTER_DISPOSE_TOGGLE is a toggle and will not be reset when another print is received

.NET Support

C#, VB.NET example included in CSGetStatusOnlyLM.zip and VBGetStatusOnlyLM.zip. (Practical Automation toolkit).

paLMPtrIsPipe

Determine if the pipe exists for a specific device. This removes the need to call paSetNewPrinter first. And aids in communicating with multiple printers from different threads.

```
BOOL APIENTRY paLMPtrIsPipe(char *pszPrinterName);

Declare Function paLMPtrIsPipe Lib "PADLL" (ByVal pszPrinterName As String) As Boolean
```

Parameters

pszPrinterName[in]

Pointer to null terminated printer name.

Return Values

TRUE if printer pipe can be opened, else FALSE.

.NET Support

C#, VB.NET example included in CSGetStatusOnlyLM.zip and VBGetStatusOnlyLM.zip. (Practical Automation toolkit).

RFID

RFID commands are only supported on some devices. At the time of writing, the ITK22G (RFID) is the only device that supports these commands. Check with PA before using these functions as not all devices will support them.

paLMPtrRFIDPrinter

Our RFID support can be treated as a transparent pass through pipe. The commands that the printer supports being received through this pipe are documented in the AN32nLL9_RFID_SUPPLEMENT.PDF document. This function is the same as paLMRFIDPrinter, except that it removes the need to call paSetNewPrinter first. And aids in communicating with multiple printers from different threads.

```
LRESULT APIENTRY paLMPtrRFIDPrinter(char *pszPrinterName, char *pDataToWrite,
DWORD lNoBytesToWrite, char *pResponseBytes, DWORD lMaxResponseLen, DWORD
*plNoBytesRead, DWORD lTimeout);

Declare Function paLMPtrRFIDPrinter Lib "PADLL" (ByVal pszPrinterName As
String, ByVal DataToWrite As String, ByVal NoBytesToWrite As Long, ByRef
ResponseBytes As Byte, ByVal MaxResponseBytesLen As Long, ByRef NoBytesRead As
Long, ByVal Timeout As Long) As Long
```

Parameters

pszPrinterName[in]
Pointer to null terminated printer name.

pCommandBytes
Pointer to a byte buffer that contains the RFID command that needs to be sent.

lLen
The length of the command supplied in pCommandBytes. The maximum length supported in 1024 bytes.

pResponseBytes
Pointer to a byte buffer to receive the response.

lMaxResponseLen
The length of the response buffer supplied in pResponseBytes. The maximum length supported in 1024 bytes.

plResponseLen
Pointer to a variable that contains the length of the response returned.

lTimeout (FOR FUTURE USE)
The maximum length of time that call should wait for a response from the printer.

Return Values

```
PA_ERR_OK                - Successful
PA_ERR_PARAM1            - Error with pCommandBytes
```

```
PA_ERR_PARAM2            - Error with lLen
PA_ERR_PARAM3            - Error with pResponseBytes
PA_ERR_PARAM4            - Error with lMaxResponseLen or plResponseLen
PA_ERR_PIPE_OPEN_BUSY    - Error opening the pipe/connection to LM
PA_ERR_PIPE_OPEN_FAILED  - Error opening the pipe/connection to LM
PA_ERR_SYSTEM            - Error reading or writing to the pipe/LM
PA_ERR_UNKNOWN_PRINTER   - LM does not support this function
PA_ERR_PIPE_TIMEOUT      - Timeout reading from LM
PA_ERR_PRINTER_NOT_RESPONDING – Printer not responding
```

.NET Support

C#,VB.NET example included in CSGetStatusOnlyLM.zip and VBGetStatusOnlyLM.zip. (Practical Automation toolkit).

paLMRFIDPrinter

Our RFID support can be treated as a transparent pass through pipe. The commands that the printer supports being received through this pipe are documented in the AN32nLL9_RFID_SUPPLEMENT.PDF document.

```
LRESULT APIENTRY paLMRFIDPrinter(char *pDataToWrite, DWORD lNoBytesToWrite,
char *pResponseBytes, DWORD lMaxResponseLen, DWORD *plNoBytesRead, DWORD
lTimeout);

Declare Function paLMRFIDPrinter Lib "PADLL" (ByVal DataToWrite As String,
ByVal NoBytesToWrite As Long, ByRef ResponseBytes As Byte, ByVal
MaxResponseBytesLen As Long, ByRef NoBytesRead As Long, ByVal Timeout As Long)
As Long
```

Parameters

pCommandBytes
Pointer to a byte buffer that contains the RFID command that needs to be sent.

lLen
The length of the command supplied in pCommandBytes. The maximum length supported in 1024 bytes.

pResponseBytes
Pointer to a byte buffer to receive the response.

lMaxResponseLen
The length of the response buffer supplied in pResponseBytes. The maximum length supported in 1024 bytes.

plResponseLen
Pointer to a variable that contains the length of the response returned.

lTimeout (FOR FUTURE USE)
The maximum length of time that call should wait for a response from the printer.

Return Values

```
PA_ERR_OK                - Successful
PA_ERR_PARAM1            - Error with pCommandBytes
PA_ERR_PARAM2            - Error with lLen
PA_ERR_PARAM3            - Error with pResponseBytes
PA_ERR_PARAM4            - Error with lMaxResponseLen or plResponseLen
PA_ERR_PIPE_OPEN_BUSY    - Error opening the pipe/connection to LM
PA_ERR_PIPE_OPEN_FAILED  - Error opening the pipe/connection to LM
PA_ERR_SYSTEM            - Error reading or writing to the pipe/LM
PA_ERR_UNKNOWN_PRINTER   - LM does not support this function
PA_ERR_PIPE_TIMEOUT      - Timeout reading from LM
PA_ERR_PRINTER_NOT_RESPONDING – Printer not responding
```


.NET Support

C#,VB.NET example included in CSGetStatusOnlyLM.zip and VBGetStatusOnlyLM.zip. (Practical Automation toolkit).

REFERENCE

Definition of status bits returned from paLMGetStatus

The function paLMGetStatus is very useful in that it returns comprehensive systems status in bit mapped form. The status bit are encoded in the 4th (high byte) and 5th (low byte) of the return status packet.
Details of the returned status bits:

High Byte

Name	Bit assignment
#define PA_PRINTER_NOT_READY	0x80

Indicates that printer is not ready to print for any reason. User must examine the remaining status bits to determine the reason the printer is not ready.

#define PA_PRINTER_SYSERROR	0x40

See chart below

#define PA_PRINTER_MAN_DESELECT	0x20

The printer has been manually deselected via the select (F0) pushbutton.

#define PA_PRINTER_OUTPUT_JAM	0x10

A paper jam has occurred in the transport presenter mechanism. This is a fatal error.

#define PA_PRINTER_REGISTRATION	0x08

A registration error has occurred. This error only occurs on paper printers if using paper with registration marks and ticket printers. The error indicates that the paper or ticket was not properly advanced to the next registration mark position. This is a fatal error.

#define PA_PRINTER_CUTTER_JAM	0x04

The cutter has failed to initialize or return to home after a cut operation. This is a fatal error.

#define PA_PRINTER_HEAD_LEVER_UP	0x02

The printer head lever is up (the non print position). If this event occurs during printer it is treated as a fatal error otherwise it can be recovered by lowering the printer head lever.

#define PA_PRINTER_OUT_OF_PAPER	0x01

The printer is out of paper. If this is a paper printer this is a fatal error. If this is a ticket printer stock can be reloaded.

Low byte

Name	Bit assignment
#define PA_PRINTER_DOC_IN_PROCESS	0x80

A document is currently being printed

#define PA_PRINTER_DATA_ERROR	0x40

See chart below

#define PA_PRINTER_BUSY	0x20

The printer is busy and cannot receive data.

#define PA_PRINTER_TEMP_WAIT 0x10
Printing is temporarily suspended because the print head is either too cold or too hot.
This is not a fatal error

#define PA_PRINTER_TICKET_NOT_TAKEN 0x08
A ticket is in the transport presenter.

#define PA_PRINTER_AVG_POWER_WAIT 0x04
The data pattern recently printed has reached the thermal capacity of the power supply. This is not a fatal error. The status bit will be cleared and printing may continue after a suitable wait period.
This is not a fatal error.

#define PA_PRINTER_DISPOSE_TOGGLE 0x02
This status bit is only used for printers that have a dispose transport presenter.
Each time a document is disposed the bit will toggle.

#define PA_PRINTER_LOW_PAPER 0x01
The printer low paper sensor has determined that the paper supply is low.

PA_PRINTER_SYSERROR	PA_PRINTER_DATA_ERROR	Meaning
0	0	Okay
0	1	Data Error. An improper code sequence has been sent to the printer.
1	0	Power on diagnostic error
1	1	A document purge has occurred

Definition of return codes returned from DLL functions

I. STATUS COLLECTION RETURN CODES

PA_ERR_OK
DLL function call completed without error. Corresponding data returned (generally status) is valid.

PA_ERR_PRINTER_NOT_RESPONDING
Error occurs when communication between caller and printer driver has functioned correctly but the physical printer has not responded to the status request. This is generally due to:
1 – Printer is powered off
2 – Printer is not connected (cable)

PA_ERR_PIPE_OPEN_FAILED
Error occurs when a status request is directed to a printer\language monitor that has not opened a status pipe. Status pipes are opened by Practical Automation printer drivers version 2.0 and above.
This error indicates one of the following:
1 – Printer driver not installed
2 – Printer driver installed but is not set to default.
Set PracticalAutomation printer to default or use paSetNewPrinter(PrinterName) to point status function to proper printer

PA_ERR_PIPE_OPEN_BUSY
Error occurs when the function paLMOpenPipe has explicitly opened the communication pipe to the printer language monitor but has not called paLMClosePipe. This error will NOT occur when using the PADLL status functions as these functions manage the proper opening and closing of the communications pipe.

PA_ERR_STATUS_TIMEOUT
This error occurs if the function paLMSetStatusTimeOut(StatusTimeOut in Milliseconds) has been used to reduce the time allowed for the printer to return status. The status timeout is defaulted to 20 seconds.
In normal practice there is no reason to shorten this timeout.
It is generally used for test only.

PA_ERR_BAD_STAT_GROUP
The ID field of the status to parse is not SNEC or A. Verify status to parse has been correctly recovered.
paParseStatus(PUCHAR pBytes, LONG lLen, PSTATUS pStat)

II. SET PRINTER PARAMETER RETURN CODES

PA_ERR_OK
DLL function call completed without error.

PA_ERR_UNKNOWN_KEY
An attempt was made to update an unknown printer key. (paSetPrinterKeyByName(name))

PA_ERR_NO_PRINTER_CHANGE
An attempt to update a printer key failed. (paSetPrinterKeyByName(name))

PA_ERR_SHEET_NOT_FOUND
Exact sheet size not found when setting sheet size. Page size was set to the next closest page size.
paSetSheetSize(PFLOAT width, PFLOAT length)

PA_ERR_BIN_NOT_FOUND
Cannot set bin name. Incorrect name specified.
paSetBinByBinName(LPSTR pBinName)

III. GET PRINTER PARAMETER RETURN CODES

PA_ERR_OK
DLL function call completed without error.

PA_ERR_PARAM4
paEnumPrinters, Incorrect PINFO structure specified

PA_ERR_NO_PRINTERS
Returned when calling paEnumeratePrinters and paGetPrinterNameAndPort and no printers installed. Install printer.

PA_ERR_BUFFER_TOO_SMALL
paEnumPrinters buffer size for enumeration too small

PA_ERR_ACCESS
Error occurred when trying to enumerate bins. Contact Practical Automation.

PA_ERR_BLOWN_POINTER
Null pointer passed to paGetCentronicsStatus or paEnumSheetSizes

IV. FIRMWARE UPDATE RETURN CODES

// Error codes for programmatic flash update

PA_ERR_OK
DLL function call completed without error.

PA_ERR_PRINTER_NOT_READY
Firmware update attempted and printer not ready

PA_ERR_BIN_FILE_NOT_FOUND
Firmware update attempted, driver could not find specified BIN file

PA_ERR_BIN_FILE_BAD_EXTENSION
Firmware file does not have extension BIN or FUL

PA_ERR_CAN_NOT_GET_MODEL
Firmware upload failed could not get printer model. Contact Practical Automation.

PA_ERR_STATUS_READ_ERROR
During firmware update driver could not read status during FW upload. Contact Practical Automation.

PA_ERR_INCORRECT_BIN_FILE
Incorrect Firmware file for this printer

PA_ERR_FLASH_NOT_ENTERED
Printer did not enter flash mode. Contact Practical Automation.

PA_ERR_FLASH_BAD_BIN_FILE
Not used …See PA_ERR_INCORRECT_BIN_FILE

PA_ERR_FLASH_REPROGRAM_FAILED
Not used. Note to get pass \ fail status of firmware upload use
paLMParseFlashProgress(LPSTR pRawFlashProgress, PFLASH_PROGRESS pFlashProgress)

```
typedef struct{
   DWORD   state;          // 0 = initial delay, 1 = sending data, 2 = end delay, 3 = complete
   DWORD   max;            // Maximum value for this state
   DWORD   current;        // Current value for this state
   char    PassFail;       // Pass 'P' or Fail 'F'
   char    note[1024];     // Reason for failure or final version
   //
}  FLASH_PROGRESS, * PFLASH_PROGRESS;
```

V. MISC. OS ERRORS RETURN CODES

(Contact Practical Automation)

PA_ERR_SYSTEM
PA_ERR_VAL_NOT_SET
PA_ERR_INSUFFICIENT_MEM

VI. OBSOLETE RETURN CODES

PA_ERR_PIPE_TIMEOUT
PA_ERR_BAD_PRINTER_NAME
PA_ERR_BAD_PORT_NAME
PA_ERR_STRING_TOO_LONG
PA_ERR_PARAM1
PA_ERR_PARAM2
PA_ERR_PARAM3
PA_ERR_REASSIGN
PA_ERR_OS_UNSUPPORTED
PA_ERR_PORT_BUSY
PA_ERR_BAD_STAT_FIELD
PA_ERR_UNKNOWN_PRINTER
PA_ERR_PORT_HAS_SPOOLED_JOB

TROUBLESHOOTING

.NET Application Permissions

Windows 10 Limits access to printer by different users. The PADLL.DLL requires Manage permissions in order to adequately work with the PA printer driver. To set this permission, open the printers "Printer Properties" and on the security tab, ensure that "ALL APPLICATION PACKAGES" has "Manage this printer" permission set to Allow.