



CONFIDENTIAL

# SDK User Guide

*Software Development Kit for Magicard Printer Drivers*

© MAGICARD LIMITED, 2019

This document contains confidential and proprietary information belonging to Magicard Limited and may not wholly or partially be copied, stored in a data retrieval system, disclosed to third parties or used for any purpose other than that for which it was supplied without the express written authority of Magicard Limited.

Any enquiries about this document or its contents should be made in the first instance to:

Applications Manager  
Magicard Ltd.  
Waverley House  
Hampshire Road  
Granby Estate  
Weymouth  
Dorset  
DT4 9XD  
United Kingdom

Tel: (01305) 470000

# Table of Contents

<b>1</b>	<b>Overview .....</b>	<b>1</b>
1.1	Compatibility .....	1
1.1.1	Operating System.....	1
1.1.2	MagAPI.dll .....	1
1.1.3	Printer Driver .....	2
1.1.4	Firmware.....	2
1.2	32/64 Bit Support.....	3
1.2.1	Packing/Alignment.....	4
1.3	Function Descriptions .....	4
1.4	Printers Supported.....	4
1.5	Shim Interfaces.....	5
1.6	Sample Code.....	5
<b>2</b>	<b>Session Control Functions .....</b>	<b>7</b>
2.1	ID_OpenSession .....	7
2.2	ID_CloseSession .....	8
<b>3</b>	<b>Printer Control Functions .....</b>	<b>9</b>
3.1	ID_FeedCard .....	9
3.2	ID_EjectCard .....	10
3.3	ID_MoveCard .....	11
3.4	ID_WaitForPrinter.....	12
3.5	ID_GeneralCommand.....	13
3.6	ID_FlipCard .....	14
3.7	ID_CleanPrinter .....	15
3.8	ID_RestartPrinter.....	16
3.9	ID_PrintTestCard.....	17
3.10	ID_EraseCard.....	18
3.11	ID_ErrorResponse.....	19
3.12	ID_PrepareForPrint.....	20
3.13	ID_WaitPrintComplete .....	21
<b>4</b>	<b>Information Functions .....</b>	<b>22</b>
4.1	ID_PrinterType .....	22
4.2	ID_PrinterModel.....	23
4.3	ID_ConnectionType.....	24
4.4	ID_PrinterStatus .....	25
4.5	ID_PrinterInfo .....	26
4.6	ID_LastMessage.....	29

4.7	ID_Temperature .....	30
4.8	ID_SDKVersion .....	31
4.9	ID_SDKBits.....	32
<b>5</b>	<b>Driver Settings Functions.....</b>	<b>33</b>
5.1	ID_PrintSettings.....	33
5.2	ID_CardSettings .....	34
5.3	ID_HoloKote .....	35
5.4	ID_HoloPatch .....	37
5.5	ID_AreaHole .....	38
5.6	ID_ResinOptions .....	40
5.7	ID_ResinArea .....	42
5.8	ID_PrintableArea .....	43
5.9	ID_ColourCorrection .....	44
5.10	ID_ColourArea .....	45
5.11	ID_ColourAdjust .....	46
5.12	ID_Sharpness.....	47
5.13	ID_PrintSpeed .....	48
5.14	ID_PowerLevel .....	49
5.15	ID_Rewritable .....	50
5.16	ID_RewritableArea.....	52
5.17	ID_Control .....	53
5.18	ID_GUIControl .....	54
5.19	ID_Resolution .....	55
<b>6</b>	<b>Printer Settings Functions.....</b>	<b>56</b>
6.1	ID_CardWidth .....	56
6.2	ID_CardHeight.....	57
6.3	ID_HandFeed .....	58
6.4	ID_EjectMode .....	59
6.5	ID_HorzEject .....	60
6.6	ID_SmartMode .....	61
6.7	ID_SmartOffset.....	62
6.8	ID_EraseSpeed .....	63
6.9	ID_Password .....	64
6.10	ID_IPSettings.....	65
6.11	ID_CardLocation.....	66
6.12	ID_HoloKoteIdentity.....	67
6.13	ID_HoloKoteCount.....	68

6.14	ID_HoloKotePreview.....	69
6.15	ID_Sensors.....	71
<b>7</b>	<b>Generation 2 Printer Settings Functions .....</b>	<b>72</b>
7.1	Overview .....	72
7.2	ID_AccessInt .....	74
7.3	ID_AccessBool .....	75
7.4	ID_AccessBuffer.....	76
7.5	ID_AccessString .....	77
<b>8</b>	<b>Magnetic Encoding Functions.....</b>	<b>78</b>
8.1	Overview .....	78
8.2	ID_EncodeMag.....	79
8.3	ID_ReadMag .....	81
8.4	ID_ReadMagTracks.....	83
8.5	ID_MagStart .....	85
<b>9</b>	<b>Printing Functions.....</b>	<b>86</b>
9.1	Drawing Canvas .....	86
9.2	High Resolution (600 DPI) Printing .....	87
9.2.1	Canvas Size .....	87
9.2.2	Selection.....	87
9.3	ID_CanvasInit.....	88
9.4	ID_CanvasDelete.....	89
9.5	ID_DrawText.....	90
9.7	ID_DrawShape .....	92
9.9	ID_DrawLine.....	94
9.11	ID_DrawImage.....	95
9.13	ID_DrawMagText.....	97
9.14	ID_PrintCard.....	98
<b>10</b>	<b>Printing Support Functions .....</b>	<b>99</b>
10.1	ID_GetDevmode.....	99
10.2	ID_SetDevmode .....	100
10.3	ID_UpdateDC .....	101
10.4	ID_UpdateDevmode .....	102
10.5	ID_PrinterPrefs .....	103
<b>11</b>	<b>Acknowledgements.....</b>	<b>104</b>
	<b>Appendix A - API Functions .....</b>	<b>105</b>

## Figures

Figure 1 – SDK Architecture .....	3
Figure 2 – Typical Application Flowchart .....	6
Figure 3 – Image Bit Assignment.....	70
Figure 4 – Canvas Dimensions & Orientation .....	86

## Tables

Table 1 – Generation 2 Parameter Identifiers .....	73
--	----

# 1 OVERVIEW

The purpose of the Software Development Kit (SDK) is to allow applications to:

- Initiate and control printer operations
- Synchronize the operation of the printer
- Interrogate and change settings of the printer
- Interrogate and change settings of the driver
- Perform basic printing operations

For example, an application may need to know when the printer has finished feeding a card into the encoding position. Without feedback from the printer, it is not possible to retrieve such information, but with the SDK this feedback is available, and an application is able to wait until the printer has completed the operation.

Specific functions are also available to control the printer e.g. place cards in the correct position for contact or contactless chip encoding, to eject a card from the printer.

The process of controlling the printer using the SDK must be serial with all actions on a card being completed before the application can proceed to handle the next card. This means it is not possible to load multiple print jobs and then control their flow to the printer by SDK calls. In this situation, the commands would not occur at the correct time in the batch sequence, and so would be out of synchronization with the print they were trying to control, thus control would be lost.

Therefore, for each card, the SDK should be used to control card positioning, then encoding should take place, and finally the card should be printed (either via the SDK and/or via Windows GDI interface) before finally the card is ejected. The SDK can then continue and position the card for the next job, and so on.

The flow of information between the application and the driver and/or printer is shown in Figure: 2

## 1.1 Compatibility

### 1.1.1 Operating System

This SDK has been tested and approved as compatible with Windows 7, 8 and 10.

### 1.1.2 MagAPI.dll

The SDK operates via a dynamic link library - MagAPI.dll - which must be V3.0.0.0 or later.

**N.B.** Ultima functionality is included from V3.0.2.0

Generation 2 printer functionality is included from V3.0.5.0

The MagAPI.dll file is placed on the client PC during installation of the Magicard Printer Driver. It is placed in the following locations: -

- 32-bit operating system:
  - Windows\System32
- 64-bit operating system:
  - Windows\System32 (64-bit dll)
  - Windows\SysWOW64 (32-bit dll)

### 1.1.3 Printer Driver

Minimum printer driver versions are:

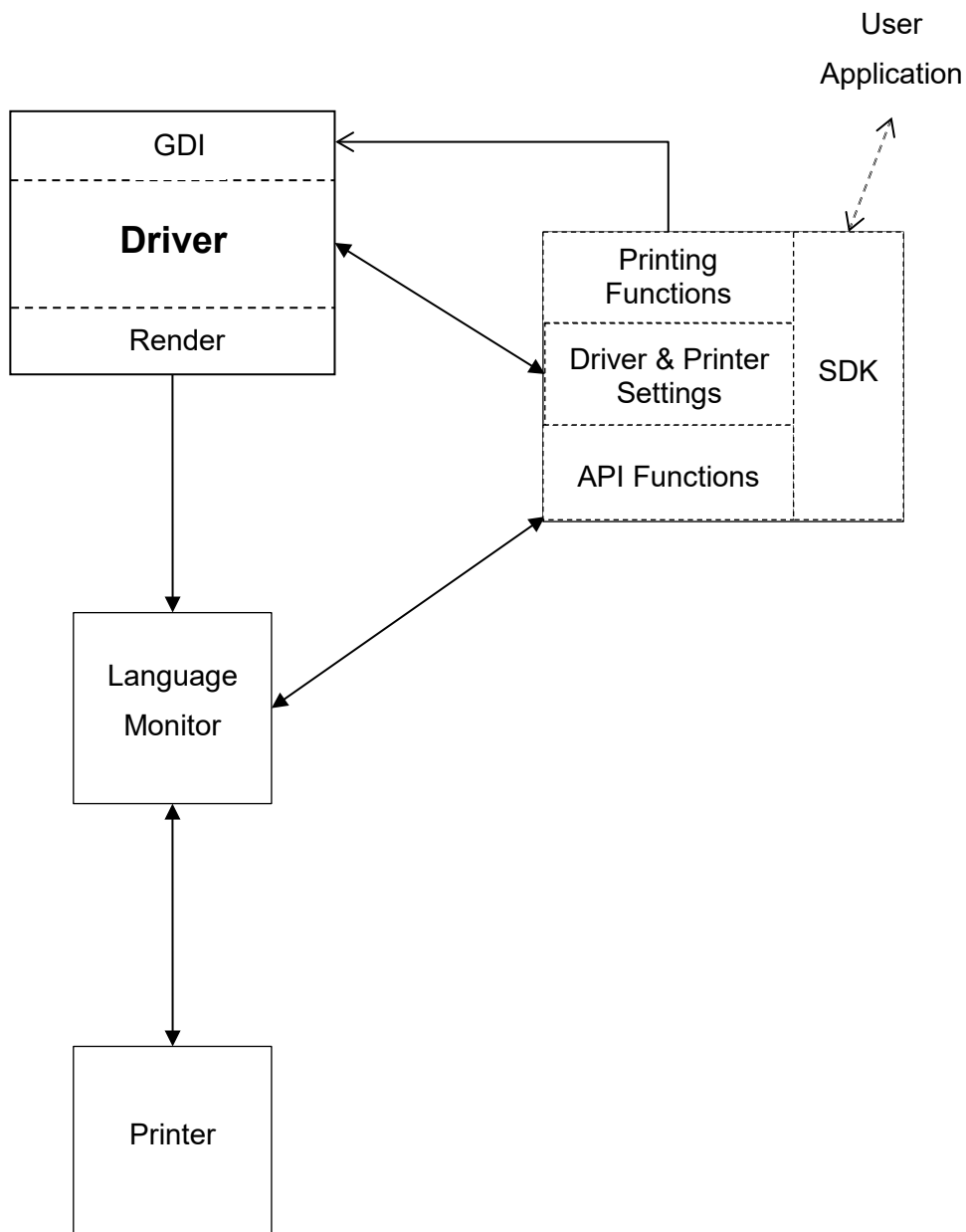
Enduro	<b>2.0.17</b>
Ultima	<b>2.0.25</b>
Generation 2	<b>2.0.35</b>

### 1.1.4 Firmware

Minimum printer firmware versions are:

Enduro	<b>7.18</b>
Pronto	<b>3.14</b>
Rio Pro	<b>2.25</b>
Ultima	<b>2.01</b>
Generation 2	<b>7.23</b>





**Figure 1 – SDK Architecture**

## **1.2 32/64 Bit Support**

Both 32 and 64 bit versions of the SDK are provided. The 32-bit DLL will operate on a 64-bit Windows OS in conjunction with a 32-bit application and must be installed in the SYSWOW64 folder.

With a 64-bit application, and particularly if a Java application is being run under a Java Virtual Machine on a 64-bit OS, the 64-bit DLL should be used and installed in the SYSTEM32 folder.

### 1.2.1 Packing/Alignment

Default packing and alignment for the selected platform are used for both 32 and 64 bit DLLs.

## 1.3 Function Descriptions

The functions provided by the SDK are split into the following logical groups:

- Session Control
- Printer Control
- Information
- Driver Settings
- Printer Settings
- Magnetic Encoding
- Printing

Each function is described together with its input parameters and return values. The calling interface and any structures which may be required are specified using C/C++ syntax.

All abstract types (e.g. HANDLE, HDC, etc.) conform to the standards for Windows development as defined by Microsoft.

## 1.4 Printers Supported

The SDK supports the following printers:

- Enduro family (Enduro; Enduro+; Enduro3e; Pronto; Rio Pro)
- Ultima reverse transfer printer
- Generation 2 family (Rio Pro 360; 300; 600, D, K)

Due to the difference in the printing technology used in Direct to Card (DTC) and Reverse Transfer printers, there are distinct functions provided. However, since the Magicard printer driver provides a common GUI, many functions are also common.

The section for each individual SDK function indicates whether the function is supported by Enduro family, Ultima, or Generation 2 family.

e.g.

Enduro	✓
Ultima	✓
Generation 2	✓

OR

Enduro	✓
Ultima	✗
Generation 2	✗

Key:

✓	= Supported.
✗	= Not supported.
✓	= Not supported, but will not generate an error.
✓	= Deprecated function (see remarks section)

## **1.5 Shim Interfaces**

To allow the use of the SDK with managed code (i.e. .Net), a shim DLL (SDKShim.dll) is provided. This can be included as a reference in .Net projects to provide an interface between .Net and the unmanaged code of the SDK. It is used in both the C# and Visual Basic demo applications.

Similarly, for Java developers, a Java Shim package (SDK\_JShim.jar) is provided which can be built into a project to provide a Java interface to the SDK.

The function prototypes detailed in this user guide are the basic function prototypes as defined in MagAPI.h and used in C/C++ applications. Derivatives are used, together with appropriately defined types, in the .Net and Java environments, details of which can be seen when referencing the shims in the user's application.

## **1.6 Sample Code**

Sample code is available in the form of demo applications in:

- C/C++
- C#
- Visual Basic
- Java.

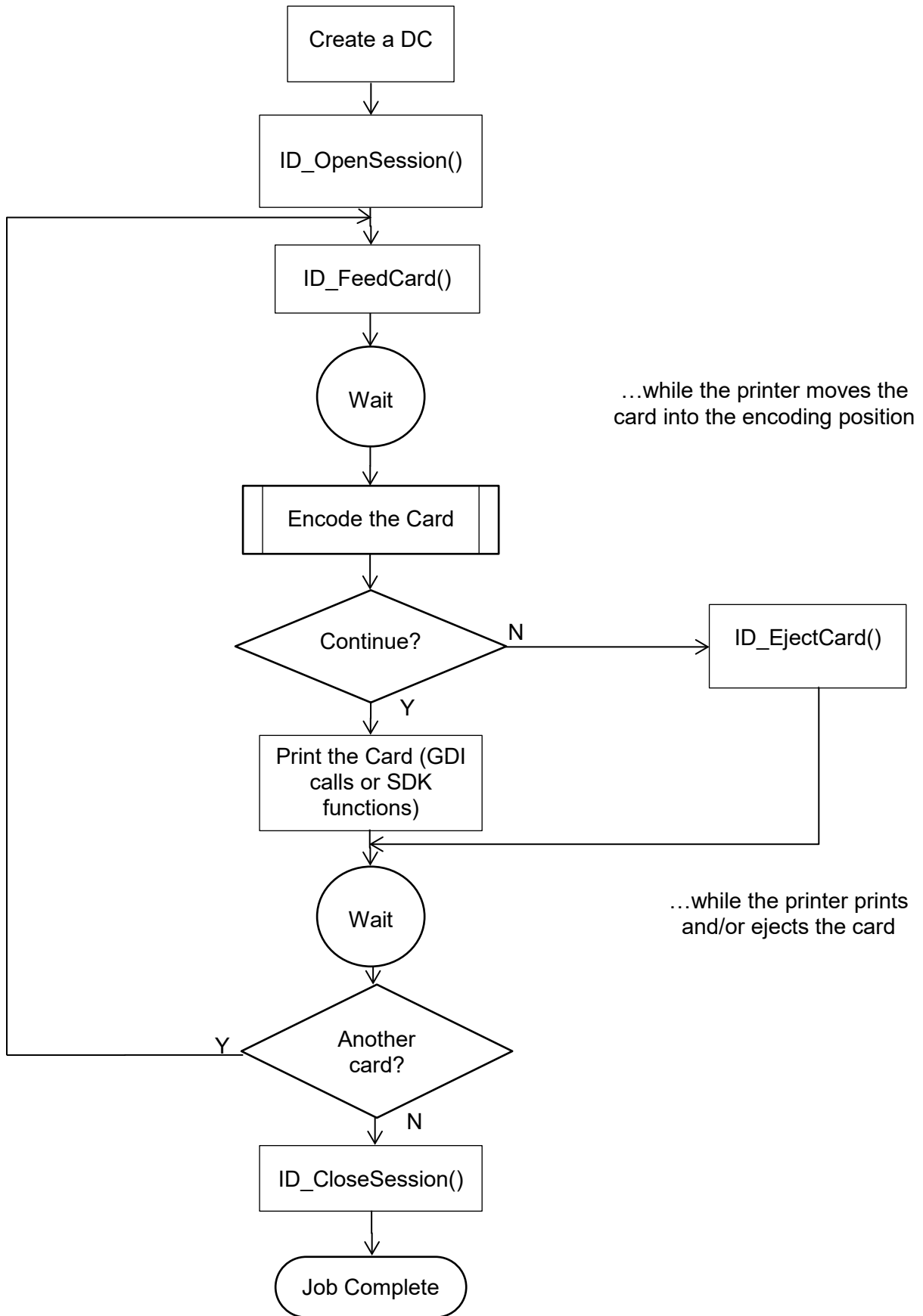
These applications exercise the full range of functions provided by the SDK, which also gives the added advantage of enabling them to be used as a test tool for the SDK.

Due to the differences in the functionality provided by the printers supported, and to provide clarity, separate suites of demo apps have been produced for each printer technology i.e. one for Direct to Card (DTC) printers, another for the Ultima retransfer printer.

The source code provided, with samples and demos, illustrates the usage of each given function and has been deliberately simplified. The additional steps required for a production-quality application (such as security checks, input validation, and error handling) have been omitted to enhance clarity.

Basic printing examples are given using the SDK and native printing techniques for that language, selectable within the demo program.

All code is provided 'AS IS', without any express or implied warranty, and may be subject to change without notice.



**Figure 2 – Typical Application Flowchart**

## 2 SESSION CONTROL FUNCTIONS

### 2.1 ID\_OpenSession

Enduro	✓
Ultima	✓
Generation 2	✓

Initialises the SDK session and its communications channel with the printer.

```
int ID_OpenSession(HDC      hDC,
                  HANDLE *  phSession,
                  DWORD     dwFlags);
```

```
int ID_OpenSession(string  Printername,
                  HANDLE *  phSession,
                  DWORD     dwFlags);
```

**N.B. This overload is only available in the Shim & Java Shim**

#### Parameters

*hDC*

A device context (DC) handle for the required printer.

*Printername*

A string defining the required printer

*phSession*

A pointer to a variable that will receive a handle that identifies the newly established session. All subsequent SDK function calls must use this handle.

*dwFlags*

Defines how the status monitor handles errors for the session:

- ID\_CONFIG\_NORMAL    Normal status monitor operation, displaying printer errors
- ID\_CONFIG\_QUIET    No status monitor displayed

#### Return Values

- ID\_SUCCESS                    Operation completed successfully.
- ID\_PARAM\_ERROR                Invalid parameter
- ID\_DRIVER\_NOTCOMPLIANT       Driver version does not support the API/SDK.
- ID\_OPENPRINTER\_ERROR         Failed to open the printer for the given DC.
- ID\_SPOOLER\_NOT\_EMPTY         Print jobs are queued for this device.
- ID\_SESSION\_ACTIVE             SDK is already in use
- ID\_ERROR                      Other SDK error

#### Remarks

- For the SDK to control and synchronise the printer correctly, all prior print jobs must be complete i.e. the print spooler must be empty before calling this function. Otherwise, the function will fail with the ID\_SPOOLER\_NOT\_EMPTY error code.
- Only one SDK session may be active at one time.
- On completion, the session **must** be closed using the *ID\_CloseSession* function.

## 2.2 ID CloseSession

Closes the SDK session, returns the status monitor to its normal error mode, and releases all resources used.

```
int ID_CloseSession(HANDLE hSession);
```

Enduro	✓
Ultima	✓
Generation 2	✓

### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

### Return Values

ID_SUCCESS	Operation completed successfully.
ID_INVALID_SESSION	Invalid session handle
ID_ERROR	Other SDK Error

## 3 PRINTER CONTROL FUNCTIONS

### 3.1 ID FeedCard

Instructs the printer to feed a card to the location specified.

```
int ID_FeedCard(HANDLE hSession,
               int     dwMode,
               int     iParam);
```

```
int ID_FeedCard(HANDLE hSession); (.Net and Java Shims only)
```

Enduro	✓
Ultima	✓
Generation 2	✓

#### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

*dwMode*

The card type

```
ID_FEED_CHIPCARD
ID_FEED_CONTACTLESS
```

*iParam*

An integer parameter that modifies the feed card position (Range: 0-99)

#### Return Values

ID_SUCCESS	Operation completed successfully.
ID_INVALID_SESSION	Invalid session handle
ID_INVALID_ACTION	Printer is not equipped for this action
ID_PARAM_ERROR	Invalid parameter
ID_ERROR	Other SDK error

#### Remarks

- Deprecated command for Ultima and Rio Pro 360 family printers - use *ID\_MoveCard* instead (see section 3.3).
- In Enduro family, *dwMode* and *iParam* are no longer used and location is specified by the Smart Mode setting e.g. Default (in the encoder) or Platen (see section 6.6). Accordingly, the first definition of this function is now overloaded in the .Net and Java shims with the second definition.
- Should be followed by use of the *ID\_WaitForPrinter* function to allow completion of the printer action (N.B. Not required for Generation 2).

### 3.2 ID EjectCard

Instructs the printer to eject any card in the mechanism.

```
int ID_EjectCard(HANDLE hSession);
```

#### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

Enduro	✓
Ultima	✓
Generation 2	✓

#### Return Values

ID_SUCCESS	Operation completed successfully.
ID_INVALID_SESSION	Invalid session handle
ID_ERROR	Other SDK error

#### Remarks




- Deprecated command for Ultima and Rio Pro 360 family printers - use ID\_MoveCard instead.
- Should be followed by use of the ID\_WaitForPrinter function to allow completion of the printer action (N.B. Not required for Generation 2).



### 3.3 ID MoveCard

Instructs the printer to feed a card to the specified position.

```
int ID_MoveCard(HANDLE hSession,
                DWORD Position);
```

Enduro	
Ultima	
Generation 2	

#### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

*Position*

Value that indicates the card destination

	Ultima	Rio Pro 360	
ID_TRANSFER_STANDBY	✓		Move the card to the standby position
ID_STANDBY		✓	
ID_HEATED_ROLLER	✓		Move the card to the heated roller position
ID_FLIP_STANDBY	✓	✓	Flip the card and return to the standby position
ID_ROTATE	✓		Rotate the card and return to the transfer standby position
ID_MAG_ENCODER	✓		Move the card to the mag encoder
ID_CHIP_ENCODER	✓	✓	Move the card to the contact smart encoder
ID_CONTACTLESS_ENCODER	✓	✓	Move the card to the contactless smart encoder
ID_EJECT	✓	✓	Eject the card
ID_REJECT	✓		Reject the card to the internal card reject bin
ID_LAMINATOR	✓		Move the card to the laminator
ID_INITIALISE		✓	Clear the printer of any cards

#### Return Values

ID_SUCCESS	Operation completed successfully.
ID_INVALID_SESSION	Invalid session handle
ID_PARAM_ERROR	Invalid parameter
ID_INVALID_PRINTER	Invalid for this printer type
ID_INVALID_ACTION	Printer is not equipped for this action
ID_ERROR	Other SDK error

#### Remarks

- Should be followed by use of the *ID\_WaitForPrinter* function to allow completion of the printer action (N.B. Not required for Generation 2).
- When used with a printer that has an exit sensor fitted, the *ID\_REJECT* command checks whether there is a card present in the exit sensor. If so, It pulls that card back into the printer, and then rejects it.

### 3.4 ID WaitForPrinter

Waits until the status monitor reports that the printer is ready or a time-out period elapses.

```
int ID_WaitForPrinter(HANDLE hSession);
```

Enduro	✓
Ultima	✓
Generation 2	

#### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

#### Return Values

ID_SUCCESS	Operation completed successfully.
ID_INVALID_SESSION	Invalid session handle
ID_ERROR	Other SDK Error
ID_TIMEOUT	A 30-second period has elapsed without receiving any status information from the status monitor.
ID_PRINTER_ERROR	The printer has aborted the operation, due to an error.

#### Remarks

- Generation 2 family:**  
This is **NOT** required for the Generation 2 family printers for any action, due to the different communications protocol being used. However, pre-existing applications using this function will not fail, it will simply return `ID_SUCCESS`.
- All Other Printers**  
This function should be used after any SDK function that causes a physical action by the printer which must complete before continuing. e.g.:

```
ID_FeedCard
ID_EjectCard
ID_MoveCard
ID_PrintCard
```

It should also be used whilst waiting for a print job to complete when printing via the Windows GDI interface.

This function:

- exits upon the printer becoming ready, or on an error.
- may timeout during lengthy operations (the timeout length is fixed at approx. 30 seconds). It is up to the application to determine how long it is going to wait, by repeating calls to the function, before deciding that the printer is not responding.

If a printer error is reported, the application can use the *ID\_LastMessage* function to retrieve the error condition at the printer.

### 3.5 ID\_GeneralCommand

This function allows the application to send a command string to the printer.

```
int ID_GeneralCommand(HANDLE hSession,
                     LPTSTR pCommandString);
```

Enduro	✓
Ultima	
Generation 2	

#### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

*pCommandString*

The command string to be sent to the printer.

#### Return Values

ID_SUCCESS	Operation completed successfully.
ID_INVALID_SESSION	Invalid session handle
ID_PARAM_ERROR	Invalid parameter
ID_INVALID_PRINTER	Invalid for this printer type
ID_ERROR	Other SDK error

#### Remarks

- This function is for sending printer commands that are NOT available as discrete functions within this SDK. If you require a command that is not listed in this document, please contact the Magicard Applications department for further information.
- This command is unidirectional to the printer only i.e. it does not support or handle any return of data by the printer.
- Both ASCII and Unicode versions of this function exist (*ID\_GeneralCommandA* and *ID\_GeneralCommandW* respectively). The appropriate function is selected for the user dependent on the Unicode setting of the application, thus the user need only access the generic function.

### 3.6 ID FlipCard

Instructs the printer to reverse the card using the flipper mechanism.

```
int ID_FlipCard(HANDLE hSession);
```

Enduro	✓
Ultima	✓
Generation 2	✓

#### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

#### Return Values

ID_SUCCESS	Operation completed successfully.
ID_INVALID_SESSION	Invalid session handle
ID_INVALID_ACTION	Printer is not equipped for this action
ID_ERROR	Other SDK error

#### Remarks

- Should be followed by use of the *ID\_WaitForPrinter* function to allow completion of the printer action (N.B. Not required for Generation 2).
- This command is not valid on single sided printers - *ID\_PrinterModel* can be used to determine if the printer is single or double sided.

### 3.7 ID CleanPrinter

Instructs the printer to execute a cleaning cycle.

```
int ID_CleanPrinter(HANDLE hSession);
```

Enduro	✓
Ultima	✓
Generation 2	

#### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

#### Return Values

ID_SUCCESS	Operation completed successfully.
ID_INVALID_SESSION	Invalid session handle
ID_ERROR	Other SDK error

#### Remarks

- Should be followed by use of the *ID\_WaitForPrinter* function to allow completion of the printer action (N.B. Not required for Generation 2).
- The cleaning cycle begins immediately that this command is received. Prior to sending the command, the user **MUST** be prompted (by the application) to remove the card hopper and ribbon from the printer, and to prepare a cleaning card for insertion into the printer.

### 3.8 ID RestartPrinter

Instructs the printer to perform a reinitialisation (warm reset).

```
int ID_RestartPrinter(HANDLE hSession);
```

Enduro	✓
Ultima	✓
Generation 2	✓

#### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

#### Return Values

ID_SUCCESS	Operation completed successfully.
ID_INVALID_SESSION	Invalid session handle
ID_ERROR	Other SDK error

#### Remarks

- Should be followed by use of the *ID\_WaitForPrinter* function to allow completion of the printer action (N.B. Not required for Generation 2).

### 3.9 ID PrintTestCard

Instructs the printer to print the internal test pattern.

```
int ID_PrintTestCard(HANDLE hSession);
```

Enduro	✓
Ultima	✓
Generation 2	✓

#### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

#### Return Values

ID_SUCCESS	Operation completed successfully.
ID_INVALID_SESSION	Invalid session handle
ID_ERROR	Other SDK error

#### Remarks

- Should be followed by use of the *ID\_WaitForPrinter* function to allow completion of the printer action (N.B. Not required for Generation 2).

### 3.10 ID\_EraseCard

Instructs the printer to perform an erase on a rewritable card. The area to be erased can be the whole card, or just a specified area of the card.

```
int ID_EraseCard(HANDLE hSession,
                DWORD Count,
                DWORD iBottomLeftX,
                DWORD iBottomLeftY,
                DWORD iTopRightX,
                DWORD iTopRightY)
```

Enduro	✓
Ultima	✗
Generation 2	✓

#### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

*Count*

The number of cards to be erased

*iBottomLeftX*

*iBottomLeftY*

*iTopRightX*

*iTopRightY*

Co-ordinates of the bottom left and top right corners of the area on the card to be erased. If all four co-ordinates are set to 0, the whole card is erased

#### Return Values

ID_SUCCESS	Operation completed successfully.
ID_INVALID_SESSION	Invalid session handle
ID_INVALID_PRINTER	Invalid for this printer type
ID_PARAM_ERROR	Invalid parameter
ID_ERROR	Other SDK error

#### Remarks

- Should be followed by use of the *ID\_WaitForPrinter* function to allow completion of the printer action (N.B. Not required for Generation 2).



### 3.11 ID\_ErrorResponse

Sends a response to an error condition on the printer.

```
int ID_ErrorResponse (HANDLE hSession,
                    int iParam)
```

Enduro	✓
Ultima	✓
Generation 2	✓

#### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

*iParam*

The error response to be sent

- ID\_RESPONSE\_CAN 'Cancel' Response
- ID\_RESPONSE\_OK 'Ok' Response

#### Return Values

- ID\_SUCCESS Operation completed successfully.
- ID\_INVALID\_SESSION Invalid session handle
- ID\_PARAM\_ERROR Invalid parameter
- ID\_ERROR Other SDK error

### 3.12 ID PrepareForPrint

Prepares the Rio Pro 360 family printer to do a print job.

```
int ID_PrepareForPrint(HANDLE hSession)
```

Enduro	
Ultima	
Generation 2	✓

#### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

#### Return Values

ID_SUCCESS	Operation completed successfully.
ID_INVALID_SESSION	Invalid session handle
ID_PARAM_ERROR	Invalid parameter
ID_ERROR	Other SDK error

#### Remarks

This function is used in conjunction with *ID\_WaitPrintComplete* to provide Generation 2 family printers with a similar functionality to that provided for Enduro family printers by *ID\_WaitForPrinter*. It is called before the print is performed, then *ID\_WaitPrintComplete* is used after the print is initiated. This then returns once the print action is over.

### 3.13 ID\_WaitPrintComplete

Waits for the Generation 2 family printer to complete the print job it is performing.

```
int ID_WaitPrintComplete(HANDLE hSession)
```

Enduro	
Ultima	
Generation 2	✓

#### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

#### Return Values

ID_SUCCESS	Operation completed successfully.
ID_INVALID_SESSION	Invalid session handle
ID_PARAM_ERROR	Invalid parameter
ID_ERROR	Other SDK error

#### Remarks

Used in conjunction with *ID\_PrepareForPrint* to provide Generation 2 family printers with a similar functionality to that provided for Enduro family printers by *ID\_WaitForPrinter*.

Having called *ID\_PrepareForPrint* before starting the print, *ID\_WaitPrintComplete* is used, which returns once the print action is over.

N.B. It can only be used in conjunction with *ID\_PrepareForPrint*. Action(s) if used alone will be unpredictable.

## 4 INFORMATION FUNCTIONS

### 4.1 ID PrinterType

Returns an identifier of the printer family.

```
int ID_PrinterType(HANDLE hSession);
```

Enduro	✓
Ultima	✓
Generation 2	✓

#### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

#### Return Values

ID_RIOTANGO	Rio/Tango Family (Rio, Tango, X-Series)
ID_AOTA	AOTA Family (Alto, Opera, Temp, Avalon)
ID_ENDURO	Enduro Family (Enduro, Pronto, Rio Pro and OEM derivatives)
ID_ULTIMA	Ultima Printer
ID_GENERATION2	Generation 2 Family (Rio Pro 360, 300, 600, D, K, and OEM derivatives)
ID_ERROR	Other SDK error

#### Remarks

The return values for RioTango and AOTA are legacy values for identification only, and the SDK **does not** support these printer families.

## 4.2 ID PrinterModel

Returns the printer model

```
DWORD ID_PrinterModel(HANDLE hSession);
```

Enduro	✓
Ultima	✓
Generation 2	✓

### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

### Return Values

Bitmask defining the model and capability of the printer

Bit	Model	Function
0		Magnetic Stripe Encoding
1		Duplex Printing
2		Rewritable Card Printing
3		Contactless Encoding (Ultima Only)
4		Chip Encoding
5	Pronto	
6		
7		
8		Extended Card Printing
9	Enduro	
10		
11		Ethernet
12	Ultima	
13		Laminator (Ultima Only)
14		600 dpi (600, D, K Models Only)
15	Rio Pro	
16	Rio Pro 360	
17	600 Model	
18	300 Model	
19	Other	
	Generation 2	
Bits 20-31 are unused		

ID_INVALID_SESSION	Invalid session handle
ID_ERROR	Other SDK error

### Remarks

- Some printers are defined by a combination of bits e.g. Rio Pro Xtended = 33024 (8100 hex – Bit 15 + Bit 8)

### 4.3 ID\_ConnectionType

Returns the type of the connection to the printer

```
int ID_ConnectionType(HANDLE hSession);
```

#### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

Enduro	✓
Ultima	✓
Generation 2	✓

#### Return Values

ID_PORT_USB	USB connection
ID_PORT_ETHERNET	Ethernet connection
ID_PORT_FILE	Printer output is directed to a file
ID_PORT_UNKNOWN	Connection unknown
ID_ERROR	Other SDK error

## 4.4 ID PrinterStatus

Obtains the current operational status of the printer.

```
int ID_PrinterStatus(HANDLE hSession);
```

### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

Enduro	✓
Ultima	✓
Generation 2	✓

### Return Values

ID_STATUS_READY	Printer is Ready
ID_STATUS_BUSY	Printer is Busy
ID_STATUS_ERROR	Printer is in Error
ID_STATUS_OFFLINE	Printer is Offline
ID_INVALID_SESSION	Invalid session handle
ID_ERROR	Other SDK error

### Remarks

#### Generation 2 Family Only:

If the printer is performing a physical action (e.g. printing a card, magnetic encoding, moving a card) it will return when that action is complete. Thus, it can be used to hold an application like *ID\_WaitForPrinter* does with an Enduro.

## 4.5 ID PrinterInfo

Returns the printer configuration information. The structure provided is loaded with the response(s) the printer returns.

```
int ID_PrinterInfo(HANDLE hSession,
                  VOID *pInfo);
```

Enduro	✓
Ultima	✓
Generation 2	

### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

*pInfo*

Pointer to either a *PrinterInfo* structure, or a *UltimaInfo* structure.

### Structure (Enduro Printers)

```
#define SERIAL_SIZE 20
typedef struct
{
    BOOL        bPrinterConnected;
    DWORD       eModel;
    char        sModel[30];
    DWORD       ePrintheadType;
    char        sPrinterSerial[SERIAL_SIZE];
    char        sPrintheadSerial[SERIAL_SIZE];
    char        sPCBSerial[SERIAL_SIZE];
    WCHAR       sFirmwareVersion[SERIAL_SIZE];

    DWORD       iDummy1;
    DWORD       iDummy2;
    DWORD       iSmartOffset;
    DWORD       iBitFields;
    DWORD       iDensity;

    DWORD       iHandFeed;
    DWORD       iCardsPrinted;
    DWORD       iCardsOnPrinthead;
    DWORD       iDyePanelsPrinted;
    DWORD       iCleansSinceShipped;
    DWORD       iDyePanelsSinceClean;
    DWORD       iCardsSinceClean;
    DWORD       iCardsBetweenCleans;

    DWORD       iPrintHeadPosn;
    DWORD       iImageStartPosn;
    DWORD       iImageEndPosn;
    DWORD       iMajorError;
    DWORD       iMinorError;
    char        sTagUID[20];
    DWORD       iShotsOnFilm;
    DWORD       iShotsUsed;
    char        sDyeFilmType[20];
    DWORD       iColourLength;
    DWORD       iResinLength;
    DWORD       iOvercoatLength;
```



```

    DWORD    eDyeFlags;
    DWORD    iCommandCode;
    DWORD    iDOB;
    DWORD    eDyeFilmManuf;
    DWORD    eDyeFilmProg;
} PRINTER_INFO;

```

### Structure (Ultima Printers)

```

#define SERIAL_SIZE 32
typedef struct
{
    char    firmware_version[ULTIMA_STRING];
    char    printer_serial[ULTIMA_STRING];
    char    model_name[ULTIMA_STRING];
    long    printer_partner_code;
    long    capability_flags;
    char    manufacturer[ULTIMA_STRING];

    long    transfer_target_temperature;
    long    transfer_actual_temperature;

    long    total_cards;
    long    total_panels;
    long    cards_on_printhead;
    long    panels_on_printhead;

    long    cards_low;
    char    colour_name[ULTIMA_STRING];
    long    colour_total_prints;
    long    colour_prints_remaining;
    long    colour_partner_code;
    char    transfer_name[ULTIMA_STRING];
    long    transfer_total_prints;
    long    transfer_prints_remaining;
    long    transfer_partner_code;

    long    error_code;
    long    error_context;

    long    cards_between_prompts;
    long    cleaning_overdue;
    long    cards_since_clean;
    long    panels_since_clean;
    long    cleaning_cycles;

    long    laminate_actual_temperature;
    long    laminate_target_temperature;
    long    laminate_type;
    long    card_speed;

    char    reserved[32];
} ULTIMA_INFO;

```

**Return Values**

ID_SUCCESS	Operation completed successfully.
ID_INVALID_SESSION	Invalid session handle
ID_PARAM_ERROR	Invalid parameter
ID_TIMEOUT	Request timed out
ID_DRIVER_NOT_COMPLIANT	Printer does not support this function (e.g. Rio/Tango)
ID_ERROR	Other SDK error

**Remarks**

- The application **must** provide a pointer to a suitable structure i.e. defined with enough space for the response for the given printer.
- **Generation 2 Family:** Pre-existing applications using this function with the Generation 2 family will not fail, they will get an Enduro *PrinterInfo* structure containing compatible Generation 2 configuration information for all possible fields.

## 4.6 ID\_LastMessage

Enduro	✓
Ultima	✓
Generation 2	✓

Retrieves the last status/error message sent by a printer

```
int ID_LastMessage(HANDLE hSession,
                  LPSTR pBuffer,
                  LPDWORD pBufferSize);
```

### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

*pBuffer*

A pointer to the buffer that is to receive the status message. If NULL, the function fails with *ID\_MORE\_DATA* and *pBufferSize* returns the buffer size required

*pBufferSize*

A pointer to a variable that contains the buffer size.

### Return Values

ID_SUCCESS	Operation completed successfully.
ID_INVALID_SESSION	Invalid session handle
ID_PARAM_ERROR	Invalid parameter
ID_DRIVER_NOT_COMPLIANT	Printer does not support this function (e.g. Rio/Tango)
ID_MORE_DATA	Buffer presented is too small
ID_ERROR	Other SDK error

### Remarks

- *ID\_LastMessage* will return the last status/error message regardless of whether the printer is currently in an error condition.
- Both ASCII and Unicode versions of this function exist (*ID\_LastMessageA* and *ID\_LastMessageW* respectively). The appropriate function is selected for the user dependent on the Unicode setting of the application, thus the user need only access the generic function.
- If *pBuffer* is set to NULL or *pBufferSize* is too small, the function fails (*ID\_MORE\_DATA*) and returns the required buffer size in *pBufferSize*

**N.B.** This is the buffer size required in characters, either Unicode or ASCII, and the buffer to receive the message should be allocated accordingly.

## 4.7 ID Temperature

Interrogates the temperature status of the specified item of equipment.

```
int ID_Temperature(HANDLE hSession,
                  ID_TEMPERATURE TempID,
                  BOOL *Status);
```

Enduro	
Ultima	✓
Generation 2	

### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

*TempID*

Defines the temperature which is being interrogated.

ID_TEMPERATURE_TRANSFER	Reverse Transfer Roller
ID_TEMPERATURE_LAMINATE	Laminator

*Status*

A pointer to a Boolean variable indicating the temperature status:

FALSE	Preheat in Progress
TRUE	Ready

### Return Values

ID_SUCCESS	Operation completed successfully.
ID_INVALID_SESSION	Invalid session handle
ID_INVALID_PRINTER	Invalid for this printer type
ID_PARAM_ERROR	Invalid parameter
ID_ERROR	Other SDK error

### Remarks

- The equipment is deemed to be ready if the actual temperature is within 3° C of the target temperature.
- The target and actual temperatures can be read using *ID\_PrinterInfo*.

## 4.8 ID\_SDKVersion

Returns the version number of the SDK.

```
int ID_SDKVersion(HANDLE hSession,
                 SDKVERSION * pVersion);
```

Enduro	✓
Ultima	✓
Generation 2	✓

### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

*pVersion*

Pointer to an SDKVersion structure

### Structures

```
typedef struct
{
    DWORD Major;
    DWORD Minor;
    DWORD Build;
    DWORD Private;
} SDKVERSION;
```

### Members:

Combined in the form:

**Major.Minor.Build.Private**

the members of the structure give the SDK version number (e.g. Version 3.0.1.2)

### Return Values

ID_SUCCESS	Operation completed successfully.
ID_INVALID_SESSION	Invalid session handle
ID_PARAM_ERROR	Invalid parameter
ID_ERROR	Other SDK error

## 4.9 ID SDKBits

Returns the bit nature of the SDK.

```
byte ID_SDKBits();
```

### Parameters

None

### Return Values

32	32 Bit SDK
64	64 Bit SDK

Enduro	✓
Ultima	✓
Generation 2	✓

## 5 DRIVER SETTINGS FUNCTIONS

### 5.1 ID PrintSettings

<b>Enduro</b>	✓
<b>Ultima</b>	✓
<b>Generation 2</b>	✓

Function to read/write the print settings in the driver.

```
int ID_PrintSettings(HANDLE hSession,
                    ID_READWRITE ReadWrite,
                    PPRINTSET pPrintSettings);
```

**Parameters**

*hSession*

The session handle returned by *ID\_OpenSession*.

*ReadWrite*

The action to be performed:

```
ID_READ
ID_WRITE
```

*pPrintSettings*

Pointer to a structure defining the Print Settings

**Structures**

```
typedef struct
{
    ID_PRINTSIDES Duplex;
    int CopyCount;
    int CardSize;
} PRINTSET;
```

**Members:**

Duplex	Which sides of the card are to be printed
	ID_FRONT_ONLY      Front of Card is printed
	ID_BOTH_SIDES      Both sides of card are printed
	ID_BACK_ONLY      Back of card is printed
CopyCount	The number of cards to be printed
CardSize	The paper ID corresponding to the card size that the printer is set to print.

**Return Values**

ID_SUCCESS	Operation completed successfully.
ID_INVALID_SESSION	Invalid session handle
ID_PARAM_ERROR	Invalid parameter
ID_ERROR	Other SDK error

**Remarks**

When setting the card size in the driver, the application can only use values that that printer is capable of handling, which varies across printer models. Appropriate values (paper IDs) can be obtained from the printer driver using the Windows API function *DeviceCapabilities* to obtain the paper ID values and their corresponding paper names (see the Demo code for an example).

## 5.2 ID CardSettings

Enduro	✓
Ultima	✓
Generation 2	✓

Function to read/write the card settings in the driver

```
int ID_CardSettings(HANDLE hSession,
                   ID_READWRITE ReadWrite,
                   ID_SIDE SideID,
                   PCARDSET pCardParams);
```

### Parameters

#### *hSession*

The session handle returned by *ID\_OpenSession*.

#### *ReadWrite*

The action to be performed:

```
ID_READ
ID_WRITE
```

#### *SideID*

Defines the side which is being written/read.

```
ID_FRONT
ID_BACK
```

#### *pCardParams*

A pointer to a CardSet structure defining the settings for the given side of the card

### Structures

```
typedef struct
{
    ID_COLOURFORMAT ColourFormat;
    BOOL Overcoat;
    ID_ORIENTATION Orientation;
    BOOL Rotation;
} CARDSET;
```

### Members:

**ColourFormat** Indicates the Colour Format for printing

```
ID_COLOURFORMAT_YMC Colour is printed with YMC
ID_COLOURFORMAT_YMCK Colour is printed with YMCK
ID_COLOURFORMAT_K Colour is printed with K Resin Only
```

**Overcoat** A Boolean indicating whether overcoat is to be printed.

**Orientation**

```
ID_ORIENTATION_PORTRAIT Card is Portrait
ID_ORIENTATION_LANDSCAPE Card is Landscape
```

**Rotation** A Boolean indicating whether the image is rotated 180°

### Return Values

```
ID_SUCCESS Operation completed successfully.
ID_INVALID_SESSION Invalid session handle
ID_PARAM_ERROR Invalid parameter
ID_ERROR Other SDK error
```



### 5.3 ID HoloKote

Enduro	✓
Ultima	✓
Generation 2	✓

Function to read/write the HoloKote settings in the driver

```
int ID_HoloKote(HANDLE hSession,
               ID_READWRITE ReadWrite,
               ID_SIDE SideID,
               PHOLOKOTE pHoloKote);
```

#### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

*ReadWrite*

The action to be performed:

```
ID_READ
ID_WRITE
```

*SideID*

Defines the side which is being written/read.

```
ID_FRONT
ID_BACK
```

*pHoloKote*

A pointer to a HoloKote structure defining the HoloKote settings for the given side of the card.

#### Structures

```
typedef struct
{
    int ImageID;
    int Map;
    ID_ROTATION Rotation;
    BOOL DisableCustomKey;
    BOOL UseLaminate;
} HOLOKOTE;
```

#### Members:

- ImageID Identifier of the HoloKote image
- Map Bit mask for the 24 HoloKote positions (B0–23, 1 = print, 0 = do not print) **(Enduro Only)**
- Rotation Indicates the rotation that is applied (if any) to the HoloKote image
  - ID\_ROTATION\_NONE No Rotation
  - ID\_ROTATION\_90 HoloKote is rotated 90° clockwise **(Enduro Only)**
  - ID\_ROTATION\_180 HoloKote is rotated 180°
  - ID\_ROTATION\_270 HoloKote is rotated 270° clockwise **(Enduro Only)**

DisableCustomKey

Boolean indicating whether the custom key is ignored (if fitted)  
(Enduro Only)

UseLaminate

Boolean indicating whether the overcoat setting is for a card that is to be laminated. (Not Ultima)

### Return Values

ID\_SUCCESS

Operation completed successfully.

ID\_INVALID\_SESSION

Invalid session handle

ID\_PARAM\_ERROR

Invalid parameter

ID\_ERROR

Other SDK error

## 5.4 ID\_HoloPatch

Enduro	✓
Ultima	
Generation 2	✓

Function to read/write the HoloPatch settings in the driver

```
int ID_HoloPatch(HANDLE hSession,
                 ID_READWRITE ReadWrite,
                 PHOLOPATCH pHoloPatch);
```

### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

*ReadWrite*

The action to be performed:

```
ID_READ
ID_WRITE
```

*pHoloPatch*

A pointer to a HoloPatch structure defining the HoloPatch settings

### Structures

```
typedef struct
{
    int    Position;
    BOOL   ColourHole;
} HOLOPATCH
```

### Members:

*Position* An integer indicating the HoloPatch position (1 to 24) that is in operation (0 = disabled)

*ColourHole* A Boolean indicating whether a hole is applied to the card image, corresponding to the specified HoloPatch position

### Return Values

ID_SUCCESS	Operation completed successfully.
ID_INVALID_SESSION	Invalid session handle
ID_PARAM_ERROR	Invalid parameter
ID_ERROR	Other SDK error

### Remarks

HoloPatch is applied to the front side of the card only

## 5.5 ID AreaHole

Enduro	✓
Ultima	
Generation 2	✓

Function to read/write the overcoat areas/holes settings in the driver

```
int ID_AreaHole(HANDLE hSession,
                ID_READWRITE ReadWrite,
                ID_SIDE SideID,
                ID_AREAHOLE AreaHoleType,
                int AreaHoleID,
                PAREA pAreaHole);
```

### Parameters

#### *hSession*

The session handle returned by *ID\_OpenSession*.

#### *ReadWrite*

The action to be performed:

```
ID_READ
ID_WRITE
```

#### *SideID*

Defines the side which is being written/read.

```
ID_FRONT
ID_BACK
```

#### *AreaHoleType*

Indicates whether the function is dealing with an Area or a Hole

```
ID_AREA    'Area' is being read/written
ID_HOLE    'Hole' is being read/ written
```

#### *AreaHoleID*

Value specifying the Area/Hole being interrogated/defined (1 to 2)

#### *pAreaHole*

A pointer to an Area structure defining the boundary of the specified Area/Hole

### Structures

```
typedef struct
{
    int    Left;
    int    Width;
    int    Bottom;
    int    Height;
} AREA
```

### Members:

```
Left        Area/Hole Left (X) co-ordinate
Width       Area/Hole Width
Bottom      Area/Hole Bottom (Y) co-ordinate
Height      Area/Hole Height
```

**Return Values**

ID_SUCCESS	Operation completed successfully.
ID_INVALID_SESSION	Invalid session handle
ID_PARAM_ERROR	Invalid parameter
ID_ERROR	Other SDK error

**Remarks**

- If the specified Area/Hole is disabled (or is to be disabled) all fields of the Area structure are zero.
- The drivers for some printers provide predefined overlay holes for magnetic stripes and chips on cards. However, these are only selectable individually.

To give the developer more flexibility and allow them to choose one of these predefined holes together with another hole, only the 'User Defined' selection (as implemented in the driver) is provided by the SDK.

To apply a hole equivalent to one of these predefined holes, the co-ordinate settings to use are given below:

Hole Type	Left	Width	Bottom	Height
Mag Stripe Normal	0	1016	420	170
Mag Stripe Wide	0	1016	400	210
Chip Normal	90	170	295	360
Chip Large	75	205	275	395

## 5.6 ID ResinOptions

Enduro	✓
Ultima	✓
Generation 2	✓

Function to read/write the K resin print settings in the driver

```
int ID_ResinOptions(HANDLE hSession,
                   ID_READWRITE ReadWrite,
                   ID_SIDE SideID,
                   int AreaID,
                   PAREA pArea);
```

### Parameters

#### *hSession*

The session handle returned by *ID\_OpenSession*.

#### *ReadWrite*

The action to be performed:

```
ID_READ
ID_WRITE
```

#### *SideID*

Defines the side which is being written/read.

```
ID_FRONT
ID_BACK
```

#### *AreaID*

Integer specifying the K Resin Area being interrogated/defined (1-10).

#### *pArea*

A pointer to an *Area* structure defining the boundary of the specified resin area (see Section 5.5).

### Structures

```
typedef struct tag_K_OPTIONS
{
    BOOL PicturesUseYMC;
    BOOL BlackText;
    BOOL MonoBitmaps;
    BOOL BlackPolygons;
    BOOL AllBlack;
} K_OPTIONS
```

#### Members:

AllBlack	If set, all black items are printed using resin
PicturesUseYMC	If set, pictures are printed using composite black, not resin (NB Only valid when AllBlack is TRUE)
BlackText	If set, text is printed using resin (NB Only valid when AllBlack is FALSE)
MonoBitmaps	If set, monochrome bitmaps are printed using resin (NB Only valid when AllBlack is FALSE)
BlackPolygons	If set, polygons are printed using resin (NB Only valid when AllBlack is FALSE)

**Return Values**

ID_SUCCESS	Operation completed successfully.
ID_INVALID_SESSION	Invalid session handle
ID_PARAM_ERROR	Invalid parameter
ID_ERROR	Other SDK error

## 5.7 ID ResinArea

Enduro	✓
Ultima	✓
Generation 2	✓

Function to read/write the K resin areas settings in the driver

```
int ID_ResinArea(HANDLE hSession,
                 ID_READWRITE ReadWrite,
                 ID_SIDE SideID,
                 int AreaID,
                 PAREA pArea);
```

### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

*ReadWrite*

The action to be performed:

```
ID_READ
ID_WRITE
```

*SideID*

Defines the side which is being written/read.

```
ID_FRONT
ID_BACK
```

*AreaID*

Integer specifying the K Resin Area being interrogated/defined (1-10).

*pArea*

A pointer to an *Area* structure defining the boundary of the specified resin area (see Section 5.5)

### Structures

See structure definition in section 5.5.

### Return Values

ID_SUCCESS	Operation completed successfully.
ID_INVALID_SESSION	Invalid session handle
ID_PARAM_ERROR	Invalid parameter
ID_ERROR	Other SDK error

### Remarks

If the specified K resin area is disabled, or is to be disabled, all members of the *Area* structure should be/are zero.



## 5.8 ID PrintableArea

Function to read/write the printable area settings in the driver

```
int ID_PrintableArea(HANDLE hSession,
                    ID_READWRITE ReadWrite,
                    PAREA pArea);
```

Enduro	
Ultima	
Generation 2	✓

### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

*ReadWrite*

The action to be performed:

ID\_READ  
ID\_WRITE

*pArea*

A pointer to an *Area* structure defining the boundary of the printable area (see Section 5.5)

### Structures

See structure definition in section 5.5.

### Return Values

ID_SUCCESS	Operation completed successfully.
ID_INVALID_SESSION	Invalid session handle
ID_PARAM_ERROR	Invalid parameter
ID_ERROR	Other SDK error

### Remarks

- If the printable area is disabled, or is to be disabled, all members of the *Area* structure should be/are zero.
- This setting is intended for use with half panel films to define the location on the card where that half panel is to be printed.



### 5.10 ID ColourArea

Enduro	✓
Ultima	✓
Generation 2	✓

Function to read/write the colour area settings in the driver. Each area has its own colour correction setting that takes precedence over the overall driver setting (as defined by *ID\_ColourCorrection*) for all elements within the bounds of that area.

```
int ID_ColourArea(HANDLE hSession,
                  ID_READWRITE ReadWrite,
                  ID_SIDE SideID,
                  int AreaID,
                  PAREA pArea,
                  PCOLCORRECT pCorrect);
```

#### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

*ReadWrite*

The action to be performed:

- ID\_READ
- ID\_WRITE

*SideID*

Defines the side which is being written/read.

- ID\_FRONT
- ID\_BACK

*AreaID*

Integer specifying the Colour Area being interrogated/defined (1 to 5).

*pArea*

A pointer to an Area structure defining the boundary of the specified area (see Section 5.5)

*pCorrect*

Pointer to an integer which receives/defines the colour correction (see 5.6 *ID\_ColourCorrection*)

#### Structures

See structure definition in section 5.5.

#### Return Values

ID_SUCCESS	Operation completed successfully.
ID_INVALID_SESSION	Invalid session handle
ID_PARAM_ERROR	Invalid parameter
ID_ERROR	Other SDK error

#### Remarks

If the specified Colour Area is disabled, or is to be disabled, all members of the Area structure are zero

## 5.11 ID\_ColourAdjust

Function to read/write the advanced colour adjustment settings in the driver.

```
int ID_ColourAdjust(HANDLE hSession,
                   ID_READWRITE ReadWrite,
                   PCOLORADJUSTMENT pColourAdj);
```

Enduro	✓
Ultima	✓
Generation 2	✓

### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

*ReadWrite*

The action to be performed:

```
ID_READ
ID_WRITE
```

*pColourAdj*

A pointer to a Windows COLORADJUSTMENT structure, which is used in the Advanced colour settings of the driver

### Return Values

ID_SUCCESS	Operation completed successfully.
ID_INVALID_SESSION	Invalid session handle
ID_PARAM_ERROR	Invalid parameter
ID_ERROR	Other SDK error

## 5.12 ID Sharpness

Function to read/write the image sharpness setting in the driver

```
int ID_Sharpness(HANDLE hSession,
                ID_READWRITE ReadWrite,
                int * pSharpness);
```

Enduro	✓
Ultima	✓
Generation 2	✓

### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

*ReadWrite*

The action to be performed:

```
ID_READ
ID_WRITE
```

*pSharpness*

Pointer to an integer which receives/defines the sharpness (Range: -2 to +2)

### Return Values

ID_SUCCESS	Operation completed successfully.
ID_INVALID_SESSION	Invalid session handle
ID_PARAM_ERROR	Invalid parameter
ID_ERROR	Other SDK error

### 5.13 ID PrintSpeed

Function to read/write the print speed setting in the driver

```
int ID_PrintSpeed(HANDLE      hSession,
                  ID_READWRITE ReadWrite,
                  PPRINTSPEED pPrintSpeed);
```

Enduro	✓
Ultima	
Generation 2	

#### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

*ReadWrite*

The action to be performed:

```
ID_READ
ID_WRITE
```

*pPrintSpeed*

Pointer to an integer which receives/defines the print speed

```
ID_HIGHSPEED      Higher Speed (Draft Quality)
ID_NORMALSPEED    Normal Speed (Normal Quality)
```

#### Return Values

ID_SUCCESS	Operation completed successfully.
ID_INVALID_SESSION	Invalid session handle
ID_PARAM_ERROR	Invalid parameter
ID_ERROR	Other SDK error

### 5.14 ID PowerLevel

Enduro	✓
Ultima	✓
Generation 2	✓

Function to read/write the power levels settings in the driver

```
int ID_PowerLevel(HANDLE hSession,
                  ID_READWRITE ReadWrite,
                  PPOWERLEVEL pPowerLevel);
```

#### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

*ReadWrite*

The action to be performed:

```
ID_READ
ID_WRITE
```

*pPowerLevel*

Pointer to a structure which receives/defines the power level

#### Structures

```
typedef struct
{
    int YMC;
    int Resin;
    int Overcoat;
} POWERLEVEL
```

#### Members:

- YMC                      Power level for YMC
- Resin                    Power level for K Resin
- Overcoat                Power level for Overcoat

#### Return Values

- ID\_SUCCESS                      Operation completed successfully.
- ID\_INVALID\_SESSION              Invalid session handle
- ID\_PARAM\_ERROR                  Invalid parameter
- ID\_ERROR                         Other SDK error

#### Remarks

The range of the power level values is 0 – 100 for all printers.

### 5.15 ID Rewritable

Enduro	✓
Ultima	
Generation 2	✓

Function to read/write the rewritable card settings in the driver

```
int ID_Rewritable(HANDLE hSession,
                 ID_READWRITE ReadWrite,
                 PREWRITABLE pRewritable);
```

#### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

*ReadWrite*

The action to be performed:

```
ID_READ
ID_WRITE
```

*pRewritable*

Pointer to a structure which receives/defines the settings for rewritable cards

#### Structures

```
typedef struct
{
    BOOL EraseBeforePrint;
    AREA EraseArea;
    int ErasePowerStart;
    int ErasePowerEnd;
    int WritePower;
} REWRITABLE
```

#### Members:

- EraseBeforePrint* Boolean - whether the card is to be erased before printing takes place
- EraseArea* Erase Area (see Section 5.5).
- ErasePowerStart* Integer - the power for the start of the ramp of the erase pass over the card
- ErasePowerEnd* Integer - the power for the end of the ramp of the erase pass over the card (**Enduro Only**).
- WritePower* Integer - the power for writing the card (**Enduro Only**).

#### Return Values

- ID\_SUCCESS Operation completed successfully.
- ID\_INVALID\_SESSION Invalid session handle
- ID\_PARAM\_ERROR Invalid parameter
- ID\_ERROR Other SDK error



**Remarks**

- The area defined here corresponds to Front Erase Area 1. To define other areas, use the SDK function *ID\_RewritableArea* (Section 5.16).
- The Enduro Family uses a 'power ramp' to erase cards, with power increasing from a start value to an end value over the length of the card as it is erased. It also has a write power setting. The Generation 2 family uses a constant value to erase and this is defined in the *ErasePowerStart* field, the other fields being set to 0 (or ignored) for read and write respectively.



### 5.17 ID Control

Reads/writes specific flags

```
int ID_Control(HANDLE hSession,
               ID_READWRITE ReadWrite,
               PCONTROL Flags);
```

Enduro	✓
Ultima	✓
Generation 2	✓

#### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

*ReadWrite*

The action to be performed:

- ID\_READ
- ID\_WRITE

*Flags*

Pointer to a bit field. Bit assignments are:

- B0            Encode Only
- B1            ISO 7810 Compliance (Ultima Only)
- Other Bits    Reserved

#### Return Values

- ID\_SUCCESS                    Operation completed successfully.
- ID\_INVALID\_SESSION           Invalid session handle
- ID\_PARAM\_ERROR               Invalid parameter
- ID\_ERROR                        Other SDK error

## 5.18 ID\_GUIControl

Controls the driver GUI

```
int ID_GUIControl(HANDLE hSession,
                 ID_READWRITE ReadWrite,
                 PID_GUICONTROL pGUIControl);
```

Enduro	✓
Ultima	✓
Generation 2	✓

### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

*ReadWrite*

The action to be performed:

```
ID_READ
ID_WRITE
```

*pGUIControl*

Pointer to a structure which defines the GUI control settings

### Structures

```
typedef struct
{
    BOOL    User;
    BOOL    Printer;
} ID_GUICONTROL;
```

### Members:

*User*                Disables the current user from access to the GUI for all Magicard printers

*Printer*            Disables the GUI for the current Magicard printer for all users

### Return Values

```
ID_SUCCESS                    Operation completed successfully.
ID_INVALID_SESSION            Invalid session handle
ID_PARAM_ERROR                Invalid parameter
ID_ERROR                        Other SDK error
```

### Remarks

Since it affects all users of the PC, the 'Printer' GUI control is governed by Windows access control. Thus, it is only effective if used by an application that has Administrator access rights. Without these rights, the operation fails without the operating system generating an error, and thus the SDK also cannot (and so does not) generate an error.

## 5.19 ID Resolution

Defines the resolution for printing in the current SDK session and sets the driver accordingly.

```
int ID_Resolution(HANDLE hSession,
                  ID_READWRITE ReadWrite,
                  LPBYTE Resolution);
```

Enduro	
Ultima	
Generation 2	✓

### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

*ReadWrite*

The action to be performed:

```
ID_READ
ID_WRITE
```

*Resolution*

Resolution setting:      0 - 300 x 300 dpi  
                                  1 - 600 x 300 dpi mono

### Return Values

ID_SUCCESS	Operation completed successfully.
ID_INVALID_SESSION	Invalid session handle
ID_INVALID_PRINTER	Invalid for this printer
ID_PARAM_ERROR	Invalid parameter
ID_ERROR	Other SDK error

### Remarks

- Resolution is in the form *card width x card height* i.e. 600 x 300 is 300 dpi for the height of the card (short side) and 600 dpi for the width (long side).
- This function **MUST** be used if 600 dpi printing is required, otherwise 300 dpi (the default setting) will be used for all subsequent printing in the session.

## 6 PRINTER SETTINGS FUNCTIONS

### 6.1 ID CardWidth

Returns the maximum card width (in pixels) for the printer

```
int ID_CardWidth(HANDLE hSession);
```

Enduro	✓
Ultima	✓
Generation 2	✓

#### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

#### Return Values

N	Card width in pixels for this model of printer
ID_ERROR	No session open
ID_INVALID_SESSION	Invalid session handle

#### Remarks

- Ultima - 1036
- Generation 2 family - 1013
- All other printers - 1016

## 6.2 ID CardHeight

Returns the card height (in pixels) for the printer

```
int ID_CardHeight(HANDLE hSession);
```

Enduro	✓
Ultima	✓
Generation 2	✓

### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

### Return Values

N	Card height in pixels for this model of printer
ID_ERROR	No session open
ID_INVALID_SESSION	Invalid session handle

### Remarks

- Ultima - 664
- All other printers - 642

### 6.3 ID HandFeed

Function to read/write the hand feed setting in the printer

```
int ID_HandFeed(HANDLE          hSession,
                ID_READWRITE    ReadWrite,
                PHANDFEED       pHandFeed);
```

Enduro	✓
Ultima	
Generation 2	✓

#### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

*ReadWrite*

The action to be performed:

```
ID_READ
ID_WRITE
```

*pHandFeed*

Pointer to an integer which receives/defines the Hand Feed setting

```
ID_HANDFEED_OFF
ID_HANDFEED_ON
```

#### Return Values

ID_SUCCESS	Operation completed successfully.
ID_INVALID_SESSION	Invalid session handle
ID_PARAM_ERROR	Invalid parameter
ID_ERROR	Other SDK error



## 6.4 ID EjectMode

Enduro	✓
Ultima	✓
Generation 2	✓

Function to read/write the eject mode of the printer

```
int ID_EjectMode(HANDLE hSession,
                 ID_READWRITE ReadWrite,
                 int * iMode);
```

### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

*ReadWrite*

The action to be performed:

```
ID_READ
ID_WRITE
```

*iMode*

Pointer to an integer defining the Eject Mode setting

```
ID_EJECT_ON      Cards are ejected when action is complete
ID_EJECT_OFF     Cards are not ejected when action is complete
```

### Return Values

```
ID_SUCCESS      Operation completed successfully.
ID_INVALID_SESSION Invalid session handle
ID_PARAM_ERROR  Invalid parameter
ID_ERROR        Other SDK error
```

### Remarks

- **Enduro Printers** - The parameter is stored in RAM memory, so is volatile i.e. when power is removed it resets to its default setting. (Default = ON - cards are automatically ejected after print jobs).
- **Ultima & Generation 2 Family** – Eject Mode is only valid for the duration of the SDK session i.e. at the start of the session, the mode initialises to 'ON'.

## 6.5 ID HorzEject

Enduro	✓
Ultima	
Generation 2	

Function to read/write the horizontal eject mode of the printer

```
int ID_HorzEject(HANDLE hSession,
                 ID_READWRITE ReadWrite,
                 PID_HORZEJECT iMode);
```

### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

*ReadWrite*

The action to be performed:

```
ID_READ
ID_WRITE
```

*iMode*

Pointer to an integer defining the Horizontal Eject Mode setting

```
ID_HORZEJECT_OFF
ID_HORZEJECT_ON
```

### Return Values

ID_SUCCESS	Operation completed successfully.
ID_INVALID_SESSION	Invalid session handle
ID_PARAM_ERROR	Invalid parameter
ID_ERROR	Other SDK error

### Remarks

Horizontal eject mode is for use with Kiosk printers. Using this mode with a standard printer could result in card jams and possible damage to the rotation mechanism (the front bezel must have been removed first).

## 6.6 ID SmartMode

Enduro	✓
Ultima	
Generation 2	

Function to read/write the smart card encoding position of the printer

```
int ID_SmartMode(HANDLE hSession,
                ID_READWRITE ReadWrite,
                int * iMode);
```

### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

*ReadWrite*

The action to be performed:

```
ID_READ
ID_WRITE
```

*iMode*

Pointer to an integer defining the smart encoding position

```
ID_SMART_DEFAULT    Default position is to be used (i.e. in the encoder)
ID_SMART_PLATEN     Encoding position is on the printer platen
ID_SMART_XLI        Encoding position for extended cards
                    (Extended printers only)
```

### Return Values

```
ID_SUCCESS          Operation completed successfully.
ID_INVALID_SESSION  Invalid session handle
ID_PARAM_ERROR      Invalid parameter
ID_ERROR            Other SDK error
```

### Remarks

- This setting is non-volatile and is retained by the printer through a power off/reset cycle.
- 'Default' is for contact encoding; 'Platen' is for contactless.

## 6.7 ID SmartOffset

Function to read/write the smart card encoding position

```
int ID_SmartOffset(HANDLE hSession,
                  ID_READWRITE ReadWrite,
                  int * iParam);
```

Enduro	✓
Ultima	
Generation 2	✓

### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

*ReadWrite*

The action to be performed:

```
ID_READ
ID_WRITE
```

*iParam*

Pointer to an integer defining the smart encoding position. This is between 0 and 99 (if no parameter is supplied, defaults to 0).

### Return Values

ID_SUCCESS	Operation completed successfully.
ID_INVALID_SESSION	Invalid session handle
ID_PARAM_ERROR	Invalid parameter
ID_ERROR	Other SDK error

### Remarks

- This setting is non-volatile and is retained by the printer through a power on/reset.
- When the parameter is 0, the card's leading edge will be placed adjacent to the front card sensor. Each increment offsets the card position by approximately 1mm towards the rear of the printer.
- The Smart Offset parameter is passed to a positioning algorithm within the printer. When reading it, the value obtained will often be one less than originally written due to rounding that occurs in the algorithm within the printer firmware when being written.

## 6.8 ID EraseSpeed

Function to read/write the erase speed (used in the erase cycle on a rewritable card) in the printer

```
int ID_EraseSpeed(HANDLE hSession,
                  ID_READWRITE ReadWrite,
                  int * iMode)
```

Enduro	✓
Ultima	
Generation 2	

### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

*ReadWrite*

The action to be performed:

```
ID_READ
ID_WRITE
```

*iMode*

A pointer to an integer defining the erase speed:

```
ID_ERASE_THOROUGH  Slower speed erasing, more thorough
ID_ERASE_QUICK      Higher speed erasing, less thorough
```

### Return Values

```
ID_SUCCESS          Operation completed successfully.
ID_INVALID_SESSION  Invalid session handle
ID_PARAM_ERROR      Invalid parameter
ID_ERROR            Other SDK error
```

## 6.9 ID Password

Function to set or provide the password to the printer for use in controlled access environments.

```
int ID_Password(HANDLE hSession,
               ID_PWDCMD Command,
               string Password1,
               string Password2);
```

Enduro	✓
Ultima	
Generation 2	

### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

*Command*

The action to be performed:

- ID\_PWDSET Sets the password in the printer
- ID\_PWDUSE Provides the password to the printer to verify access

*Password1*

*Password2*

The string(s) to be used in the password interchange with the printer

### Return Values

ID_SUCCESS	Operation completed successfully.
ID_INVALID_SESSION	Invalid session handle
ID_PARAM_ERROR	Invalid parameter
ID_INVALID_PRINTER	Invalid for this printer type
ID_ERROR	Other SDK error

### Remarks

<u>Action</u>	<u>Command</u>	<u>Password 1</u>	<u>Password 2</u>
Set a new password	ID_PWDSET	NULL	New Password
Reset to default password	ID_PWDSET	Current Password	NULL
Change password	ID_PWDSET	Current Password	New Password
Supply password to printer to enable the subsequent action	ID_PWDUSE	Current Password	NULL

## 6.10 ID\_IPSettings

Enduro	✓
Ultima	✓
Generation 2	✓

Function to read/write the ethernet settings of the printer

```
int ID_IPSettings(HANDLE hSession,
                  ID_READWRITE ReadWrite,
                  PIPDATA pIPData);
```

### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

*ReadWrite*

The action to be performed:

```
ID_READ
ID_WRITE
```

*pIPData*

A pointer to a structure defining the IP settings of the printer

### Structures

```
typedef struct
{
    ID_ADDRESSING IPAddressMode;
    int IPAddress;
    int SubnetMask;
    int Gateway;
} IPDATA
```

### Members:

IPAddressMode

```
ID_STATIC    Static IP addressing
ID_DYNAMIC   Dynamic IP addressing
```

IPAddress     The IP address of the printer (defined as an integer)

SubnetMask    The subnet mask of the printer (defined as an integer)

Gateway       The IP address of the network gateway (defined as an integer)

### Return Values

```
ID_SUCCESS            Operation completed successfully.
ID_INVALID_SESSION    Invalid session handle
ID_PARAM_ERROR        Invalid parameter
ID_ERROR              Other SDK error
```

## 6.11 ID\_CardLocation

Function to determine the location of a card in the printer (if any).

```
int ID_CardLocation(HANDLE hSession,
                   PID_LOCATION location);
```

Enduro	
Ultima	
Generation 2	✓

### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

*location*

Pointer to a variable to receive the location in the printer.

ID_NO_CARD	No card is in the printer
ID_PRINT_READY	Card is in the print standby position
ID_ENCODER	Card is in the encoder
ID_UNKNOWN	Card location is not recognised

### Return Values

ID_SUCCESS	Operation completed successfully.
ID_INVALID_SESSION	Invalid session handle
ID_PARAM_ERROR	Invalid Parameter
ID_ERROR	Other SDK Error

### Remarks

Since it responds only when card moving is not in progress, calling *ID\_CardLocation* can be used to detect that printing or encoding is complete, as only then will it return.



## 6.12 ID\_HoloKoteIdentity

Reads the identity string(s) for the HoloKote images

```
int ID_HoloKoteIdentity(HANDLE hSession,
                       BYTE ImageNo,
                       LPSTR pBuffer,
                       LPDWORD pSize);
```

Enduro	
Ultima	✓
Generation 2	

### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

*ImageNo*

Number of the HoloKote Image (1 to 10, 0 = All Slots).

*pBuffer*

Pointer to a buffer to receive the Identity String.

*pSize*

Pointer to a variable which is the size of the buffer to receive the Identity String (in characters)

### Return Values

ID_SUCCESS	Operation completed successfully.
ID_INVALID_SESSION	Invalid session handle
ID_PARAM_ERROR	Invalid parameter
ID_ERROR	Other SDK error
ID_MORE_DATA	Buffer presented is too small

### Remarks

- Both ASCII and Unicode versions of this function exist (*ID\_HoloKoteIdentityA* and *ID\_HoloKoteIdentityW* respectively). The appropriate function is selected for the user dependent on the Unicode setting of the application, thus the user need only access the generic function.
- If *pBuffer* is set to NULL or *pSize* is too small, the function fails with ID\_MORE\_DATA and returns the required buffer size in *pSize*
- If an *ImageNo* of 0 is requested, a comma separated string is returned that contains the identities of all 10 slots (in ascending order). **N.B.** If a slot is empty, it is indicated by successive commas.

### 6.13 ID\_HoloKoteCount

Reads the HoloKote count for the printer

```
int ID_HoloKoteCount(HANDLE hSession);
```

Enduro	
Ultima	✓
Generation 2	✓

#### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

#### Return Values

N

Number of HoloKotes for this model of printer

ID\_INVALID\_PRINTER

Invalid function for this model of printer

ID\_INVALID\_SESSION

Invalid session handle

#### Remarks

For example:

Ultima – 10 HoloKotes

Rio Pro 360 – 10 HoloKotes

300 Model – 3 or 4 HoloKotes

600, D, K Models – 10 HoloKotes

## 6.14 ID\_HoloKotePreview

Enduro	
Ultima	✓
Generation 2	✓

Obtains preview images of the HoloKotes that are set in the printer.

```
int ID_HoloKotePreview(HANDLE hSession,
                      HKPREVIEW pBuffer[N],
                      LPDWORD pSize);
```

```
int ID_HoloKotePreview(HANDLE hSession,
                      Bitmap[] Npreviews); (.Net shim only)
```

```
int ID_HoloKotePreview(HANDLE hSession,
                      BufferedImage[] Jpreviews); (Java shim only)
```

### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

*pBuffer*

Pointer to a buffer which receives the preview(s)

*pSize*

Pointer to a variable which is the size of the buffer to receive the preview(s)

[*Npreviews* - An array of .Net Bitmaps (.Net shim only)]

[*Jpreviews* - An array of Java BufferedImage (Java shim only)]

### Structures

```
typedef struct
{
    char Header[256];
    char Preview[259 * 166];
} HKPREVIEW;
```

#### Members:

Header	A free format data area. Standard holokotes do not contain any relevant information, but custom holokotes can contain user defined information (e.g. identifying text).
Preview	A 259 x 166 bit buffer, image origin is the top left corner. The image is byte packed with no end of line padding. Padding is to the byte boundary at the end of the HoloKote image.

### Return Values

ID_SUCCESS	Operation completed successfully.
ID_INVALID_SESSION	Invalid session handle
ID_INVALID_PRINTER	Invalid action for this printer type
ID_PARAM_ERROR	Invalid parameter
ID_MORE_DATA	Buffer presented is too small
ID_ERROR	Other SDK error

**Remarks**

- The number of previews returned (N) depends on the printer and can be determined using *ID\_HoloKoteCount*.
- If *pBuffer* is set to NULL or *pSize* is too small, the function fails with ID\_MORE\_DATA and returns the required buffer size in *pSize*

	Byte 0								Byte 1								Byte 2			
	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15	B16	B17	...	...
Line 0	B7	B6	B5	B4	B3	B2	B1	B0	B7	B6	B5	B4	B3	B2	B1	B0	B7	B6	...	...

**Figure 3 – Image Bit Assignment**

### 6.15 ID Sensors

Enduro	
Ultima	
Generation 2	✓

Reads the current sensor states

```
int ID_Sensors(HANDLE hSession,
               LPDWORD pStateize);
```

#### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

*pState*

Pointer to a variable that receives the sensor states, which is a bitmask

<u>Bit</u>	<u>Function</u>
0	Lid Sensor
1	Hopper Sensor (0 = Cards Low, 1 = Cards in Hopper)
2	Hopper Sensor Valid
3	Exit Sensor (0 = Card Absent 1 = Card Present)
4	Exit Sensor Valid
Bits 5-31 are unused	

#### Return Values

ID_SUCCESS	Operation completed successfully.
ID_INVALID_SESSION	Invalid session handle
ID_PARAM_ERROR	Invalid parameter
ID_ERROR	Other SDK error
ID_MORE_DATA	Buffer presented is too small

#### Remarks

The 'Valid' bits are set if the sensor is fitted, in which case the corresponding sensor state bit is valid; otherwise the state bit should be ignored. N.B. The lid sensor is always fitted.

## 7 GENERATION 2 PRINTER SETTINGS FUNCTIONS

### 7.1 Overview

The Generation 2 printers use a different communications protocol, so it is possible to access many of the printer settings by generic type i.e. accessing the parameter as an integer, a boolean, a buffer or a string, as determined by the SDK function used. The chosen function requires an identifier and either a pointer (if reading), or the new value (if writing).

The table below lists the parameters that can be accessed, and the appropriate type.

**N.B.** Any attempt to access a parameter using an incorrect type, or to write to a read-only parameter, generates an error.

Parameter	Parameter ID	String/ Buffer	Integer	Boolean	Read Only
Double Sided	OP_CAPABILITY_DUPLEX	✓		✓	✓
Magnetic Encoding	OP_CAPABILITY_MAG	✓		✓	✓
Smart Encoding	OP_CAPABILITY_SMART	✓		✓	✓
System Version	OP_VERSION_SYSTEM	✓			✓
Printer App Version	OP_VERSION_PRINTER_APP	✓			✓
Firmware Version	OP_VERSION_FIRMWARE	✓			✓
Mag Encoding F/W Version	OP_VERSION_MAG_FIRMWARE	✓			✓
Printer Serial No.	OP_SERIAL_PRINTER	✓			✓
Printhead Serial No.	OP_SERIAL_PRINthead	✓			✓
PCB Serial No.	OP_SERIAL_BOARD	✓			✓
Model Name	OP_MODEL_NAME	✓			✓
Dealer Code	OP_PRINTER_DEALER_CODE	✓	✓		✓
Project Code	OP_PRINTER_PROJECT_CODE	✓	✓		✓
Total Cards Printed	OP_STATS_TOTAL_CARDS	✓	✓		✓
Total Panels Printed	OP_STATS_TOTAL_PANELS	✓	✓		✓
Cards Printed on Printhead	OP_STATS_PRINthead_CARDS	✓	✓		✓
Panels Printed on Printhead	OP_STATS_PRINthead_PANELS	✓	✓		✓
Cards since Clean	OP_STATS_CARDS_SINCE_CLEAN	✓	✓		✓
Panels since Clean	OP_STATS_PANELS_SINCE_CLEAN	✓	✓		✓
Cleaning Cycles	OP_STATS_CLEANS	✓	✓		✓
Dye Film Name	OP_DYE_NAME	✓			✓
Dye Film Serial No.	OP_DYE_SERIAL	✓			✓
Dye Film Total Prints	OP_DYE_PRINTS_TOTAL	✓	✓		✓
Dye Film Prints Remaining	OP_DYE_PRINTS_REMAINING	✓	✓		✓
Dye Film manufacturer	OP_DYE_MANUFACTURER	✓			✓
Cleans between Prompts	OP_CLEAN_CARDS_INTERVAL	✓	✓		✓
Clean Overdue	OP_CLEAN_OVERDUE	✓		✓	✓
MAC Address	OP_NET_MAC_ADDRESS	✓			✓
Use DHCP	OP_NET_USE_DHCP	✓		✓	
Dynamic IP Address	OP_NET_IP_ADDRESS	✓			
Dynamic Gateway	OP_NET_GATEWAY	✓			
Dynamic Subnet Mask	OP_NET_SUBNET	✓			
Static IP Address	OP_NET_STATIC_IP	✓			
Static Gateway	OP_NET_STATIC_GATEWAY	✓			
Static Subnet Mask	OP_NET_STATIC_SUBNET	✓			
Error Code	OP_ERROR	✓	✓		✓
Hand Feed	OP_HAND_FEED	✓		✓	
Card Position	OP_CARD_LOCATION	✓	✓		✓
Encoding Start	OP_MAG_START	✓	✓		

Continues...

Parameter	Parameter ID	String/ Buffer	Integer	Boolean	Read Only
...continued					
Colour Panel Length	OP_YMC_LENGTH	✓	✓		✓
K Resin Panel Length	OP_K_LENGTH	✓	✓		✓
Overcoat Panel Length	OP_O_LENGTH	✓	✓		✓
Smart Encoding Offset	OP_SMART_OFFSET	✓	✓		
HoloKote Image Slots	OP_HOLOKOTE_SLOTS	✓	✓		✓
600 dpi available	OP_CAPABILITY_600DPI	✓		✓	✓
Manufacturer Name	OP_MANUFACTURER	✓		✓	✓
Card Hopper Sensor	OP_HOPPER_SENSOR	✓		✓	✓
Printer Exit Sensor	OP_EXIT_SENSOR	✓		✓	✓
Printer Lid Sensor	OP_LID_SENSOR	✓		✓	✓
Card Hopper Sensor Valid	OP_HOPPER_SENSOR_FITTED	✓		✓	✓
Printer Exit Sensor Valid	OP_EXIT_SENSOR_FITTED	✓		✓	✓

**Table 1 – Generation 2 Parameter Identifiers**

## 7.2 ID AccessInt

Function to read an integer parameter from the printer.

```
int ID_AccessInt(HANDLE      hSession,
                ID_READWRITE ReadWrite,
                G2_PARAM    param_id,
                int *        ptr);
```

Enduro	
Ultima	
Generation 2	✓

### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

*ReadWrite*

The action to be performed:

```
ID_READ
ID_WRITE
```

*param\_id*

Identifier of the parameter to be accessed (see Table 1)

*ptr*

Pointer to an integer which receives the parameter

### Return Values

ID_SUCCESS	Operation completed successfully.
ID_INVALID_SESSION	Invalid session handle
ID_INVALID_PRINTER	Invalid for this printer type
ID_PARAM_ERROR	Invalid parameter
ID_READ_ERROR	Cannot read this parameter as this type
ID_ERROR	Other SDK error



### 7.3 ID AccessBool

Function to read a boolean parameter from the printer.

```
int ID_AccessBool(HANDLE      hSession,
                  ID_READWRITE ReadWrite,
                  G2_PARAM    param_id,
                  bool *       ptr);
```

Enduro	
Ultima	
Generation 2	✓

#### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

*ReadWrite*

The action to be performed:

```
ID_READ
ID_WRITE
```

*param\_id*

Identifier of the parameter to be accessed (see Table 1)

*ptr*

Pointer to a boolean which receives the parameter

#### Return Values

ID_SUCCESS	Operation completed successfully.
ID_INVALID_SESSION	Invalid session handle
ID_PARAM_ERROR	Invalid parameter
ID_READ_ERROR	Cannot read this parameter as this type
ID_ERROR	Other SDK error

## 7.4 ID AccessBuffer

Enduro	
Ultima	
Generation 2	✓

Function to read a null terminated string parameter from the printer.

```
int ID_AccessBuffer(HANDLE hSession,
                   ID_READWRITE ReadWrite,
                   G2_PARAM param_id,
                   char * ptr);
int * size);
```

### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

*ReadWrite*

The action to be performed:

- ID\_READ
- ID\_WRITE

*param\_id*

Identifier of the parameter to be accessed (see Table 1)

*ptr*

Pointer to a buffer which receives the parameter

*size*

Pointer to an integer which receives the size of the buffer

### Return Values

ID_SUCCESS	Operation completed successfully.
ID_INVALID_SESSION	Invalid session handle
ID_PARAM_ERROR	Invalid parameter
ID_MORE_DATA	Size specified is too small to receive the parameter
ID_ERROR	Other SDK error

### Remarks

This function is not supported by the .Net shim. If programming using C# or Visual Basic, use *ID\_AccessString* instead.

## 7.5 ID AccessString

Function to read a string parameter from the printer.

```
int ID_AccessString(HANDLE hSession,
                   ID_READWRITE ReadWrite,
                   G2_PARAM param_id,
                   CString & param);
```

Enduro	
Ultima	
Generation 2	✓

### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

*ReadWrite*

The action to be performed:

ID\_READ  
ID\_WRITE

*param\_id*

Identifier of the parameter to be accessed (see Table 1)

*param*

Reference to the CString that receives the parameter

### Return Values

ID_SUCCESS	Operation completed successfully.
ID_INVALID_SESSION	Invalid session handle
ID_PARAM_ERROR	Invalid parameter
ID_ERROR	Other SDK error

### Remarks

With CString being an MFC class, this function can be used with Visual Studio compilations only.

## 8 MAGNETIC ENCODING FUNCTIONS

### 8.1 Overview

Magnetic encoding can be implemented in one of several ways:-

1. If printing via GDI, either ...

- a) include the encoding data with the GDI image data as text strings in the form: ~TRACK NUMBER,DATA (e.g. "~1,MAGDATA").

If not smart encoding, this option has the advantage that the application does not need to manage the serial process of encoding and then printing. Therefore, multiple card images and their mag data can be created as a single multi-page print job and sent through the Windows driver, and the application can continue with its next tasks.

- b) Use the *ID\_EncodeMag* function (as described in Section 8.2).

This option enables the application to control the individual serial processes of encoding and then printing, which may important especially when also smart encoding.

2. If printing via the SDK's printing functions, either...

- a) Use the *ID\_DrawMagText* function (as described in Section 9.8).

If not smart encoding, this option has the advantage that the application does not need to manage the serial process of encoding and then printing. Therefore, multiple card images and their mag data can be created and sent through the Windows driver, and the application can continue with its next tasks.

- b) Use the *ID\_EncodeMag* function (as described in Section 8.2).

This option enables the application to control the individual serial processes of encoding and then printing, which may important especially when also smart encoding.

## 8.2 ID EncodeMag

Writes data to the magnetic stripe on the card

```
int ID_EncodeMag(HANDLE hSession,
                 PID_MAGDEF pData);
```

Enduro	✓
Ultima	✓
Generation 2	✓

### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

*pData*

Pointer to a data structure defining the data to be encoded

### Structures

```
typedef struct
{
    int iCharCount,
    char */wchar *pData,
    int iBitsPerChar,
    int iBitsPerInch,
    int iParity,
    int iLRC
} ID_MAGTRACK

typedef struct
{
    int iEncodingSpec,
    int iCoercivity,
    int iVerify,
    MAGTRACK Track[3]
} ID_MAGDEF;
```

### Members - MagDef

*iEncodingSpec* The encoding method to be used

```
ID_ENCODING_ISO
ID_ENCODING_JIS2
```

*iVerify* Specifies whether verification is required

```
ID_VERIFY_OFF
ID_VERIFY_ON
```

*iCoercivity* The coercivity of the encoding

```
ID_COERCIVITY_DEFAULT Default Coercivity
ID_COERCIVITY_HICO High Coercivity
ID_COERCIVITY_LOCO Low Coercivity
ID_COERCIVITY_MIDCO Mid Coercivity
```

**Members - MagTrack**

<code>iCharCount</code>	The number of characters to be written to the track (including start and end sentinels)
<code>lpData</code>	Pointer to a buffer containing the data to be written
<code>iBitsPerChar</code>	The number of bits per character for the encoding (ISO only) <code>ID_BITSPERCHAR_DEFAULT</code> <code>ID_BITSPERCHAR_1</code> <code>ID_BITSPERCHAR_5</code> <code>ID_BITSPERCHAR_7</code>
<code>iBitsPerInch</code>	The number of bits per inch for the encoding (ISO only) <code>ID_BITSPERINCH_DEFAULT</code> <code>ID_BITSPERINCH_75</code> <code>ID_BITSPERINCH_210</code>
<code>iParity</code>	The parity for the encoding (ISO only) <code>ID_PARITY_DEFAULT</code> <code>ID_PARITY_OFF</code> <code>ID_PARITY_ODD</code> <code>ID_PARITY_EVEN</code>
<code>iLRC</code>	The LRC for the encoding (ISO only) <code>ID_LRC_DEFAULT</code> <code>ID_LRC_OFF</code> <code>ID_LRC_ODD</code> <code>ID_LRC_EVEN</code>

**Return Values**

<code>ID_SUCCESS</code>	Operation completed successfully.
<code>ID_INVALID_SESSION</code>	Invalid session handle
<code>ID_PARAM_ERROR</code>	Invalid parameter
<code>ID_ERROR</code>	Other SDK error

**Remarks**

Both ASCII and Unicode versions of this function exist (*ID\_EncodeMagA* and *ID\_EncodeMagW* respectively). The appropriate function is selected for the user dependent on the Unicode setting of the application, thus the user need only access the generic function.

### 8.3 ID ReadMag

Reads data from the magnetic stripe on the card

```
int ID_ReadMag(HANDLE hSession,
               VOID *pMagData,
               int iEncodingSpec);
```

Enduro	✓
Ultima	✓
Generation 2	✓

#### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

*pMagData*

Pointer to a structure to be filled with the magnetic stripe data.

*iEncodingSpec*

The encoding method in use

```
ID_ENCODING_ISO
ID_ENCODING_JIS2
ID_ENCODING_RAW (Ultima Only)
```

#### Structures (Enduro Printers)

```
typedef struct
{
    DWORD    msv_id;
    DWORD    msg_len;
    DWORD    tk1_pass;
    DWORD    tk2_pass;
    DWORD    tk3_pass;
    DWORD    tk1_len;
    DWORD    tk2_len;
    DWORD    tk3_len;
    RAW_DATA raw;
} ID_MAGDATA;
```

```
typedef struct
{
    char    tk1[172];
    char    tk2[172];
    char    tk3[172];
} RAW_DATA;
```

#### Members:

- msv\_id*                      Unique ID to distinguish this message
- msg\_len*                    Size of message, including this field
- tk1\_pass*                    TRUE if Track 1 passed; FALSE if failed or not tested
- tk2\_pass*                    Similar for Track 2
- tk3\_pass*                    Similar for Track 3
- tk1\_len*                     Number of bytes returned for Track 1 from start sentinel to LRC inclusive
- tk2\_len*                    Similar for Track 2

tk3_len	Similar for Track 3
raw	Raw data for each track

### Structures (Ultima & Generation 2 Printers)

```
typedef struct
{
    char data[112];
} TRACK_DATA;

typedef struct
{
    TRACK_DATA track[3];
} ID_MAGDATA2;
```

### Return Values

ID_SUCCESS	Operation completed successfully.
ID_INVALID_SESSION	Invalid session handle
ID_PARAM_ERROR	Invalid parameter
ID_TIMEOUT	A 30-second period has elapsed without receiving any magnetic stripe data.
ID_ERROR	Other SDK error

### Remarks

- Enduro Printers - data retrieved using this SDK function is automatically decoded from raw data format, unlike the deprecated API function (*ReadMagStripe*) where it was necessary to manually decode the data by calling a further API function (*DecodeMagData*).
- Ultima & Generation 2 Printers - there is also the option of RAW encoding in which case the data returned is as read from the card, with any decoding necessary being done by the calling application.
- The data structure returned is dependent on the type of printer in use, hence the use of a generic pointer (VOID \*) as the parameter for the buffer to be populated.



## 8.4 ID ReadMagTracks

Enduro	✓
Ultima	✓
Generation 2	✓

Reads data from the specified tracks of the magnetic stripe on the card

```
int ID_ReadMag(HANDLE hSession,
               VOID *pMagData,
               int iEncodingSpec,
               BYTE Tracks);
```

### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

*pMagData*

Pointer to a structure to be filled with the magnetic stripe data.

*iEncodingSpec*

The encoding method in use

```
ID_ENCODING_ISO
ID_ENCODING_JIS2
ID_ENCODING_RAW (Ultima & Generation 2 Only)
```

*Tracks*

Bit mask defining the tracks to be read:

B0	Track 1
B1	Track 2
B2	Track 3
All Others	Unused

### Structures

See *ID\_ReadMag* (Section 8.3)

### Return Values

ID_SUCCESS	Operation completed successfully.
ID_INVALID_SESSION	Invalid session handle
ID_PARAM_ERROR	Invalid parameter
ID_TIMEOUT	A 30-second period has elapsed without receiving any magnetic stripe data.
ID_ERROR	Other SDK error

### Remarks

- Enduro Printers - data retrieved using this SDK function is automatically decoded from raw data format (unlike the deprecated API function *ReadMagStripe* where it was necessary to manually decode the data by further calling *DecodeMagData*).
- Ultima & Generation 2 Printers - there is also the option of RAW encoding in which case the data returned is as read from the card, no decoding being done.
- The data structure returned is dependent on the type of printer in use, hence the use of a generic pointer (VOID \*) as the parameter for the buffer to be populated.

- For JIS2 Encoding, only Track 1 is used, so the Tracks parameter is ignored, and the operation is the same as for *ID\_ReadMag*.

## 8.5 ID MagStart

Reads/writes the offset position for the start of magnetic encoding.

```
int ID_MagStart(HANDLE      hSession,
                ID_READWRITE ReadWrite,
                DWORD        *Offset);
```

Enduro	
Ultima	✓
Generation 2	✓

### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

*ReadWrite*

The action to be performed:

```
ID_READ
ID_WRITE
```

*Offset*

Pointer to a DWORD containing the magnetic start offset

### Return Values

ID_SUCCESS	Operation completed successfully.
ID_INVALID_SESSION	Invalid session handle
ID_PARAM_ERROR	Invalid parameter
ID_ERROR	Other SDK error

## 9 PRINTING FUNCTIONS

The SDK's printing functions are a very basic set of tools, based on the Windows GDI, offering limited functionality. They have been provided specifically to give inexperienced developers a simplified and basic printing interface, allowing them to draw text, shapes, lines and images to the front and rear of a card.

Due to the basic nature of these functions, we recommend that experienced developers of printing applications should use the Windows GDI directly – especially when more complex features are required.

### 9.1 Drawing Canvas

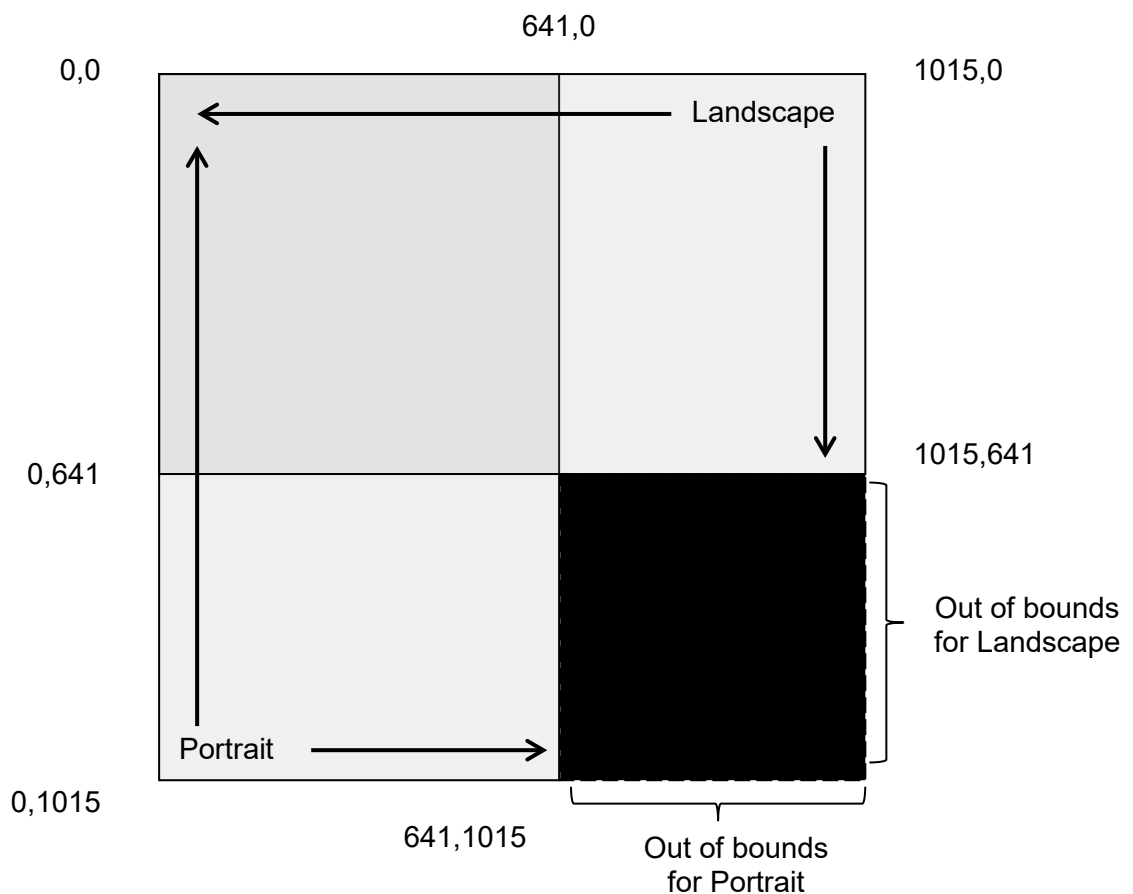
All drawing is done on a 'canvas', with 2 canvasses being provided for each side of the card:

Colour canvas - Objects drawn on this canvas are printed in composite colour (Black items will be composite black)

Resin canvas - Objects drawn on this canvas are printed in K resin (i.e. black)

Printing uses the current orientation as set in the driver. This can be modified if necessary, using the *ID\_CardSettings* function.

Any objects (or parts of objects) that are printed to a canvas but lay outside of the area for the current orientation, will not be visible.



(N.B. Dimensions shown are for Enduro family printers)

**Figure 4 – Canvas Dimensions & Orientation**

## **9.2 High Resolution (600 DPI) Printing**

### **9.2.1 Canvas Size**

Some Generation 2 printer models have the capability of printing in 600 dpi (in the K resin plane and on the long axis of the card only).

When printing in 300 dpi, the canvas size is 1013 pixels square for Generation 2 printers (1016 pixels square for all other printers).

Accordingly, when printing in 600 dpi on printers that will support this resolution, the canvas width is 2026 pixels square.

### **9.2.2 Selection**

Selection of 600 dpi printing is made using the *ID\_Resolution* function (see section 5.19). This function updates the driver resolution setting **AND** sets the resolution to be used for the current SDK session. It **MUST** be used if 600 dpi printing is required, otherwise 300 dpi (the default setting) will be used for all subsequent printing in the session.

### 9.3 ID CanvasInit

Initialises the 'canvas' (the drawing area) for the given card side, returning a Handle to the Device Context (HDC) to the calling application.

```
int ID_CanvasInit(HANDLE    hSession,
                  HDC *     pHDC,
                  ID_SIDE   Canvas);
```

Enduro	✓
Ultima	✓
Generation 2	✓

#### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

*pHDC*

Pointer which returns the HDC of the specified canvas.

*Canvas*

The canvas being initialised

- ID\_FRONT
- ID\_FRONT\_RESIN
- ID\_BACK
- ID\_BACK\_RESIN

#### Return Values

ID_SUCCESS	Operation completed successfully.
ID_INVALID_SESSION	Invalid session handle
ID_PARAM_ERROR	Invalid parameter
ID_ERROR	Other SDK error
ID_SESSION_ACTIVE	HDC is already in use

#### Remarks

- The SDK provides a basic set of drawing functions to allow a user application the ability to produce simple card designs. The HDC allows an application to access the canvas and use the full range of graphics functions provided by the Windows operating system through GDI and GDI+ should it require to do so. The card can then be printed (using *ID\_PrintCard* SDK function) in the normal manner.
- The dpi of the canvas is set automatically to the current dpi setting (300 or 600) of the printer driver.

## 9.4 ID CanvasDelete

Deletes the canvas for the given side

```
int ID_CanvasDelete(HANDLE hSession,
                   ID_SIDE Canvas);
```

Enduro	✓
Ultima	✓
Generation 2	✓

### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

*Canvas*

The canvas to be deleted

```
ID_FRONT
ID_FRONT_RESIN
ID_BACK
ID_BACK_RESIN
```

### Return Values

ID_SUCCESS	Operation completed successfully.
ID_INVALID_SESSION	Invalid session handle
ID_PARAM_ERROR	Invalid parameter
ID_ERROR	Other SDK error

## 9.5 ID DrawText

Enduro	✓
Ultima	✓
Generation 2	✓

Draws text on the given side using the provided parameters

```
int ID_DrawText(HANDLE      hSession,
                ID_SIDE     CanvasID,
                PIDTEXTDEF  pTextDef);
```

### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

*CanvasID*

The canvas on which the text is to be drawn

```
ID_FRONT
ID_FRONT_RESIN
ID_BACK
ID_BACK_RESIN
```

*pTextDef*

Pointer to an *IDTextDef* structure that defines the text to be drawn

### Structures

```
typedef struct
{
    char */wchar * Text;
    int           X;
    int           Y;
    int           Angle;
    char */wchar * FontName;
    int           Size;
    int           Colour;
    int           Style;
} IDTEXTDEF
```

### Members:

Text	Text to be drawn on the canvas
X, Y	Start co-ordinates for the text
Angle	Angle that the text is to be drawn (in 0.1° e.g. 30° = 300)
FontName	Name of the font to be used
Size	Font Size
Colour	Font Colour
Style	Font Style (Bitmask): BOLD           0x01 ITALIC         0x02 UNDERLINE    0x04 STRIKEOUT   0x08



**Return Values**

ID_SUCCESS	Operation completed successfully.
ID_INVALID_SESSION	Invalid session handle
ID_PARAM_ERROR	Invalid parameter
ID_ERROR	Other SDK error

**Remarks**

Both ASCII and Unicode versions of this function exist (*ID\_DrawTextA* and *ID\_DrawTextW* respectively). The appropriate function is selected for the user dependent on the Unicode setting of the application, thus the user need only access the generic function.

## 9.7 ID DrawShape

Enduro	✓
Ultima	✓
Generation 2	✓

Draws a shape on the given side using the provided parameters

```
int ID_DrawShape(HANDLE hSession,
                 ID_SIDE CanvasID,
                 PIDSHAPEDEF pShapeDef);
```

### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

*CanvasID*

The canvas on which the shape is to be drawn

```
ID_FRONT
ID_FRONT_RESIN
ID_BACK
ID_BACK_RESIN
```

*pShapeDef*

Pointer to an *IDShapeDef* structure that defines the shape to be drawn

### Structures

```
typedef struct
{
    int Shape;
    int PenColour;
    int PenWidth;
    int FillColour;
    RECT Bound; (Rectangle in .Net Shim)
    int p1;
    int p2;
    int p3;
    int p4;
} IDSHAPEDEF;
```

### Members:

Shape	The shape to be drawn
ID_SHAPE_RECTANGLE	Square/Rectangle
ID_SHAPE_ELLIPSE	Circle/Ellipse
ID_SHAPE_ROUNDRECT	Square/rectangle with rounded corners
ID_SHAPE_PIE	Pie
ID_SHAPE_CHORD	Chord
PenColour	Colour of the pen used to draw the outline of the shape
PenWidth	Width of the pen used to draw the outline
FillColour	Colour of the pen used to fill in the shape
Bound	Windows RECT structure defining the boundary of the shape

p1, p2, p3, p4 Dependent on the shape selected:-

Shape	p1	p2	p3	p4
ID_SHAPE_RECTANGLE	Not Used			
ID_SHAPE_ELLIPSE				
ID_SHAPE_ROUNDRECT	Corner Width	Corner Height	Not Used	
ID_SHAPE_PIE	Radial 1 X Co-ord	Radial 1 Y Co-ord	Radial 2 X Co-ord	Radial 2 Y Co-ord
ID_SHAPE_CHORD				

### Return Values

ID_SUCCESS	Operation completed successfully.
ID_INVALID_SESSION	Invalid session handle
ID_PARAM_ERROR	Invalid parameter
ID_ERROR	Other SDK error

### Remarks

It is important to note that the structure definition for the `Bound` field is a `RECT` for C++, and in the Java shim (defined by `Left`, `Top`, `Right`, `Bottom` co-ordinates) but the equivalent in the .Net shim (for C# and VB) is a `Rectangle` structure (defined in the constructor by `Left`, `Top`, `Width`, `Height`)

## 9.9 ID DrawLine

Enduro	✓
Ultima	✓
Generation 2	✓

Draws a line on the given canvas using the provided parameters

```
int ID_DrawLine(HANDLE hSession,
                ID_SIDE CanvasID,
                PIDLINEDEF pLineDef);
```

### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

*CanvasID*

The canvas on which the line is to be drawn

```
ID_FRONT
ID_FRONT_RESIN
ID_BACK
ID_BACK_RESIN
```

*pLineDef*

Pointer to an *IDLineDef* structure that defines the line to be drawn

### Structures

```
typedef struct
{
    int Colour;
    int Width;
    int StartX;
    int StartY;
    int EndX;
    int EndY;
} IDLINEDEF;
```

### Members:

Colour	Colour of the pen used for the line
Width	Width of the pen used for the line
StartX, StartY	Start co-ordinates of the line
EndX, EndY	End co-ordinates of the line

### Return Values

ID_SUCCESS	Operation completed successfully.
ID_INVALID_SESSION	Invalid session handle
ID_PARAM_ERROR	Invalid parameter
ID_ERROR	Other SDK error

### 9.11 ID DrawImage

Draws an image (BMP, JPG, PNG, TIF) on the given side using the provided parameters

```
int ID_DrawImage(HANDLE hSession,
                 ID_SIDE CanvasID,
                 PIDIMAGEDEF pImageDef);
```

Enduro	✓
Ultima	✓
Generation 2	✓

#### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

*CanvasID*

The canvas on which the image is to be drawn

- ID\_FRONT
- ID\_FRONT\_RESIN
- ID\_BACK
- ID\_BACK\_RESIN

*pImageDef*

Pointer to an *IDImageDef* structure that defines the image to be drawn

#### Structures

```
typedef struct
{
    char */wchar *  Filename;
    int              X;
    int              Y;
    int              p1;
    int              p2;
} IDIMAGEDEF;
```

#### Members:

- Filename           Filename (including path) of the image to be printed.
- X, Y               Co-ordinates for the top left corner of the image.
- p1, p2             These parameters define the scaling and/or boundary of the image

Structure members	p1	p2
Image printed at original size	0	0
Image scaled (maintaining aspect ratio)	Scale Factor (%)	0
Image scaled to fit the bounding rectangle	Bottom Right X	Bottom Right Y

**Return Values**

ID_SUCCESS	Operation completed successfully.
ID_INVALID_SESSION	Invalid session handle
ID_PARAM_ERROR	Invalid parameter
ID_ERROR	Other SDK error

**Remarks**

- Both ASCII and Unicode versions of this function exist (ID\_DrawImageA and ID\_DrawImageW respectively). The appropriate function is selected for the user dependent on the Unicode setting of the application, thus the user need only access the generic function.
- Images drawn to the K resin plane (either front or back) MUST be in negative i.e. white on black.

## 9.13 ID\_DrawMagText

Provides text for magnetic encoding.

```

int ID_DrawMagText(HANDLE      hSession,
                  ID_SIDE     SideID,
                  PID_MAGTEXT  pMagText)

```

Enduro	✓
Ultima	✓
Generation 2	✓

### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

*SideID*

Defines the side which is being written/read.

ID\_FRONT

ID\_BACK

*pMagText*

Pointer to an *IDMagText* structure that defines the text to be encoded

### Structures

```

typedef struct
{
    int          Track;
    char */wchar * Data;
} IDMAGTEXT

```

### Members:

Track           Track to be encoded

Data            Data to be encoded, including start/end sentinels

### Return Values

ID_SUCCESS	Operation completed successfully.
ID_INVALID_SESSION	Invalid session handle
ID_PARAM_ERROR	Invalid parameter
ID_ERROR	Other SDK error

### Remarks

- Both ASCII and Unicode versions of this function exist (*ID\_DrawMagTextA* and *ID\_DrawMagTextW* respectively). The appropriate function is selected for the user dependent on the Unicode setting of the application, thus the user need only access the generic function.
- In the case of multiple calls of this function for a given track, only the text/data from the last call will be encoded.
- Data is encoded as provided so start/end sentinels **MUST** be included by the calling application.

**9.14 ID PrintCard**

Prints the card using the canvas (or canvasses) currently set up.

```
int ID_PrintCard(HANDLE hSession)
```

Enduro	✓
Ultima	✓
Generation 2	✓

**Parameters**

*hSession*

The session handle returned by *ID\_OpenSession*.

**Return Values**

ID_SUCCESS	Operation completed successfully.
ID_INVALID_SESSION	Invalid session handle
ID_PARAM_ERROR	Invalid Parameter
ID_ERROR	Other SDK Error

**Remarks**

- Should be followed by use of the *ID\_WaitForPrinter* function to allow completion of the printer action (N.B. Not required for Generation 2).
- When the print is complete, all canvasses are deleted.



## 10 PRINTING SUPPORT FUNCTIONS

### 10.1 *ID\_GetDevmode*

Reads the devmode from the driver.

```
int ID_GetDevmode(HANDLE hSession,
                  PDEVMODE pBuffer,
                  int *size);
```

Enduro	✓
Ultima	✓
Generation 2	✓

#### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

*pBuffer*

A pointer to the buffer that is to receive the devmode. If NULL, the function fails with *ID\_MORE\_DATA* and *size* returns the buffer size required.

*size*

Pointer to an integer that contains the buffer size (in bytes)

#### Return Values

<i>ID_SUCCESS</i>	Operation completed successfully.
<i>ID_INVALID_SESSION</i>	Invalid session handle
<i>ID_MORE_DATA</i>	Buffer presented is too small
<i>ID_ERROR</i>	Other SDK error

#### Remarks

- This function is used to apply driver settings changes made using SDK functions prior to printing. Having obtained the devmode, it can be applied as needed (e.g. *CreateDC*) in the GDI/native printing process.
- If *pBuffer* is set to NULL or *size* is too small, the function fails (*ID\_MORE\_DATA*) and returns the required number of bytes for the buffer in the variable pointed to by *size*.

## 10.2 ID\_SetDevmode

Sets the devmode in the driver.

```
int ID_SetDevmode(HANDLE hSession,
                 PDEVMODE pBuffer,
                 int size);
```

Enduro	✓
Ultima	✓
Generation 2	✓

### Parameters

*hSession*

The session handle returned by *ID\_OpenSession*.

*pBuffer*

A pointer to the buffer that contains the devmode.

*size*

An integer that contains the buffer size (in bytes)

### Return Values

ID_SUCCESS	Operation completed successfully.
ID_INVALID_SESSION	Invalid session handle
ID_ERROR	Other SDK error

### Remarks

- This function is used to apply an updated devmode to the driver.
- If *pBuffer* is set to NULL or *size* is too small, the function fails (*ID\_ERROR*).

### **10.3ID UpdateDC**

Updates the given Device Context (DC) with the current devmode settings of the driver.

```
int ID_UpdateDC(HANDLE hSession,
                HDC      hDC);
```

Enduro	✓
Ultima	✓
Generation 2	✓

#### **Parameters**

*hSession*

The session handle returned by *ID\_OpenSession*.

*hDC*

The HDC of the Device Context to be updated

#### **Return Values**

ID_SUCCESS	Operation completed successfully.
ID_INVALID_SESSION	Invalid session handle
ID_ERROR	Other SDK error

#### **Remarks**

- This function is used to apply driver settings changes made using SDK functions. An existing Device Context (DC) is updated as needed for the GDI/native printing process.
- This function is **NOT** available via the shims

## **10.4ID UpdateDevmode**

Enduro	✓
Ultima	✓
Generation 2	✓

### **N.B. .Net Shim Only**

Updates the given PrintDocument with the current devmode settings of the driver.

```
Return ID_UpdateDC (IntPtr hSession,
                    ref PrintDocument pd);
```

### **Parameters**

*hSession*

The session handle returned by *OpenSession*.

*pd*

Reference to a PrintDocument class

### **Return Values**

Success	Operation completed successfully.
InvalidSession	Invalid session handle
MoreData	Buffer presented is too small
Error	Other SDK error

### **Remarks**

- This function is used to apply driver settings changes made using SDK functions. An existing instance of a PrintDocument class is updated as needed for the native printing process.
- This function is **ONLY** available via the .Net shim.

**10.5ID PrinterPrefs**

Displays the Printer Preferences dialog, and applies any changes that are made, to the driver settings.

```
int ID_PrinterPrefs(HANDLE hSession,
                   HWND hWnd);
```

Enduro	✓
Ultima	✓
Generation 2	✓

**Parameters**

*hSession*

The session handle returned by *ID\_OpenSession*.

*hWnd*

The handle of the Window of the application

**Return Values**

ID_SUCCESS	Operation completed successfully.
ID_INVALID_SESSION	Invalid session handle
ID_ERROR	Other SDK error

## **11 ACKNOWLEDGEMENTS**

This software uses the pugixml library (<http://pugixml.org>).

pugixml is Copyright © 2006-2019 Arseny Kapoulkine.

pugixml is released under the MIT license.

## APPENDIX A - API FUNCTIONS

To provide a common naming convention, these functions, originally defined in the API, have been superseded with an equivalent in the SDK. They remain in the SDK to maintain backwards compatibility with applications already developed.

Original API Function	Corresponding SDK Function
<i>CleanPrinter</i>	<b>ID_CleanPrinter</b>
<i>DecodeMagData</i>	<i>N/A - Decoding is performed automatically in SDK function ID_ReadMag</i>
<i>DisableStatusReporting</i>	<b>ID_CloseSession</b>
<i>EnableStatusReporting</i>	<b>ID_OpenSession</b>
<i>EjectCard</i>	<b>ID_EjectCard</b>
<i>EncodeMagStripe</i>	<p><b>Overview</b></p> <p><b>Magnetic</b> encoding can be implemented in one of several ways:-            If printing via GDI, either ...            include the encoding data with the GDI image data as text strings in the form: ~TRACK NUMBER,DATA (e.g. "~1,MAGDATA").</p> <p style="padding-left: 40px;">If not smart encoding, this option has the advantage that the application does not need to manage the serial process of encoding and then printing. Therefore, multiple card images and their mag data can be created as a single multi-page print job and sent through the Windows driver, and the application can continue with its next tasks.</p> <p style="padding-left: 20px;">c) Use the ID_EncodeMag function (as described in Section 8.2).</p> <p style="padding-left: 40px;">This option enables the application to control the individual serial processes of encoding and then printing, which may important especially when also smart encoding.</p> <p>3. If printing via the SDK's printing functions, either...</p> <p style="padding-left: 20px;">a) Use the ID_DrawMagText function (as described in Section 9.8).</p> <p style="padding-left: 40px;">If not smart encoding, this option has the advantage that the application does not need to manage the serial process of encoding and then printing. Therefore, multiple card images and their mag data can be created and sent through the Windows driver, and the application can continue with its next tasks.</p> <p style="padding-left: 20px;">b) Use the ID_EncodeMag function (as described in Section 8.2).</p> <p style="padding-left: 40px;">This option enables the application to control the individual serial processes of encoding and then printing, which may important especially when also smart encoding.</p> <p style="text-align: center;">ID_EncodeMag</p>

<i>EraseCard</i>	<b>ID_EraseCard</b>								
<i>ErrorResponse</i>	<b>ID_ErrorResponse</b>								
<i>FeedCard</i>	<b>ID_FeedCard</b>								
<i>FlipCard</i>	<b>ID_FlipCard</b>								
<i>GeneralCommand</i>	<b>ID_GeneralCommand</b>								
<i>GetAPIVersion</i>	<p><b>ID_SDKVersion</b> Returns the version number of the SDK.</p> <pre>int ID_SDKVersion(HANDLE hSession,                   SDKVERSION * pVersion);</pre> <p><b>Parameters</b></p> <p><i>hSession</i> The session handle returned by <i>ID_OpenSession</i>.</p> <p><i>pVersion</i> Pointer to an SDKVersion structure</p> <p><b>Structures</b></p> <pre>typedef struct {     DWORD Major;     DWORD Minor;     DWORD Build;     DWORD Private; } SDKVERSION;</pre> <p><b>Members:</b></p> <p>Combined in the form:</p> <p style="text-align: center;"><b>Major.Minor.Build.Private</b></p> <p>the members of the structure give the SDK version number (e.g. Version 3.0.1.2)</p> <p><b>Return Values</b></p> <table style="width: 100%; border: none;"> <tr> <td style="width: 50%;">ID_SUCCESS</td> <td>Operation completed successfully.</td> </tr> <tr> <td>ID_INVALID_SESSION</td> <td>Invalid session handle</td> </tr> <tr> <td>ID_PARAM_ERROR</td> <td>Invalid parameter</td> </tr> <tr> <td>ID_ERROR</td> <td>Other SDK error</td> </tr> </table> <p>ID_SDK</p>	ID_SUCCESS	Operation completed successfully.	ID_INVALID_SESSION	Invalid session handle	ID_PARAM_ERROR	Invalid parameter	ID_ERROR	Other SDK error
ID_SUCCESS	Operation completed successfully.								
ID_INVALID_SESSION	Invalid session handle								
ID_PARAM_ERROR	Invalid parameter								
ID_ERROR	Other SDK error								
<i>GetLastEnduroMessage</i> <i>GetLastPrinterMessage</i>	<b>ID_LastMessage</b>								
<i>GetConnectionType</i>	<b>ID_ConnectionType</b>								
<i>GetEnduroInfo</i>	<i>Already Deprecated in the API – Use SDK function ID_PrinterInfo</i>								
<i>GetPrinterInfo</i>	<b>ID_PrinterInfo</b>								



<i>GetPrinterStatus</i>	<b><i>ID_PrinterStatus</i></b>
<i>GetPrinterType</i>	<b><i>ID_PrinterType</i></b>
<i>PrintTestCard</i>	<b><i>ID_PrintTestCard</i></b>
<i>RestartPrinter</i>	<b><i>ID_RestartPrinter</i></b>
<i>ReadMagData</i>	<i>Already Deprecated in the API – Use SDK function <b>ID_ReadMag</b></i>
<i>RequestMagData</i>	
<i>ReadMagStripe</i>	<b><i>ID_ReadMag</i></b>
<i>SetEjectMode</i>	<b><i>ID_EjectMode</i></b>
<i>SetEraseSpeed</i>	<b><i>ID_EraseSpeed</i></b>
<i>SetSmartMode</i>	<b><i>ID_SmartMode</i></b>
<i>SetSmartLocation</i>	<b><i>ID_SmartOffset</i></b>
<i>WaitForPrinter</i>	<b><i>ID_WaitForPrinter</i></b>