ExpressCard 1000

PROGRAMMING REFERENCE MANUAL
WINDOWS XML SPECIFICATIONS

PART NUMBER 99875368-6

JUNE 2010

MAGTEK®

REGISTERED TO ISO 9001:2000

1710 Apollo Court
Seal Beach, CA 90740
Phone: (562) 546-6400
FAX: (562) 546-6301
Technical Support: (651) 415-6800
www.magtek.com

REVISIONS

Rev Number	Date	Notes
1	Sept 7, 2007	Initial Release.
2	Mar 4, 2008	Added commands; misc. formatting changes.
3	Oct 13, 2008	Identified Optional keys with (Default If Omitted). Demonstrations for minimum required keys.
4	Mar 2, 2009	Removed D2T2CardCount from DeviceInfo response. Added support for International printing using Base64 encoding.
5	Sept 14, 2009	Added KeyLine and OVERLAY to the D2T2 section Added KeyLine to Emboss section. Added Queueing, Queue Notification, and Encryption. Updated return codes and job states Updated DeviceStatus and DeviceInfo with device options and queue status.
6	June 7, 2010	Added SendCancelOnError to CommandInfo section.

SOFTWARE LICENSE AGREEMENT

IMPORTANT: YOU SHOULD CAREFULLY READ ALL THE TERMS, CONDITIONS AND RESTRICTIONS OF THIS LICENSE AGREEMENT BEFORE INSTALLING THE SOFTWARE PACKAGE. YOUR INSTALLATION OF THE SOFTWARE PACKAGE PRESUMES YOUR ACCEPTANCE OF THE TERMS, CONDITIONS, AND RESTRICTIONS CONTAINED IN THIS AGREEMENT. IF YOU DO NOT AGREE WITH THESE TERMS, CONDITIONS, AND RESTRICTIONS, PROMPTLY RETURN THE SOFTWARE PACKAGE AND ASSOCIATED DOCUMENTATION TO THE ABOVE ADDRESS, ATTENTION: CUSTOMER SUPPORT.

TERMS, CONDITIONS, AND RESTRICTIONS

MagTek, Incorporated (the "Licensor") owns and has the right to distribute the described software and documentation, collectively referred to as the "Software".

LICENSE: Licensor grants you (the "Licensee") the right to use the Software in conjunction with MagTek products. LICENSEE MAY NOT COPY, MODIFY, OR TRANSFER THE SOFTWARE IN WHOLE OR IN PART EXCEPT AS EXPRESSLY PROVIDED IN THIS AGREEMENT. Licensee may not decompile, disassemble, or in any other manner attempt to reverse engineer the Software. Licensee shall not tamper with, bypass, or alter any security features of the software or attempt to do so.

TRANSFER: Licensee may not transfer the Software or license to the Software to another party without the prior written authorization of the Licensor. If Licensee transfers the Software without authorization, all rights granted under this Agreement are automatically terminated.

COPYRIGHT: The Software is copyrighted. Licensee may not copy the Software except for archival purposes or to load for execution purposes. All other copies of the Software are in violation of this Agreement.

TERM: This Agreement is in effect as long as Licensee continues the use of the Software. The Licensor also reserves the right to terminate this Agreement if Licensee fails to comply with any of the terms, conditions, or restrictions contained herein. Should Licensor terminate this Agreement due to Licensee's failure to comply, Licensee agrees to return the Software to Licensor. Receipt of returned Software by the Licensor shall mark the termination.

LIMITED WARRANTY: Licensor warrants to the Licensee that the disk(s) or other media on which the Software is recorded are free from defects in material or workmanship under normal use.

THE SOFTWARE IS PROVIDED AS IS. LICENSOR MAKES NO OTHER WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Because of the diversity of conditions and PC hardware under which the Software may be used, Licensor does not warrant that the Software will meet Licensee specifications or that the operation of the Software will be uninterrupted or free of errors.

IN NO EVENT WILL LICENSOR BE LIABLE FOR ANY DAMAGES, INCLUDING ANY LOST PROFITS, LOST SAVINGS, OR OTHER INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE, OR INABILITY TO USE, THE SOFTWARE. Licensee's sole remedy in the event of a defect in material or workmanship is expressly limited to replacement of the Software disk(s) if applicable.

GOVERNING LAW: If any provision of this Agreement is found to be unlawful, void, or unenforceable, that provision shall be removed from consideration under this Agreement and will not affect the enforceability of any of the remaining provisions. This Agreement shall be governed by the laws of the State of California and shall inure to the benefit of MagTek, Incorporated, its successors or assigns.

ACKNOWLEDGMENT: LICENSEE ACKNOWLEDGES THAT HE HAS READ THIS AGREEMENT, UNDERSTANDS ALL OF ITS TERMS, CONDITIONS, AND RESTRICTIONS, AND AGREES TO BE BOUND BY THEM. LICENSEE ALSO AGREES THAT THIS AGREEMENT SUPERSEDES ANY AND ALL VERBAL AND WRITTEN COMMUNICATIONS BETWEEN LICENSOR AND LICENSEE OR THEIR ASSIGNS RELATING TO THE SUBJECT MATTER OF THIS AGREEMENT.

QUESTIONS REGARDING THIS AGREEMENT SHOULD BE ADDRESSED IN WRITING TO MAGTEK, INCORPORATED, ATTENTION: CUSTOMER SUPPORT, AT THE ABOVE ADDRESS, OR E-MAILED TO support@magtek.com

TABLE OF CONTENTS

FIGURES

SECTION 1. OVERVIEW

The sections of this manual are as follows:

Section 1. Overview
Section 2. Architecture and Operation
Section 3. Key-Value Pair Specifications – describes the options for the print process, sent by the application to ExpressCard 1000
Section 4. Key-Value Pair Specifications- describes the options for the print process, returned to the application by ExpressCard 1000
Section 5. Independent and Smart Card Commands
Appendix A. Code Examples
Appendix B. Return Codes and Messages

REQUIREMENTS

The following item is required for software installation:

P/N 33090060 ExpressCard Software Development Kit (This CD installs ExpressCard 1000 Demo Program and MTECSDK.dll)

INTRODUCTION

The ExpressCard 1000 (EC1000) is a desktop card production/personalization system that has the capability to instantly issue financial or ID cards in approximately 2 to 3 minutes per card.

The primary functions and features of the EC1000 are as follows:

- Dual card-input hoppers (100 cards per hopper)
- Manual Insertion Slot (Single card entry)
- Encoder module (Magstripe/smartcard/contactless-card/MagnePrint)
- Thermal/Dye-Sublimation Printer (Print on dual sides of card)
- Embosser/Rear-Indent module (Mechanically embosses/indents characters on card surface)
- Exit module (Exits completed card, or stores rejected card in internal hopper)
- Secure enclosure that can logically detect/permit access to the device
- Touchscreen LCD display that provides for primary user interface.
- Ethernet interface
- XML protocol, using MagTek defined key-value pairs.

This manual is intended for use by Application Programmers who want to develop custom application code to directly control and interface to the EC1000 device.
In regard to interface options, programmers can choose from 2 approaches to facilitate an interface to the device:
1. Directly utilize XML protocol and MagTek defined key-value pairs.
2. Utilize a MagTek API DLL.

Both options are effective in facilitating an interface to the device.
However, there is probably an advantage in utilizing the MagTek API, as it will ensure that if any changes occur to the underlying hardware interface of the device (e.g. USB, RS-232, Firewire, etc…), the Application layer will be abstracted from these changes, and will not have to make any code changes.

This Programmer's reference manual provides definitions for both the API DLL interface, and the XML KV-Pair definitions. It is recommended that the application programmer review all of this information, so as to gain a broad perspective of the command, status, and functional aspects the device interface.
Of primary importance, is to review the code samples contained in the appendices of this document. The code samples provide clear instructions regarding the operational sequence of events, as well as the code formatting that is typically used during interface to the device.
By starting with these samples, the application programmer can build more complex functionality into their overall application.

SECTION 2. EXPRESSCARD 1000 SOFTWARE ARCHITECTURE

The architecture of the system is shown in Figure 2-1. Descriptions of the terms and operations used follow the illustration.

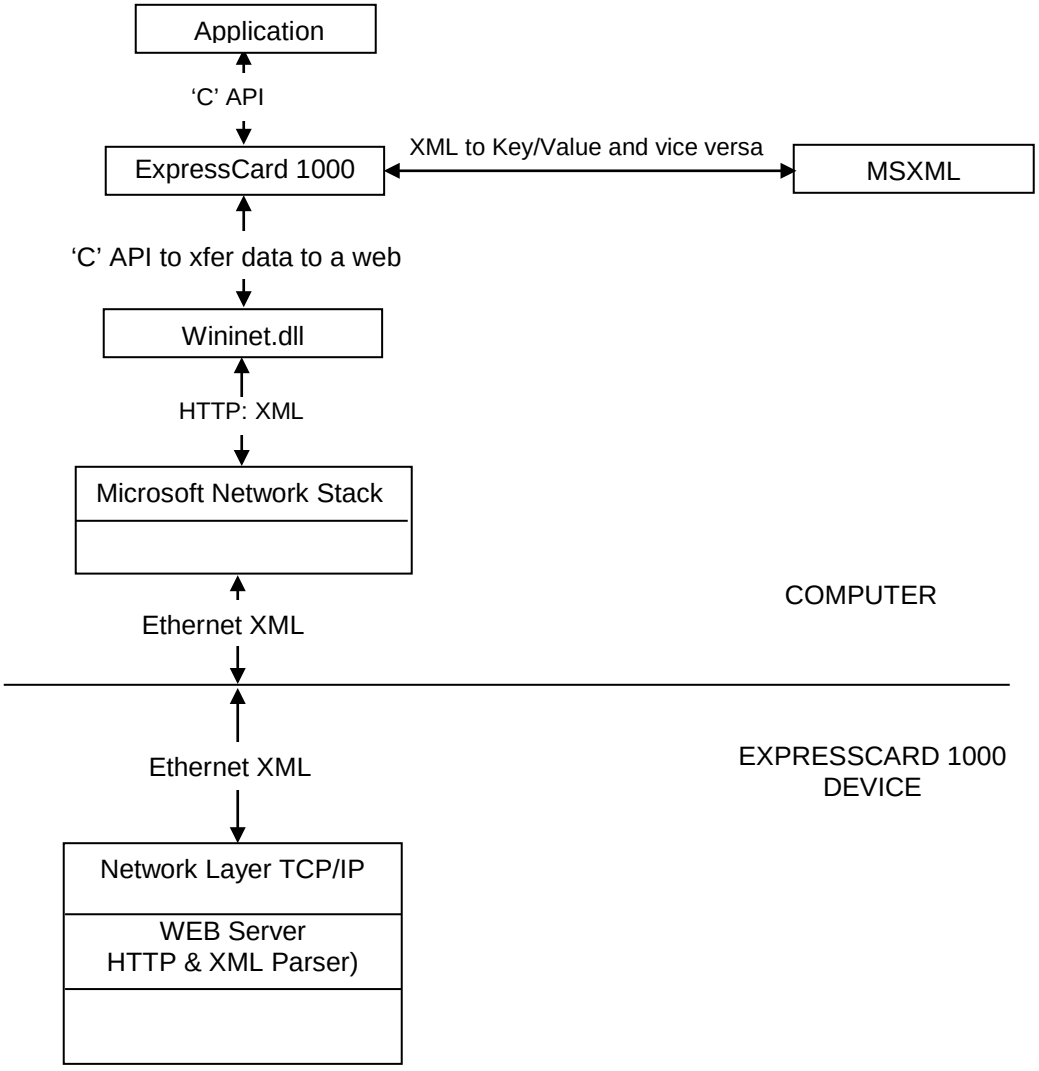


Figure 2-1. ExpressCard 1000 Software Architecture

TERM DESCRIPTION

MTECSDK.dll:
Provides API to upper level application to talk to the device. The application does not have any knowledge of how the device is connected to the computer. Thus it does not expose the transport protocol to the application. We are using HTTP protocol with XML to communicate with the device. The **MTECSDK.dll** uses **wininet.dll** to talk to internet protocols. The **MTECSDK.dll.** provides functions to convert the print request into XML format using **msxml4.dll** and then send it to the device using **wininet.dll**. It also provides functions to convert the response from the device (XML format) to key/value pairs.

msxml4.dll: Microsoft XML Parser**.**
MTECSDK.dll uses **msxml4.dll** to convert the key/value pair to XML language.
MTECSDK.dll also uses the **msxml4.dll** to convert the XML data back to key/ value pair.

wininet.dll: MFC Win32 Internet Extension. **wininet.dll**, provide access to common Internet protocols, including Gopher, FTP, and HTTP. **MTECSDK.dl** uses **wininet.dll** to establish an Internet connection with ExpressCard 1000 device. It then communicates with ExpressCard 1000 using GET and POST requests provide by **wininet.dll**.

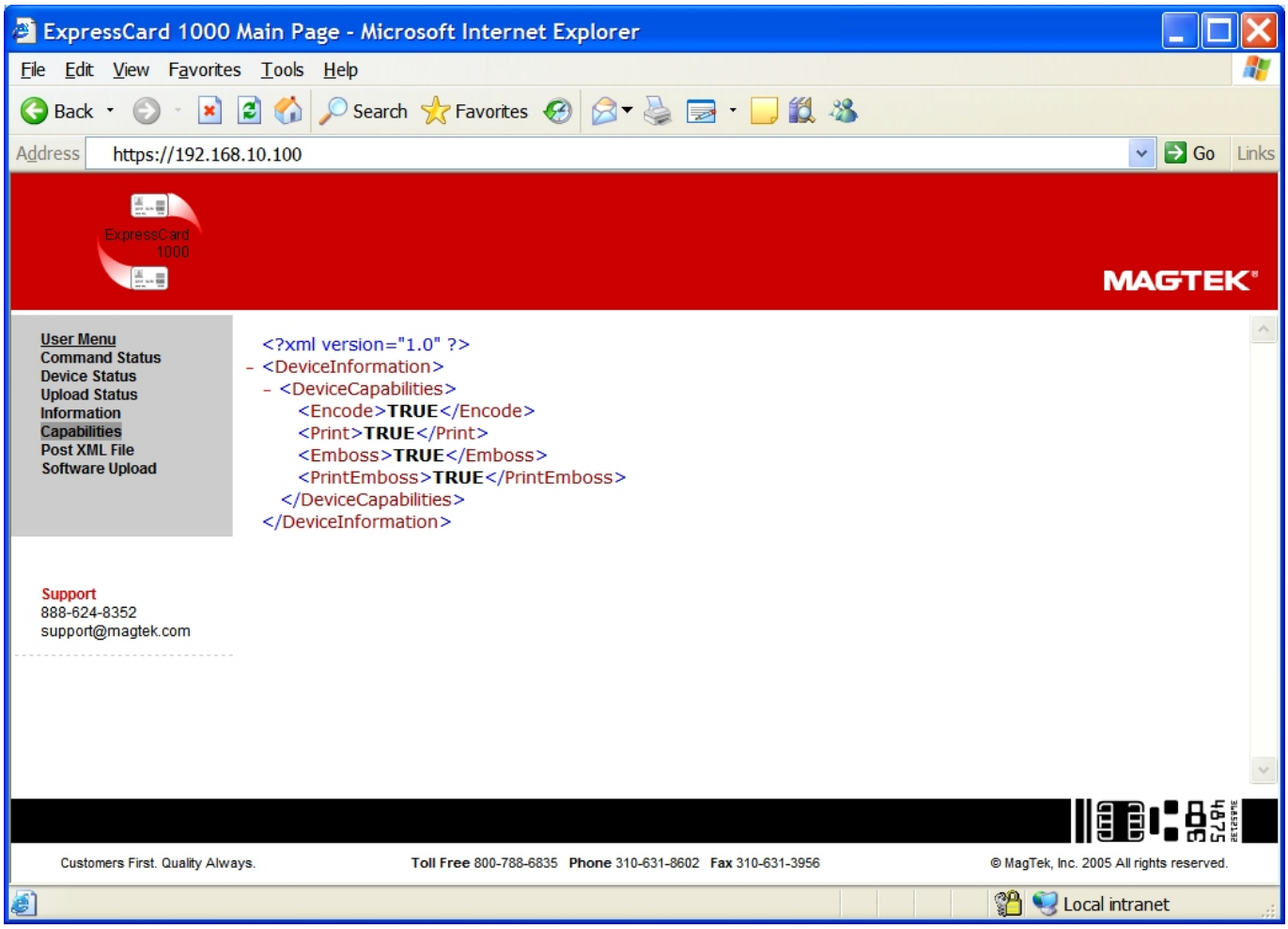
HOW TO COMMUNICATE WITH EXPRESSCARD 1000 USING WEB BROWSER

ExpressCard 1000 can be accessed from an Internet Web browser using the IP address of the device. An example in using the IP address of the ExpressCard 1000 device to obtain device capabilities information is followed.

Assuming the ExpressCard 1000 device has IP address 192.168.10.100, type the following line in the address box of the Web browser. Internet Explorer is used in this example:

https:// 192.168.10.100/Index.aspx?DeviceInformation=DeviceCapabilities

Press the Enter key, and the ExpressCard 1000 responds with the results shown below:

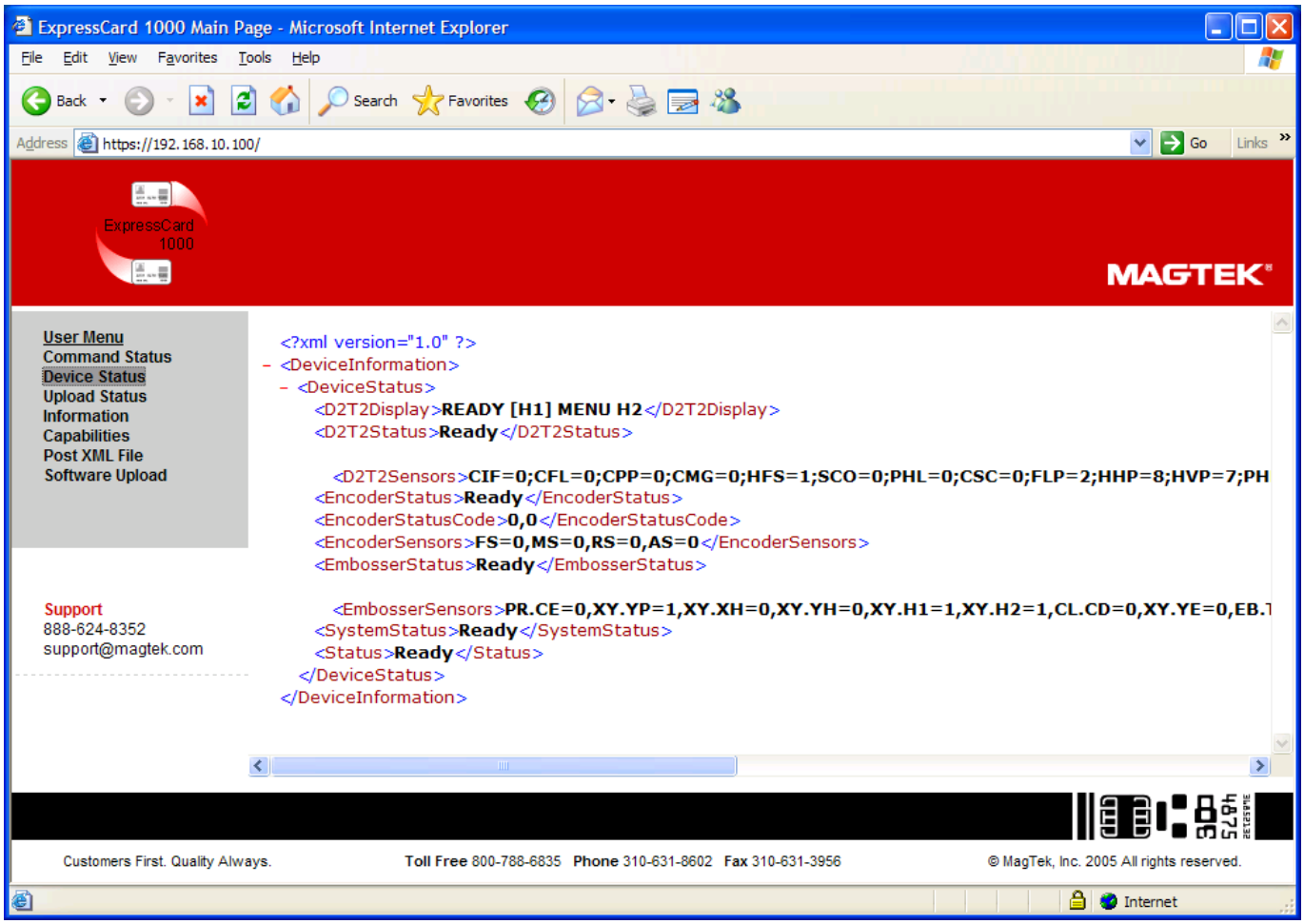


Get Device Status

The following is an example in using the IP address of the ExpressCard 1000 device to obtain device status information. Assuming the ExpressCard 1000 device has IP address 192.168.10.100, type the following line in the address box of the Web browser. Internet Explorer is used in this example:

https://192.168.10.100/Index.aspx?DeviceInformation=DeviceStatus

Press the Enter key, and the ExpressCard 1000 responds with the results shown below:

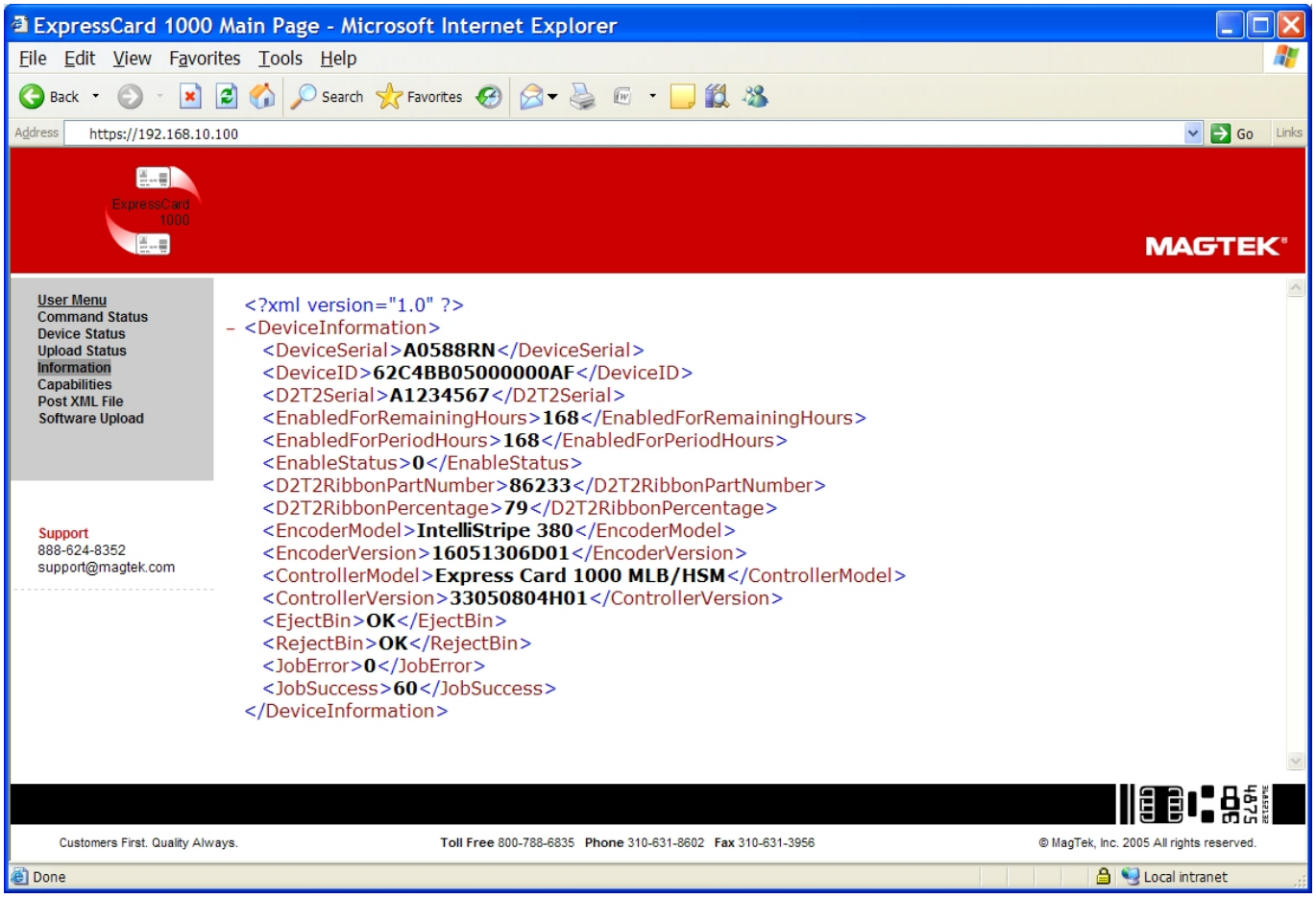


Get Device Information

The following is an example in using the IP address of the ExpressCard 1000 device to obtain device - usage information. Assuming the ExpressCard 1000 device has IP address 192.168.10.100, type the following line in the address box of the Web browser. Internet Explorer is used in this example:

https:// 192.168.10.100/Index.aspx?DeviceInformation=DeviceInformation

Press the Enter key, and the ExpressCard 1000 responds with the results shown below:



SOFTWARE FLOW FOR DOCUMENT PROCESSING

Figure 2-4 illustrates the sequence of document processing.

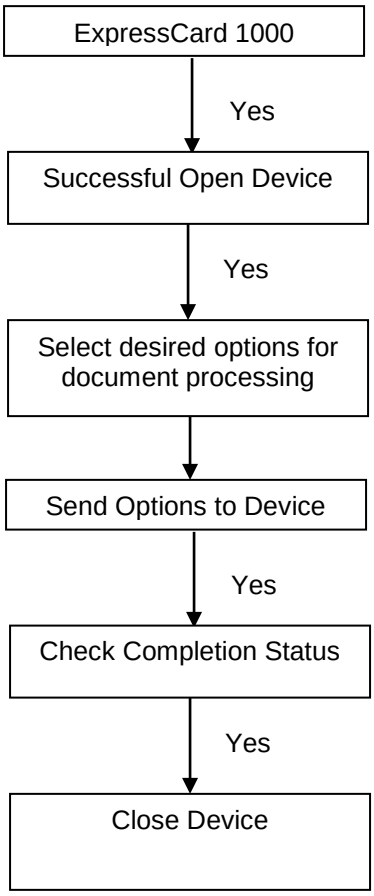


Figure 2-4. Software Flow for Card Processing

HOW TO PROCESS CARD USING EXPRESSCARD 1000 API

To process card, follow these steps:

Find the attached ExpressCard 1000 device by calling function **MTECSDK_GetDevice**.

Use function **MTECSDK_OpenDevice** to open the device.

ExpressCard 1000 is now ready to accept the XML options using function **MTECSDK_ProcessDoc**. For every document that is processed, a set of options must be sent to the device. These options define how the document will be processed, i.e., image file, text lines, magstripe encode data, emboss data, etc. The options are stored as key/value pairs in a buffer. The memory for this buffer must be allocated big enough to store all selected options. These options must be sent to ExpressCard 1000 device using function **MTECSDK_ProcessDoc**. Use function **MTECSDK_SetValue** to store the key/value pairs in the option buffer. The options can be set in the buffer only once.

Use function **MTECSDK_ProcessDoc** to send the process options to the ExpressCard 1000 device.

ExpressCard 1000 processes the document with the given process options and returns the result of the process operation. The result is stored in the third parameter of the function **MTECSDK_ProcessDoc**. The result contains only the information of the operation and information of the device. Use function **MTECSDK_GetValue** to retrieve key/value pair from document information.

ERROR REPORTING

When a document is processed, if there is an error in the ExpressCard 1000 device or in the processing operation, the error is returned in the function **MTECSDK_ProcessDoc**. Use function **MTECSDK_GetValue** to retrieve the error information. Appendix B has a list of return codes and return messages from the ExpressCard 1000.

If an error occurs due to the failure of API function, the error is returned as the return code of the API provided by **MTECSDK.dll.** Appendix B contains a list of return codes from **MTECSDK.dll**

DEBUGGING API

On the ExpressCard 1000 Demo GUI program, there is an option to enable the error logging function. If this option is enabled, all errors returned from the device are logged into log file named “Demo.log”. This file is resided in the ExpressCard 1000 Demo program installation directory.

SECTION 3. KEYS SENT TO DEVICE

Section 3 describes all the Key-Value pairs to be prepared by the application and sent to the device for card processing. The applicant can send information to the device using the following sections: *CommandInfo, D2T2, Emboss, MagEncode*, *SmartCard*.

In the tables below, in the Value column, the word *Value* represents a number. The actual value that can be set for *Value* is described in the Value Description column. For example, in the Key "Number", the Value Description is "1 <= *Value* <= 4"; that is, the Key "Number" has a *Value* of 1 or 2 or 3 or 4.

In the Value column, the term *string* represents a text string; for example, in the Key "PrintData", the Value Description is "text string"; that is the Key "PrintData" has a Value of a text string.

All Key-Value pairs belong to a group referred to as a Section. The following are top level sections sent during request:

Sections

- CommandInfo
- D2T2
- Emboss
- MagEncode
- SmartCard

This usage table describes which sections are required to perform a particular process. Sample code for each process is provided in Appendix A and Section 5 for Smart Card.

Usage Table for Sections

Operation	Required Sections
Print Image & Text	CommandInfo, D2T2
Emboss Data	CommandInfo, D2T2, Emboss
Encode Data	CommandInfo, MagEncode
Encode Print Emboss	CommandInfo, D2T2, Emboss, MagEncode
Smart Card	CommandInfo, SmartCard

SECTION = COMMANDINFO

The *CommandInfo* section includes keys that allow the application to specify the parameters for a specific command.

The following keys are included in the CommandInfo section:

- UniqueID
- CardHopperIndex
- CommandID
- Description
- SendCancelOnError

UniqueID

This key allows the host application to present a unique "job number" ID string that can be used to name (tag) a particular card personalization job sequence. This may be useful for auditing purposes to track whether a particular card print job was successful or not.

Values	Value Description
String "Card ID 00023654"	ID string can be any alpa-numeric string that the host application defines. This will be used as a "job tag" to track a given card personalization sequence.

Example XML Code: <UniqueID>Card ID 00023654</UniqueID>

CardHopperIndex

This key determines which card hopper will be used for a given card processing operation.

Values	Value Description
Num Value 0	Hopper 1
Num Value 1	Hopper 2
Num Value 2	Toggle Between Hopper 1 and Hopper 2
Num Value 3	Current Hopper
Num Value 4	Exception Feed Slot (Manual Card Insertion)

Example XML Code: <CardHopperIndex>1</CardHopperIndex>

CommandID

This key determines the request type that is being sent. This is optional if the card is being personalized in one request. For Smart Card personalization, a card is personalized using multiple commands starting with command 1.

Values	Value Description
Num Value 1	Move Card to Encoder Station
Num Value 2	Eject Card
Num Value 3	Abort Card
Num Value 4	Smart Card (Smart Card related commands)
Num Value 5	Encode a Card
Num Value 6	Read a Card
Num Value 7	Process a Card This command indicates that the personalization of a card will be done in one request (Default If Omitted)
Num Value 8	Unlatch Covers
Num Value 10	Move Card to Contactless Position
Num Value 11	Move Card to Consume Position
Num Value 14	Move Card to Encoder with Notification

Example XML Code: <CommandID>1</CommandID>

Description

This key accepts a description string which is to be displayed for the queueing version of the ExpressCard 1000 front panel application. This key allows the host application to provide conceptional information to describe a particular card personalization job sequence.

Values	Value Description
String "Gift Card"	Description string can be any alphanumeric string that the host application defines. Where size of string is 0 to 25 characters.
String ""	(Default If Omitted)

Example XML Code: <Description>Gift Card</Description>

SendCancelOnError

This key defines whether a Busy state will automatically be cleared if a card is not fed from the hopper to the encoder. When set to TRUE, a cancel is invoked to place the device into Ready state with a response of "Timeout Moving Card to Smart Card Station". When set to FALSE, the device will remain in its current state until further commands are sent.

Values	Value Description
String TRUE	Auto cancellation is enabled.
String FALSE	Auto cancellation is disabled. (Default If Omitted)

Example XML Code: <SendCancelOnError>TRUE</ SendCancelOnError >

SECTION = D2T2

The *D2T2* section includes keys that control various options for printing on the card. If the card is not printed or Embossed, this section can be optional.

The following keys are included in the D2T2 section:

- ColorManagement
- Dither
- Orientation
- PrintImageEnabled
- PrintTextEnabled
- RotateFront
- RotateBack
- SplitRibbon
- ImageDataEncodeType[n]
- ImageScaleXYWH[n]
- ImageSize[n]
- ImageSide[n]
- ImageType[n]
- NumImages
- ImageData[n]
- NumTextDataLines
- EncryptedText
- TextData[n]
- TextDataLength[n]
- TextDataEncoded[n]
- TextFontBold[n]
- TextFontColor[n]
- TextFontItalic[n]
- TextFontName[n]
- TextFontSize[n]
- TextFontStrikeThru[n]
- TextFontUnderline[n]
- TextPositionXY [n]
- TextSide[n]
- TextKeyLine[n]
- TextKeyLineColor[n]
- TextKeyLineTop[n]
- TextKeyLineLeft[n]
- TextKeyLineBottom[n]
- TextKeyLineRight[n]
- OVERLAY

ColorManagement

This key specifies the type of color management to be used.

Values	Value Description
String None	None
String Algebraic	Algebraic
String Monitor	Monitor
String System Color Management	System Color Management

Example XML Code: <ColorManagement>System Color Management</ColorManagement>

Dither

This key specifies the type of dithering effect to be used.

Values	Value Description
String Optimized For Graphics	Dithering for graphics
String Optimized For Photo	Dithering for photos

Example XML Code: <Dither>Optimized For Graphics</Dither>

Orientation

This key defines the orientation for a given image file.

Values	Value Description
String PORTRAIT	Image file will be oriented in PORTRAIT format
String LANDSCAPE	Image file will be oriented in LANDSCAPE format

Example XML Code: <Orientation>LANDSCAPE</Orientation>

PrintImageEnabled

This key defines the enable/disable of a given image to be printed.

Values	Value Description
String TRUE	Image file is enabled for printing
String FALSE	Image file is disabled for printing

Example XML Code: <PrintImageEnabled>TRUE</PrintImageEnabled>

PrintTextEnabled

This key defines the enable/disable of a given text data string to be printed.

Values	Value Description
String TRUE	Text data string is enabled for printing
String FALSE	Text data string is disabled for printing

Example XML Code: <PrintTextEnabled>TRUE</PrintTextEnabled>

RotateFront

This key defines the enable/disable of 180 degree rotation of given image/text print data on the front of the card.

Values	Value Description
String TRUE	Enable 180 rotation of given image/text data
String FALSE	Disable rotation of given image/text data

Example XML Code: <RotateFront>FALSE</RotateFront>

RotateBack

This key defines the enable/disable of 180 degree rotation of given image/text print data on the back of the card.

Values	Value Description
String TRUE	Enable 180 rotation of given image/text data
String FALSE	Disable rotation of given image/text data

Example XML Code: <RotateBack>FALSE</RotateBack>

SplitRibbon

This key defines if standard or split ribbons are being used.
”Split” references the condition where different types of ribbon panels are used on the front and back of the card.
”Standard” references the condition where the same types of ribbon panels are used on both the front and back sides of the card.

Values	Value Description
String TRUE	Split Ribbon is being used. (YMCKOK)
String FALSE	Standard Ribbon is being used (YMCKO)

Example XML Code: <SplitRibbon>FALSE</SplitRibbon>

ImageDataEncodeType[n]

This key determines the type of encode algorithm that will be utilized to pre-process/encode the print image data.

Values	Value Description
Num Value 64	Use Base-64 Encode algorithm to pre-process image data [n] = ID number for a given image box (0 < n < 100)

Example XML Code: <ImageDataEncodeType1>64</ImageDataEncodeType1>

ImageScaleXYWH[n]

This key defines the upper-left XY coordinate location of a given [n] image box, the Width and Height of a given image box, and the ID reference index for a given image box. Units of measurement are: 1/100th of an inch (0.010”).

Values	Value Description
NumValue X,Y,W,H[n]	Where: X, Y = Upper left corner of image box. (0 < X < 336), (0 < Y < 212) W, H = Width and Height dim of image box (0 < W < 336), (0 <H< 212) (**Note: If W, H are set to -1, -1, it “locks” the aspect ratio of the image**) [n] = ID number for a given image box (0 < n < 100)

Example XML Code: <ImageScaleXYWH1>0,0,336,212</ImageScaleXYWH1>

Physical Definition of CR-80 Card:

A standard CR-80 card has a nominal max length (X dimension) of 3.375" inches, and a nominal max height (Y dimension) of 2.127" inches. However, it should be noted that the card printer utilized in the ExpressCard 1000 cannot physically print directly to the edges of the card. As such, its operational boundaries on a CR-80 card are defined as: Length X = 3.362" and Y = 2.114". The unit of measurement for ImageScaleXYWH[n] function is 1/100th of an inch. The numeric operational ranges for X and Y are: 0 < X < 336 and 0 < Y < 212. The origin of the coordinate system (0,0) is the top-left corner of the card.

ImageSize[n]

This key defines a given image file in bytes.

Values	Value Description
NumValue X[n]	Size of image file in bytes. Where: X = Size of image file in bytes. (0 < X < no limit) [n] = ID number for a given image. (0 < n < 100)

Example XML Code: <ImageSize1>136405</ImageSize1>

ImageSide[n]

This key defines the side of the card upon which a given image will be placed.

Values	Value Description
String FRONT	Place given image on the Front side of the card
String BACK	Place given image on the Rear side of the card (Typically side with magstripe)
NumValue [n]	Where [n] = ID number for given image. (0 < n < 100)

Example XML Code: <ImageSide1>FRONT</ImageSide1>

ImageType[n]

This key defines the file format in which a given image will be presented. Typically, Bitmap (BMP), JPEG (JPG), or GIF formats.

Values	Value Description
String BMP	Image file is in BMP format
String JPG	Image file is in JPG format
String GIF	Image file is in GIF format
Num Value [n]	Where [n] = ID number for given image. (0 < n < 100)

Example XML Code: <ImageType1>JPG</ImageType1>

NumImages

This key defines the total number of images that are included in a print file: 0 = None. There is no theoretical limit on the number of image files that can be included. However, the image size and card size will dictate the maximum values that can be practically used. There is no boundary checking; thus, any values that exceed the physical boundaries of the card will be rotated out of the printer buffer.

Values	Value Description
NumValue X	Numeric value that represents the total number of images being included in a print file, where: 0 < X < 100) (May be omitted if PrintImageEnabled is set to FALSE)

Example XML Code: <NumImages>2</NumImages>

ImageData[n]

This key contains the encoded image to be printed on the card.

Values	Value Description
String ASCII	Image file encoded data. (Typically Base 64)
Num Value [n]	Where [n] = ID number for a given image. (0 < n < 100)

Example XML Code: <ImageData1>/9j/4AAQSkZJRgABAgEBLAE</ImageData1>

NumTextDataLines

This key defines the number of text data lines that are included in a print file: 0 = None. There is no theoretical limit on the number of text data lines that can be used. However, the font size and card size will dictate the maximum values that can be practically used. There is no boundary checking; thus, any values that exceed the physical boundaries of the card will be rotated out of the printer buffer.

Values	Value Description
NumValue X	Numeric value that represents the total number of text data lines images being included in a print file, where: 0 < X < 100) (May be omitted if PrintTextEnabled is set to FALSE)

Example XML Code: <NumTextDataLines>3</NumTextDataLines>

EncryptedText

This key defines the whether encryption is enabled/disabled for all text data strings to be printed. When set to TRUE, all textdata[n] lines within the D2T2 section are required to be encrypted.

Values	Value Description
String TRUE	Text data string is encrypted
String FALSE	Text data string is not encrypted

Example XML Code: <EncryptedText>TRUE</EncryptedText>

TextData[n]

This key defines the text data string and ID number that will be assigned to a given text line. There is no theoretical limit on the length of data that can be included on a given text line. However, the font size and card size will dictate the maximum length that can be practically used. There is no boundary checking; thus, any values that exceed the physical boundaries of the card will be rotated out of the printer buffer.

This key supports International characters when encoded in Base 64 format and the TextDataEncoded[n] key is set to TRUE.

Values	Value Description
String ASCII	ASCII data string to be printed.
String BASE64	Base 64 encoded data string to be printed.
NumValue [n]	Where [n] = ID number for given text line. (0 < n < 100)

Example XML Code: <TextData1>1234</TextData1>

TextDataLength[n]

This key defines the length of the clear text characters of a given text line before encryption. This key is only required for encrypted text data strings and when EncryptedText tag is set to TRUE.

Values	Value Description
NumValue X[n]	Size of clear text line in bytes. Where: X = Size of clear text line in bytes. (0 < X < no limit) [n] = ID number for a given image. (0 < n < 100)

Example XML Code:
Where clear text = "1234" and encrypted text = "bMfQMUtMKDY="

<TextData1>bMfQMUtMKDY=</TextData1>
<TextDataLength1>4</TextDataLength1>

TextDataEncoded[n]

This key defines whether the data string to be printed is encoded in Base 64 format. International characters may be printed as long as the supporting font is installed on both the client side and the target ExpressCard 1000 machine. Base 64 encoding ensures that the data string is preserved during transport to the machine.

Values	Value Description
String TRUE	Enable Base 64 decoding of TextData[n].
String FALSE	Disable Base 64 decoding of TextData[n]. (Default If Omitted)
NumValue [n]	Where [n] = ID number for given text line. (0 < n < 100)

Example XML Code:

ASCII <TextData1>Employee ID # 123456</TextData1>
<TextDataEncoded1>FALSE</TextDataEncoded1>

BASE64 <TextData1>RW1wbG95ZWUgSUQgIyAxMjM0NTY=</TextData1>
<TextDataEncoded1>TRUE</TextDataEncoded1>

TextFontBold[n]

This key defines if BOLD attributes will be applied to the font of a given text line.

Values	Value Description
String TRUE	BOLD attributes will be applied to given text line
String FALSE	BOLD attributes will not be applied to a given text line
NumValue [n]	Where [n] = ID number for given text line. (0 < n < 100)

Example XML Code: <TextFontBold1>TRUE</TextFontBold1>

TextFontColor[n]

This key defines the color attributes that will be applied to the font of a given text line.

Values	Value Description
NumValue 000000 - FFFFFF	MS Windows color range
NumValue [n]	Where [n] = ID number for given text line. (0 < n < 100)

Example XML Code: <TextFontColor1>FF0000</TextFontColor1>

TextFontItalic[n]

This key defines if *italic* attributes will be applied to the font of a given text line.

Values	Value Description
String TRUE	*Italic* attributes will be applied to given text line.
String FALSE	*Italic* attributes will not be applied to given text line.
NumValue [n]	Where [n] = ID number for given text line. (0 < n < 100)

Example XML Code: <TextFontItalic1>TRUE</TextFontItalic1>

TextFontName[n]

This key defines the type of font that will be applied to a given text line. Any type of font styles supported by MS Windows OS can be utilized. However, these font types must be loaded into the OS prior to selection.

Values	Value Description
String MS Sans Serif	Any type of font supported by MS Windows OS. (e.g. Courier, Arial, etc..)
NumValue [n]	Where [n] = ID number for given text line. (0 < n < 100)

Example XML Code: <TextFontName1>MS Sans Serif</TextFontName1>

TextFontSize[n]

This key defines the font size that will be applied to a given text line. There is no theoretical limit to the size of the font that can be used. However, the physical size of the card will dictate the practical largest font size that can be used. There is no boundary checking; thus, any values that exceed the physical boundaries of the card will be rotated out of the printer buffer.

Values	Value Description
NumValue X	Where X is an integer value that represents the font size referenced in MS Windows OS.
NumValue [n]	Where [n] = ID number for given text line. (0 < n < 100)

Example XML Code: <TextFontSize1>MS Sans Serif</TextFontSize1>

TextFontStrikeThru[n]

This key defines if the strikethrough (~~strikethrough~~) attribute will be applied to the font of a given text line.

Values	Value Description
String TRUE	Strikethrough (~~strikethrough~~) attributes will be applied to given text line
String FALSE	Strikethrough (~~strikethrough~~) attributes will not be applied to given text line
NumValue [n]	Where [n] = ID number for given text line. (0 < n < 100)

Example XML Code: <TextFontStrikeThru1>TRUE</TextFontStrikeThru1>

TextFontUnderline[n]

This key defines if the underline (<u>underline</u>) attribute will be applied to the font of a given text line.

Values	Value Description
String TRUE	Underline (<u>underline</u>) attributes will be applied to given text line
String FALSE	Underline (<u>underline</u>) attributes will not be applied to given text line
NumValue [n]	Where [n] = ID number for given text line. (0 < n < 100)

Example XML Code: <TextFontUnderline1>TRUE</TextFontUnderline1>

TextPositionXY[n]

This key defines the X,Y coordinate position for a text line to be placed on the card. Units of measurement are: 1/100th of an inch (0.010”).

Physical Definition of CR-80 Card:

A standard CR-80 card has a nominal max length (X dimension) of 3.375” inches, and a nominal max height (Y dimension) of 2.127” inches. However, it should be noted that the card printer utilized in the ExpressCard 1000 cannot physically print directly to the edges of the card. As such its operational boundaries on a CR-80 card are defined as: Length X = 3.362” and Y = 2.114”. The unit of measurement for TextPositionXY[n] function is 1/100th of an inch. The numeric operational ranges for X and Y are: 0 < X < 336 and 0 < Y < 212. The origin of the coordinate system (0,0) is the top-left corner of the card.

.

Values	Value Description
NumValue X,Y[n]	Where: X ,Y = Upper left corner of text line. (0 < X < 336), (0 < Y < 212) [n] = ID number for a given image box (0 < n < 100)

Example XML Code: <TextPositionXY1>70,147</TextPositionXY1>

TextSide[n]

This key defines the side of the card on which a given font will be placed for a given text line.

Values	Value Description
String FRONT	Place given text line on the Front side of the card.
String BACK	Place given text line on the Rear side of the card. (Typically side with magstripe)
NumValue [n]	Where [n] = ID number for given text line. (0 < n < 100)

Example XML Code: <TextSide1>FRONT</TextSide1>

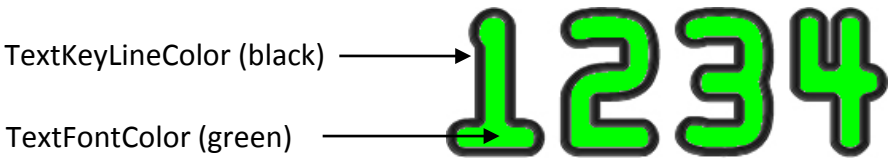
TextKeyLine[n]

This key defines whether a keyline effect will be applied to a given text data line.
TextFontColor[n] is the color in the foreground of the keyline. TextKeyLineColor[n] is the background or border color of the keyline.

Keylining enhances the distinguishment between the printed text and any background image by creating a boundary line around each individual character of a text line. The boundary line is controlled by its thickness and color attributes. This boundary color should be different from the character color to have an appreciable effect.

Values	Value Description
String TRUE	Keyline data string is enabled for printing.
String FALSE	Keyline data string is disabled for printing.
NumValue [n]	Where [n] = ID number for given text line. (0 < n < 100)

Example XML Code: <TextKeyLine1>TRUE</TextKeyLine1>



TextKeyLineColor[n]

This key defines the color attributes that will be applied to the keyline on a given text line.
The color attribute should be different from TextFontColor[n] to have an appreciable effect.

Values	Value Description
NumValue 000000 - FFFFFF	MS Windows color range (RGB)
NumValue [n]	Where [n] = ID number for given text line. (0 < n < 100)

Example XML Code: <TextKeyLineColor1>000000</TextKeyLineColor1>

TextKeyLineTop[n]

This key defines the thickness of the keyline applied to the top of each character on a given text line.

Values	Value Description
NumValue -X	Numeric value that represents the upper thickness. Where: (-5 <= X <= 0) in increments of 0.1 and 0 takes no effect
NumValue [n]	Where [n] = ID number for given text line. (0 < n < 100)

Example XML Code: <TextKeyLineTop1>-1.0</TextKeyLineTop1>

TextKeyLineLeft[n]

This key defines the thickness of the keyline applied to the left of each character on a given text line.

Values	Value Description
NumValue -X	Numeric value that represents the left thickness. Where: (-5 <= X <= 0) in increments of 0.1 and 0 takes no effect
NumValue [n]	Where [n] = ID number for given text line. (0 < n < 100)

Example XML Code: <TextKeyLineLeft1>-1.0</TextKeyLineLeft1>

TextKeyLineBottom[n]

This key defines the thickness of the keyline applied to the bottom of each character on a given text line.

Values	Value Description
NumValue X	Numeric value that represents the bottom thickness. Where: (0 <= X <= 5) in increments of 0.1 and 0 takes no effect
NumValue [n]	Where [n] = ID number for given text line. (0 < n < 100)

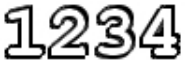
Example XML Code: <TextKeyLineBottom1>1.0</TextKeyLineBottom1>

TextKeyLineRight[n]

This key defines the thickness of the keyline applied to the right of each character on a given text line.

Values	Value Description
NumValue X	Numeric value that represents the right thickness. Where: (0 <= X <= 5) in increments of 0.1 and 0 takes no effect
NumValue [n]	Where [n] = ID number for given text line. (0 < n < 100)

Example XML Code: <TextKeyLineRight1>1.0</TextKeyLineRight1>

Border around text: 

```
<TextData1>1234</TextData1>
<TextPositionXY1>10,10</TextPositionXY1>
<TextSide1>FRONT</TextSide1>
<TextFontName1>Courier New</TextFontName1>
<TextFontSize1>18</TextFontSize1>
<TextFontColor1>FFFFFF</TextFontColor1>
<TextFontBold1>TRUE</TextFontBold1>
<TextFontItalic1>TRUE</TextFontItalic1>
<TextFontStrikeThru1>FALSE</TextFontStrikeThru1>
<TextKeyLine1>TRUE</TextKeyLine1>
<TextKeyLineColor1>000000</TextKeyLineColor1>
<TextKeyLineTop1>-1.0</TextKeyLineTop1>
<TextKeyLineLeft1>-1.0</TextKeyLineLeft1>
<TextKeyLineBottom1>1.0</TextKeyLineBottom1>
<TextKeyLineRight1>1.0</TextKeyLineRight1>
```

Shadow around text: 

```
<TextData1>1234</TextData1>
<TextPositionXY1>10,10</TextPositionXY1>
<TextSide1>FRONT</TextSide1>
<TextFontName1>Courier New</TextFontName1>
<TextFontSize1>18</TextFontSize1>
<TextFontColor1>000000</TextFontColor1>
<TextFontBold1>TRUE</TextFontBold1>
<TextFontItalic1>TRUE</TextFontItalic1>
<TextFontStrikeThru1>FALSE</TextFontStrikeThru1>
<TextKeyLine1>TRUE</TextKeyLine1>
<TextKeyLineColor1>646464</TextKeyLineColor1>
<TextKeyLineTop1>0</TextKeyLineTop1>
<TextKeyLineLeft1>0</TextKeyLineLeft1>
<TextKeyLineBottom1>2.0</TextKeyLineBottom1>
<TextKeyLineRight1>2.0</TextKeyLineRight1>
```

SECTION = OVERLAY

The *OVERLAY* section includes keys that control how the Overlay (O) Panel and/or the print area will appear on a card. This option is helpful if it is necessary to omit the overlay or printing around a card's smart chip, magnetic stripe, or user defined area.

The OVERLAY section is a subkey of the D2T2 section. OVERLAY is optional and temporary. Any card request that includes an OVERLAY section overrides the previous OVERLAY and default settings for the machine. Any card request that does not include an OVERLAY section will use the default settings for the machine.

The following keys are included in the OVERLAY section:

- Front
- Back
- Area

The following attributes belong to keys of the OVERLAY section:

- Units
- PrintArea
- PrintAreaType

Units

This attribute determines the unit of measurement to be used. The Units attribute is contained directly inside the OVERLAY key.

Values	Value Description
String inches	Apply measurements in inches.
String mm	Apply measurements in millimeters.

Example XML Code:

```
<D2T2>
   <OVERLAY   Units=”inches”></OVERLAY>
</D2T2>
```

Front

This key defines the effects which are to be applied to the front of the card.

Values	Value Description
Attribute PrintArea=""	Attribute to describe the effect to be applied to a print area.
Attribute PrintAreaType=""	Attribute to determine which phase of printing to apply the print area effect.

Example XML Code:

```
<OVERLAY Units="inches">
    <Front   PrintArea="Omit Smart Chip Area"   PrintAreaType="For Print and Overlay">
    </Front>
</OVERLAY>
```

Back

This key defines the effects which are to be applied to the back of the card.

Values	Value Description
Attribute PrintArea=""	Attribute to describe the effect to be applied to a print area.
Attribute PrintAreaType=""	Attribute to determine which phase of printing to apply the print area effect.

Example XML Code:

```
<OVERLAY Units="inches">
    <Back   PrintArea="Omit Mag Stripe Chip Area"   PrintAreaType="For Print and Overlay">
    </Back>
</OVERLAY>
```

PrintArea

This attribute describes the effect to be applied to a print area. The PrintArea attribute is contained directly inside either the Front or Back key.

Values	Value Description
String Full Card	Overlay and/or print the entire card. When specified the area is pre-defined. <Area> keys are ignored.
String Defined Area(s)	Overlay and/or print only in the defined area or areas.
String Undefined Area(s)	Overlay and/or print only in the space outside the defined area.
String Omit Smart Chip Area	Overlay and/or print only in the space outside the standard location of a smart chip. When specified the area is pre-defined. <Area> keys are ignored.
String Omit Mag Stripe Area	Overlay and/or print only in the space outside the standard location of an ISO Magnetic Stripe. When specified the area is pre-defined. <Area> keys are ignored.
String Omit Signature Area	Overlay and/or print only in the space outside the standard location of a signature panel. When specified the area is pre-defined. <Area> keys are ignored.

Example XML Code:

```
<OVERLAY Units="inches">
   <Back   PrintArea="Omit Mag Stripe Chip Area"   PrintAreaType="For Print and Overlay">
   </Back>
</OVERLAY>
```

PrintAreaType

This attribute determines which phase of printing to apply the print area effect. The PrintAreaType attribute is contained directly inside either the Front or Back key.

Values	Value Description
String For Print and Overlay	Apply to both the printing and overlay process.
String For Overlay Only	Apply only to the overlay process. Printing will still be allowed over the entire card.
String For Print Only (No Overlay)	Apply only to the print process. In this mode, the overlay is completely disabled and will not be applied.

Example XML Code:

```
<OVERLAY Units="inches">
   <Back   PrintArea="Omit Mag Stripe Chip Area"   PrintAreaType="For Print and Overlay">
   </Back>
</OVERLAY>
```

Area

This key defines the position to apply a print area effect. Area is a subkey belonging to either the Front or Back key. Up to 5 areas may be defined for both the Front and Back of card.

Values	Value Description
Attribute id=”[n]”	Identifies an area to which the effect will be applied. Where[n]: (1 <= n <= 5) Up to 5 areas may be defined for both the Front and Back of card.
Attribute Width=”[w]”	Width of the area where [w]: (.2”/5mm <= w <= card width) For inches express in 0.000 (4 significant figures) For mm express in 00.0 (3 significant figures)
Attribute Height=”[h]”	Height of the area where [h]: (.2”/5mm <= h <= card height) For inches express in 0.000 (4 significant figures) For mm express in 00.0 (3 significant figures)
Attribute XPosition=”[x]”	X coordinate for bottom left corner of defined area box. Where [x]: (0 <= x <= card width) For inches express in 0.000 (4 significant figures) For mm express in 00.0 (3 significant figures)
Attribute YPosition=”[y]”	Y coordinate for bottom left corner of defined area box. Where[y]: (0 <= y <= card height) For inches express in 0.000 (4 significant figures) For mm express in 00.0 (3 significant figures)

Example XML Code:

```
<OVERLAY Units="inch">
   <Back   PrintArea="Undefined Area(s)"   PrintAreaType="For Print and Overlay" />
      <Area   id="1" Width="0.500" Height="0.500" XPosition="0.100" YPosition="0.100" />
   </Back>
 </OVERLAY>
```

SECTION = EMBOSS

The *Emboss* section includes keys that control various options to mechanically emboss and rear-indent characters on the card surface. Some sections of D2T2 are required to emboss a card. They are SplitRibbon, Dither, ColorManagement, RotateFront,RotateBack, Orientation, PrintTextEnabled and PrintImageEnabled.

Typically, embossed characters (raised characters) are used to represent the customer name, account number, and expiration date on the front surface of a financial card per ISO 7810 guidelines. Rear-indent (indented characters) is used to physically embed the account number into the signature panel of a financial card per ISO 7810 guidelines.

In the key definitions below, the term “emboss” will be used to represent both emboss and rear-indent functions.

The following keys are included in the Emboss section:

- Encrypted
- EmbossData[n]
- EmbossDataLength[n]
- EmbossDataLines
- EmbossDataPOSXY[n]
- EmbossEnabled
- EmbossFontID[n]
- EmbossFontColor[n]
- PrintEmboss[n]
- EmbossKeyLine[n]
- EmbossKeyLineColor[n]
- EmbossKeyLineTop[n]
- EmbossKeyLineLeft[n]
- EmbossKeyLineBottom[n]
- EmbossKeyLineRight[n]

Encrypted

This key defines the whether encryption is enabled/disabled for all emboss data lines. When set to TRUE all EmbossData[n] lines within the Emboss section are required to be encrypted.

Values	Value Description
String TRUE	Emboss data string is encrypted.
String FALSE	Emboss data string is not encrypted.

Example XML Code: <Encrypted>TRUE</Encrypted>

EmbossData[n]

This key defines the emboss data string and ID number that will be assigned to a given emboss line.

There is boundary checking to ensure that the length of a given emboss line (and its associated physical font) cannot exceed the physical boundaries of the card size.

Values	Value Description
String ASCII	ASCII data string to be embossed. (string length = 0 < len < Boundary)
NumValue [n]	Where [n] = ID number for given emboss line. (0 < n < 100)

Example XML Code: <EmbossData1>1234</EmbossData1>

EmbossDataLength[n]

This key defines the length of the clear text emboss characters of a given emboss line. This key is only required for encrypted emboss data strings and when Encrypted tag is set to TRUE.

Values	Value Description
NumValue X	Number of clear text emboss characters.
NumValue [n]	[n] = ID number for a given image. (0 < n < 100)

Example XML Code:
Where clear text = “1234” and encrypted text = “bMfQMUtMKDY=”

<EmbossData1>bMfQMUtMKDY=</EmbossData1>
<EmbossDataLength1>4</EmbossDataLength1>

EmbossDataLines

This key defines the number of emboss data lines that will be included in a job file: 0 = None. There is boundary checking to ensure that the number of emboss lines (and their associated physical fonts) cannot exceed the physical boundaries of the card size.

Values	Value Description
NumValue X	Numeric value that represents the total number of emboss data lines being included in a job file, where: 0 < X < Boundary)

Example XML Code: <EmbossDataLines>3</EmbossDataLines>

EmbossDataPosXY[n]

This key defines the X,Y coordinate position for an emboss line to be placed on the card. Units of measurement are: 1/100th of an inch (0.010")

Physical Definition of CR-80 Card:
A standard CR-80 card has a nominal max length (X dimension) of 3.375" inches, and a nominal max height (Y dimension) of 2.127" inches. However, it should be noted that the card embosser utilized in the ExpressCard 1000 cannot physically emboss directly to the edges of the card. As such its operational boundaries on a CR-80 card are defined as: Length X = 3.362" and Y = 2.114". The unit of measurement for EmbossDataPosXY[n] function is 1/100th of an inch. The numeric operational ranges for X and Y are: 0 < X < 336 and 0 < Y < 212. The origin of the coordinate system (0,0) is the top-left corner of the card.

Values	Value Description
NumValue X,Y[n]	Where: X ,Y = Upper left corner of text line. (0 < X < 336), (0 < Y < 212) [n] = ID number for a given image box (0 < n < 100)

Example XML Code: <EmbossDataPosXY1>30,197</EmbossDataPosXY1>

EmbossEnabled

This key defines the enable/disable of a given emboss data string to be printed.

Values	Value Description
String TRUE	Emboss data string is enabled for printing
String FALSE	Emboss data string is disabled for printing

Example XML Code: <EmbossEnabled>TRUE</EmbossEnabled>

EmbossFontID[n]

This key defines the type of physical (mechanical) font that will be associated with a given emboss line.

Typically, there are four types of mechanical fonts used with card embossing and rear-indent operations:

Farrington 7B – Used to print account number on front of financial cards.
Gothic – Used to print cardholder name and expiration date on front of financial cards.
Front Indent – Used to print BIN (Bank ID Number) on front of financial cards.
Rear Indent – Used to print the account number on the signature panel. Rear of financial cards.

Values	Value Description
NumValue 1 - 4	Where: 1 = Farrington 7B emboss font 2 = Gothic emboss font 3 = Front indent font 4 = Rear indent font
NumValue [n]	Where [n] = ID number for given emboss line. (0 < n < 100)

Example XML Code: <EmbossFont1>1</EmbossFont1>

EmbossFontColor[n]

This key defines the color attributes that will be applied to a given emboss line.

Values	Value Description
NumValue 000000 - FFFFFF	MS Windows color range
NumValue [n]	Where [n] = ID number for given text line. (0 < n < 100)

Example XML Code: <EmbossFontColor1>000000</EmbossFontColor1>

PrintEmboss[n]

This key defines the enable/disable of whether colorized print data will be placed on the card by the thermal printer (prior to the emboss process). This key is typically "TRUE" when the ExpressCard 1000 configuration does not include a dedicated foil hot-stamp tipping station.

Values	Value Description
String TRUE	Enable thermal printing of given emboss data.
String FALSE	Disable thermal printing of given emboss data.
NumValue [n]	Where [n] = ID number for given text line. (0 < n < 100)

Example XML Code: <PrintEmboss1>TRUE</PrintEmboss1>

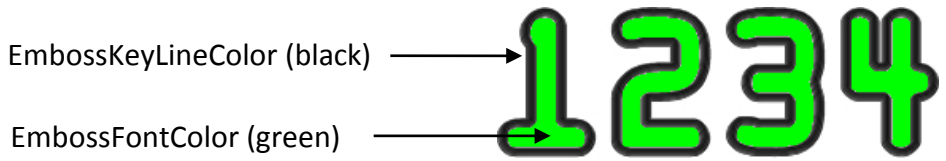
EmbossKeyLine[n]

This key defines the whether a keyline effect will be applied to a given emboss data line. EmbossFontColor[n] is the color in the foreground of the keyline. EmbossKeyLineColor[n] is the background or border color of the keyline.

Keylining enhances the distinguishment between the emboss text and any background image by creating a boundary line around each individual character of an emboss line. The boundary line is controlled by its thickness and color attributes. This boundary color should be different from the character color to have an appreciable effect.

Values	Value Description
String TRUE	Keyline data string is enabled for embossing.
String FALSE	Keyline data string is disabled for embossing.
NumValue [n]	Where [n] = ID number for given text line. (0 < n < 100)

Example XML Code: <EmbossKeyLine1>TRUE</EmbossKeyLine1>



EmbossKeyLineColor[n]

This key defines the color attributes that will be applied to the keyline on a given emboss line. The color attribute should be different from EmbossFontColor[n] to have an appreciable effect.

Values	Value Description
NumValue 000000 - FFFFFF	MS Windows color range
NumValue [n]	Where [n] = ID number for given text line. (0 < n < 100)

Example XML Code: <EmbossKeyLineColor1>000000</EmbossKeyLineColor1>

EmbossKeyLineTop[n]

This key defines the thickness of the keyline applied to the top of each character on a given emboss line.

Values	Value Description
NumValue -X	Numeric value that represents the upper thichness. Where: (-5 <= X <= 0) in increments of 0.1 and 0 takes no effect
NumValue [n]	Where [n] = ID number for given text line. (0 < n < 100)

Example XML Code: <EmbossKeyLineTop1>-1.0</EmbossKeyLineTop1>

EmbossKeyLineLeft[n]

This key defines the thickness of the keyline applied to the left of each character on a given emboss line.

Values	Value Description
NumValue -X	Numeric value that represents the left thickness. Where: (-5 <= X <= 0) in increments of 0.1 and 0 takes no effect
NumValue [n]	Where [n] = ID number for given text line. (0 < n < 100)

Example XML Code: <EmbossKeyLineLeft1>-1.0</EmbossKeyLineLeft1>

EmbossKeyLineBottom[n]

This key defines the thickness of the keyline applied to the bottom of each character on a given emboss line.

Values	Value Description
NumValue X	Numeric value that represents the bottom thickness. Where: (0 <= X <= 5) in increments of 0.1 and 0 takes no effect
NumValue [n]	Where [n] = ID number for given text line. (0 < n < 100)

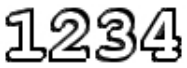
Example XML Code: <EmbossKeyLineBottom1>1.0</EmbossKeyLineBottom1>

EmbossKeyLineRight[n]

This key defines the thickness of the keyline applied to the right of each character on a given emboss line.

Values	Value Description
NumValue X	Numeric value that represents the right thickness. Where: (0 <= X <= 5) in increments of 0.1 and 0 takes no effect
NumValue [n]	Where [n] = ID number for given text line. (0 < n < 100)

Example XML Code: <EmbossKeyLineRight1>TRUE</EmbossKeyLineRight1>

Border around text: 

```
<EmbossDataLines>1</EmbossDataLines>
<PrintEmboss1>TRUE</PrintEmboss1>
<EmbossData1>1234</EmbossData1>
<EmbossDataPosXY1>40,120</EmbossDataPosXY1>
<EmbossFontID1>1</EmbossFontID1>
<EmbossFontColor1>FFFFFF</EmbossFontColor1>
<EmbossKeyLine1>TRUE</EmbossKeyLine1>
<EmbossKeyLineColor1>000000</EmbossKeyLineColor1>
<EmbossKeyLineTop1>-1.0</EmbossKeyLineTop1>
<EmbossKeyLineLeft1>-1.0</EmbossKeyLineLeft1>
<EmbossKeyLineBottom1>1.0</EmbossKeyLineBottom1>
<EmbossKeyLineRight1>1.0</EmbossKeyLineRight1>
```

Shadow under text: 

```
<EmbossDataLines>1</EmbossDataLines>
<PrintEmboss1>TRUE</PrintEmboss1>
<EmbossData1>1234</EmbossData1>
<EmbossDataPosXY1>40,120</EmbossDataPosXY1>
<EmbossFontID1>1</EmbossFontID1>
<EmbossFontColor1>000000</EmbossFontColor1>
<EmbossKeyLine1>TRUE</EmbossKeyLine1>
<EmbossKeyLineColor1>646464</EmbossKeyLineColor1>
<EmbossKeyLineTop1>0</EmbossKeyLineTop1>
<EmbossKeyLineLeft1>0</EmbossKeyLineLeft1>
<EmbossKeyLineBottom1>2.0</EmbossKeyLineBottom1>
<EmbossKeyLineRight1>2.0</EmbossKeyLineRight1>
```

SECTION = MAGENCODE

The *MagEncode* section includes keys that control various operations associated with encoding magstripe data to a card.

The following keys are included in the MagEncode section:

- Coercivity
- MagEncodeEnable
- MagReadEnable
- NumTracks
- StripeRetry
- TrackBPI[n]
- TrackBPC[n]
- Encrypted
- TrackData[n]
- TrackDataLength[n]
- VerifyBeforeEncode
- VerificationData

Coercivity

This key defines whether the magnetic encoder will utilize low-coercivity (LoCo), high-coercivity (HiCo), or Auto-Coercivity energy levels to encode magnetic data onto the magstripe.

In general, LoCo energy is used on magstripe cards that are approximately rated at 300 Oersteds of coercivity; HiCo energy is used on magstripe cards that are rated at 2750 Oersteds or higher coercivity.

If it is known what type of magstripe card stock is being utilized, the application program can directly choose LoCo or HiCo mode to apply the correct encoding energy to the cards. However, if the application does not know what type of magstripe is being used, then use of the Auto-Coercivity feature is recommended. The only penalty for using this mode is an approximate 2 to 3 second delay per card encode operation, as the device “tests” the coercivity level of the magstripe before encoding.

Values	Value Description
String LOW	Use LoCo energy level for encoding. (300 Oe)
String HIGH	Use HiCo energy level for encoding. (2750 – 4000 Oe)
String AUTO	Allow encoder to automatically detect if LoCo or HiCo should be used on a particular card.

Example XML Code: <Coercivity>AUTO</Coercivity>

MagEncodeEnable

This key defines the enable/disable of magstripe encode operations.

Values	Value Description
String TRUE	Magnetic encoder is enabled for encode operations.
String FALSE	Magnetic encoder is disabled for encode operations.

Example XML Code: <MagEncodeEnable>TRUE</MagEncodeEnable>

MagReadEnable

This key defines whether magstripe data is returned in an XML response after encode operations.

Values	Value Description
String TRUE	Magstripe data is returned after encode operations. (Default If Omitted)
String FALSE	Magstripe data is not returned after encode operations.

Example XML Code: <MagReadEnable>TRUE</MagReadEnable>

NumTracks

This key is reserved for future use. Currently, its default value is permanently set to “3”.

Values	Value Description
NumValue 3	Magnetic encoder enabled to operate on ISO tracks 1, 2, 3

Example XML Code: <NumTracks>3</NumTracks>

StripeRetry

This key defines the maximum number of times that the magnetic encoder will attempt to read-verify that data has been properly encoded onto a magstripe. It is recommended that this value not exceed 3.

Values	Value Description
NumValue 0 – 3	Number of read-verify attempts. (3 is recommended)

Example XML Code: <StripeRetry>3</StripeRetry>

TrackBPI[n]

This key defines the data-density in "bits per inch" (bpi) that should be utilized on a given data track on the magstripe.

Per ISO 7810 guidelines, either 75 bpi or 210 bpi should be utilized on data tracks. Typical data densities are:

Track 1 = 210 bpi, Track 2 = 75 bpi, Track 3 = 210 bpi

Values	Value Description
NumValue 75 or 210	Where: 75 = 75 bpi (Default If Omitted for Track 2) 210 = 210 bpi (Default If Omitted for Track 1, Track 3)
NumValue [n]	[n] Designates which data track of the magstripe is being referenced (Track 1, Track 2, Track 3)

Example XML Code: <TrackBPI2>75</TrackBPI2> (Means: Set Track 2 to 75 bpi)
Example XML Code: <TrackBPI1>210</TrackBPI1> (Means: Set Track 1 to 210 bpi)

TrackBPC[n]

This key defines the "bits per character" (bpc) that should be utilized on a given data track on the magstripe.

Per ISO 7810 guidelines, either 5 bpc or 7 bpc should be utilized for data bytes.
Typical values are:
Track 1 = 7 bpc, Track 2 = 5 bpc, Track 3 = 5 bpc

Values	Value Description
NumValue 7 or 5	Where: 5 = 5 bits per character (Typically used on Track 2) (Default If Omitted for Track 2) 7 = 7 bits per character (Typically used on Track 1 and Track 3) (Default If Omitted for Track 1 and Track 3)
NumValue [n]	[n] Designates which data track of the magstripe is being referenced (Track 1, Track 2, Track 3)

Example XML Code: <TrackBPI1>7</TrackBPC1> (Means: Set Track 1 to 7 bpc)
Example XML Code: <TrackBPI2>5</TrackBPC2> (Means: Set Track 2 to 5 bpc)

Encrypted

This key defines whether encryption is enabled/disabled for all track data lines. When set to TRUE all TrackData[n] lines within the MagEncode section are required to be encrypted.

Values	Value Description
String TRUE	Track data string is encrypted.
String FALSE	Track data string is not encrypted.

Example XML Code: <Encrypted>TRUE</Encrypted>

TrackData[n]

This key defines the data strings that are to be encoded on a given data track.

Per ISO 7810 guidelines, the maximum data lengths and data types for ISO track data are as follows:

Track 1 = 79 characters of alphanumeric data (210bpi, 7 bpc)
Track 2 = 40 characters numeric-only data (75 bpi, 5 bpc)
Track 3 = 107 bytes of numeric-only data (210 bpi, 5 bpc)

Start-sentinel, End-sentinel, and LRC values are included in the max data lengths shown above. However, these characters do not need to be included in track data strings submitted to the TrackData[n] function. (These are automatically calculated and included by the TrackData[n] function.)

Values	Value Description
String	Where: (Per ISO 7810) Track 1 Data = 76 characters. 7-bit alphanumeric Track 2 Data = 37 characters. 5-bit numeric Track 3 Data = 104 characters. 5 bit numeric
NumValue [n]	[n] Designates which data track of the magstripe is being referenced (Track 1, Track 2, Track 3)

Example XML Code: <TrackData1>B4024113466452897^Robert Jones</TrackData1>
Example XML Code: <TrackData2>4024113466452897=44332</TrackData2>

TrackData[n]Length

This key defines the length of the clear text track encode characters of a given track. This key is only required for encrypted track data strings and when Encrypted tag is set to TRUE.

Values	Value Description
NumValue X	Number of clear text track characters.
NumValue [n]	[n] = ID number for a given image. (0 < n < 100)

Example XML Code:
Where clear text = "1234" and encrypted text = "bMfQMUtMKDY="

<TrackData1>bMfQMUtMKDY=</TrackData1>
<TrackData1Length>4</TrackData1Length>

VerifyBeforeEncode

This key defines whether a comparison is done to check if the existing magstripe data before encoding matches data specified by the VerificationData key. The VerificationData is compared against Tracks 1, 2, and 3. If a match is found, encode operations are performed, otherwise encode operations are not performed and an error returns stating "Error Encode Verify".

Values	Value Description
String TRUE	Compare VerificationData against Tracks 1, 2, and 3 before encode operations.
String FALSE	No comparison before encode operations. (Default If Omitted)

Example XML Code: <VerifyBeforeEncode>TRUE</VerifyBeforeEncode>

VerificationData

This key defines the data to compare against existing magstripe data on Tracks 1, 2, and 3 given the VerifyBeforeEncode key is enabled. If a match is found, encode operations are performed, otherwise encode operations are not performed and an error returns stating "Error Encode Verify".

Values	Value Description
String	Data compared against Tracks 1, 2, and 3 before encode operations. (May be omitted if VerifyBeforeEncode is set to FALSE)

Example XML Code: <VerificationData>JOHN</VerificationData>

SECTION = SMARTCARD

The *SmartCard* section includes keys that control various operations associated with personalizing a SmartCard.

The following keys are included in the SmartCard section:

- Encrypted
- Type
- CommandID
- PropertyID
- PropertyType
- Data
- DataLength
- NumCommands

Encrypted

This key defines the whether encryption is enabled/disabled for all data lines. When set to TRUE all Data[n] lines within the SmartCard section are required to be encrypted.

Values	Value Description
String TRUE	Data string is encrypted.
String FALSE	Data string is not encrypted.

Example XML Code: <Encrypted>TRUE</Encrypted>

Type

This key is used to identify which type of SmartCard to process.

Values	Value Description
String Contact	Contact SmartCard
String Contactless	Contactless SmartCard (Not supported at this time)

Example XML Code: <Type> Contact </Type>

CommandID

This key is used to identify the command being sent. Please refer to the IntelliStripe 380 Command Reference manual for available supported commands related to SmartCard Application.

Values	Value Description
Numeric 0	Get Property
Numeric 1	Set Property
Numeric XX	XX refers to Command being requested

Example XML Code: <CommandID>0</CommandID>

PropertyID

This key is used to identify the property being requested. Please refer to the IntelliStripe 380 Command Reference manual for available supported properties related to SmartCard Application.

Values	Value Description
Numeric XX	XX refers to Property being requested

Example XML Code: <PropertyID>AE</PropertyID>

PropertyType

This key is used to identify the type of property requested.

Values	Value Description
Numeric 0	NONE
Numeric 1	DWORD
Numeric 2	STRING
Numeric 3	BOOLEAN
Numeric 4	BINARY

Example XML Code: <PropertyType>3</PropertyType>

Data

This key is used to identify the data being sent to the device.

Values	Value Description
String XX…	2 Byte ASCII Hex 0-9 A-F

Example XML Code: <Data>01</Data>

DataLength

This key defines the length of the clear text data characters of a given data line. This key is only required when the Encrypted tag is set to TRUE and the data strings are encrypted.

Values	Value Description
NumValue X	Number of clear text characters.
NumValue [n]	[n] = ID number for a given image. (0 < n < 100)

Example XML Code:
Where clear text = "1234" and encrypted text = "bMfQMUtMKDY="

<Data>bMfQMUtMKDY=</Data>
<DataLength>4</DataLength>

NumCommands

This key sets the number of multiple Smart Card commands to be processed from a single request. The NumCommands key is optional and, if not included, all other Smart Card keys require no enumeration. When NumCommands is included, each of the keys listed below must be terminated with a command count [n].

Key enumeration structure for multiple command processing:

- CommandID[n]
- PropertyID[n]
- PropertyType[n]
- Data[n]
- DataLength[n]

Values	Value Description
NumValue xx	Number of multiple Smart Cards to be enumerated.

Example XML Code: <NumCommands>3</NumCommands>

SECTION 4. KEYS RECEIVED FROM DEVICE

Section 4 describes all the Key-Value pairs received by the application from the device after the requested document has been processed. The device returns six sections: *DeviceStatus, DeviceInfo, DeviceCapabilities, CommandStatus*, *PrintStatus, SmartCard*.

All Key-Value pairs belong to a group referred to as a Section. The following are top level sections returned from the device after a job request:

Sections

- DeviceStatus
- DeviceInfo
- DeviceCapabilites
- CommandStatus
- SmartCard
- PrintStatus

SECTION = DeviceStatus

This *DeviceStatus* section includes keys that report the logical state of various modules, as well as status of all opto and mechanical sensors within the device.

The following keys are included in the DeviceStatus section:

- D2T2Display
- D2T2Status
- D2T2Sensors
- EncoderStatus
- EncoderStatusCode
- EncoderSensors
- EmbosserStatus
- QueueStatus

D2T2Display

This key returns values that represent the operational status of the virtual display used on the D2T2 printer.

Values	Value Description
String READY [H1] MENU H2	Virtual display is ready for operation

Example XML Code: <D2T2Display>READY[H1] MENU H2</D2T2Display>

D2T2Status

This key returns values that represent the operational status of the D2T2 thermal printer.

Values	Value Description
String READY	D2T2 Thermal printer is ready for operation
String BUSY	D2T2 Thermal printer is busy and not available for operation

Example XML Code: <D2T2Status>READY</D2T2Status>

D2T2Sensors

This key returns values that represent blocked/unblocked status of the opto-sensors in the D2T2 thermal printer.

Values	Value Description
String CIF = x CFL = x CPP = x CMG = x HFS = x SCO = x PHL = x CSC = x FLP = y HHP = y HVP = y PHP = y	CIF Card in feed sensor - 1/0 CFL Card Flip Sensor - 1/0 CPP Card Print Platen sensor - 1/0 CMG Card Magnetic sensor - 1/0 HFS Hopper Full sensor - 1/0 SCO Sensor Cover open - 1/0 PHL Print Head Latch - 1/0 CSC Card Smart Card sensor - 1/0 FLP Flipper Position values - (0 - 12) HHP Hopper Horizontal Position - (-1 - 11) HVP Hopper Vertical Position - (-1 - 11) PHP Print Head Position - (0 - 5) Where x: 0 = Unblocked 1 = Blocked Where y: (-1 - 12) = TBD

Example XML Code:
<D2T2Sensors>CIF=0;CFL=0;CPP=0;CMG=0;HFS=1;SCO=0;PHL=0;CSC=0;FLP=2;HHP=8;HVP=7;PHP=0</D2T2Sensors>

EncoderStatus

This key returns values that represent the operational status of the encoder module.

Values	Value Description
String READY	Encoder is ready for operation

Example XML Code: <EncoderStatus>READY </EncoderStatus>

EncoderStatusCode

This key returns values that represent the operational status of the encoder module.

Values	Value Description
String 0, 0	

Example XML Code: <EncoderStatusCode>0,0</EncoderStatusCode>

EncoderSensors

This key returns values that represent the status of the encoder module opto-sensors.

Values	Value Description
String FS = x MS = x RS = x AS = x	Where X 0 = Unblocked 1 = Blocked FS = Front Sensor, MS = Middle Sensor, RS = Rear Sensor AS = “Logical OR” of sensors FS, MS, and RS

Example XML Code: <EncoderSensors>FS=0,MS=0,RS=0,AS=0</EncoderSensors>

EmbosserStatus

This key returns values that represent the operational status of the embosser module.

Values	Value Description
String READY	Embosser is ready for operation

Example XML Code: <EmbosserStatus>READY </EmbosserStatus>

QueueStatus

This key returns values that represent the operational status of the Queue. During Ready status the machine will process queued request in first in first out order in which the request are received. During Paused status request may be received but are not processed until the status has ben set to Ready.

Values	Value Description
String READY	Queueing is active. Device will process any queued request.
String PAUSED	Queueing is paused.

Example XML Code: <QueueStatus>READY </QueueStatus>

SECTION = DeviceInfo

This *DeviceInfo* section includes keys that report the conditional status of media supplies. Additionally, serial numbers and revision codes for various modules are available.

The following keys are included in the DeviceStatus section:

- D2T2Serial
- D2T2RibbonPartNumber
- D2T2RibbonPercentage
- EncoderModel
- EncoderVersion
- ControllerModel
- ControllerVersion
- EjectBin
- RejectBin
- EnabledForRemainingHours
- EnabledForPeriodHours
- EnableStatus
- DeviceOptions
- CleanEncoderCount
- CleanPrinterCount

D2T2Serial

This key returns the unique serial number assigned to the D2T2 thermal printer.

Values	Value Description
String Axxxxxxx	Where X = 7 digit numeric serial number

Example XML Code: <D2T2Serial>A1234567</D2T2Serial>

D2T2RibbonPartNumber

This key returns the part number of the ribbon being used in the D2T2 thermal printer.

Values	Value Description
NumValue	5 digit part number of ribbon type being used in the D2T2 thermal printer

Example XML Code: <D2T2RibbonPartNumber>86230</D2T2RibbonPartNumber>

D2T2RibbonPercentage

This key returns a numeric string that is representative of the percentage of ribbon that remains in the D2T2 thermal printer.

Values	Value Description
NumValue	Percentage (%) of ribbon that remains in the D2T2 thermal printer.

Example XML Code: <D2T2RibbonPercentage>91</D2T2RibbonPercentage>

EncoderModel

This key returns a string that represents the model number of the encoder module.

Values	Value Description
String IntelliStripe 380	Returns model number of the encoder module.

Example XML Code: <EncoderModel>IntelliStripe 380</EncoderModel>

EncoderVersion

This key returns a string that is representative of the software version loaded onto the encoder module.

Values	Value Description
String 16051306A03	Software version of encoder module.

Example XML Code: <EncoderVersion>16051306A03</EncoderVersion>

ControllerModel

This key returns a string that is representative of the model number of the Main Logic PCB.

Values	Value Description
String ExpressCard 1000 MLB	Model number of Main Logic PCB

Example XML Code: <ControllerModel>Express Card 1000 MLB</ControllerModel>

ControllerVersion

This key returns a string that is representative of the software version loaded onto the Main Logic PCB

Values	Value Description
String 33050804B02	Version number of software on Main Logic PCB

Example XML Code: <ControllerVersion>33050804B02</ControllerVersion>

EjectBin

This key returns a string that indicates if the Eject Bin is “Full” or “OK”.

Values	Value Description
String OK	Eject bin is not full. “OK” for continued operation.
String FULL	Eject bin is “Full”. Needs to be emptied.

Example XML Code: <EjectBin>OK</EjectBin>

RejectBin

This key returns a string that indicates if the Reject Bin is “Full” or “OK”.

Values	Value Description
String OK	Eject bin is not full. “OK” for continued operation.
String FULL	Eject bin is “Full”. Needs to be emptied.

Example XML Code: <RejectBin>OK</RejectBin>

EnabledForRemainingHours

This key returns a numeric value that is representative of the hours remaining for which the device is enabled. This value decreases by one each hour during operation and when device is not powered.

Values	Value Description
NumValue	Current value for remaining enabled hours.

Example XML Code: <EnabledForRemainingHours>168</EnabledForRemainingHours>

EnabledForPeriodHours

This key returns a numeric value that is representative of the period of time in hours the device is set to be enabled. This key holds the starting value of remaining hours and does not decrease over time.

Values	Value Description
NumValue	Starting value for remaining enabled hours.

Example XML Code: <EnabledForPeriodHours>168</EnabledForPeriodHours>

EnableStatus

This key returns a numeric value of the enabled health to be used by support help.

Values	Value Description
NumValue	Health of enabled status.

Example XML Code: <EnableStatus>2</EnableStatus>

DeviceOptions

This key returns a numeric value that is representative of the enabled functionality for the machine.

Values	Value Description
NumValue	Enabled functionality for the machine.

Example XML Code: <DeviceOptions>127</DeviceOptions>

Maintenance/CleanEncoderCount

This key returns a numeric value that is representative of the numer of job request. When the count reaches 2000, the machine will display a message for cleaning the encoder and printer. This value may be reset to 0 at the device.

Values	Value Description
NumValue X	Counter for the number of job request. X >= 2000 causes a clean printer message to be displayed.

Example XML Code:
<Maintenance>
<CleanEncoderCount>0</CleanEncoderCount>
<Mainteneance>

Maintenance/CleanPrinterCount

This key returns a numeric value that is representative of the numer of job request. When the count reaches 400, the machine will display a message for cleaning the printer head. This value may be reset to 0 at the device.

Values	Value Description
NumValue X	Counter for the number of job request. X >= 400 causes a clean printer message to be displayed.

Example XML Code:

```
<Maintenance>
        <CleanPrinterCount>0</CleanPrinterCount>
<Mainteneance>
```

SECTION = DeviceCapabilities

The *DeviceCapabilities* section includes keys that report the "device present" status of the various modules within the EC1000. This information can be useful in determining the specific configuration and capabilities that a given device may have.

The following keys are included in the DeviceCapabilities section:

- Encode
- Print
- Emboss
- PrintEmboss

Encode

This key returns a value that indicates if an encode module is installed within the EC1000.

Values	Value Description
String TRUE	Encode module is installed.
String FALSE	Encode module is not installed.

Example XML Code: <Encode>TRUE</Encode>

Print

This key returns a value that indicates if a D2T2 thermal printer is installed within the EC1000.

Values	Value Description
String TRUE	Printer module is installed.
String FALSE	Printer module is not installed.

Example XML Code: <Print>TRUE</Print>

Emboss

This key returns a value that indicates if an embosser module is installed within the EC1000.

Values	Value Description
String TRUE	Embosser module is installed.
String FALSE	Embosser module is not installed.

Example XML Code: <Emboss>TRUE</Emboss>

PrintEmboss

This key returns a value that indicates if a Print ribbon system is installed within the EC1000.

Values	Value Description
String TRUE	Print ribbon module is installed.
String FALSE	Print ribbon module is not installed.

Example XML Code: <PrintEmboss>TRUE</PrintEmboss>

SECTION = CommandStatus

The *CommandStatus* section includes keys that report the status of commands sent to the device.

The following keys are included in the CommandStatus section:

- CommandID
- ReturnCode
- ReturnMsg
- UniqueID
- MagnePrint
- Track1Data
- Track2Data
- Track3Data
- Location

CommandID

This key returns a numeric value that represents the command that was sent to the device.

Values	Value Description
NumValue X	Where X: CommandID sent to the device

Example XML Code: <CommandID>0</CommandID>

ReturnCode

This key returns a numeric value that represents the “return code” for a given operation.
”0” represents a successful operation. Any non-zero value represents some form of an error condition.
A definition of return code values is provided in Appendix B.

Values	Value Description
NumValue X	Where X: 0 = Successful operation. Non-0 = Error code. (See list in Appendix B)

Example XML Code: <ReturnCode>0</ReturnCode>

ReturnMsg

This key returns a string that represents the “return message” for a given operation. A definition of return messages is in Appendix B.

Values	Value Description
String MSG	Where MSG = Return Message List (See list in Appendix B)

Example XML Code: <ReturnMsg>OK</ReturnMsg>

UniqueID

This key returns a string that represents the Unique ID tag that the host application previously assigned to a given print job.

Values	Value Description
String “MSG”	Where MSG = Unique ID string previously assigned by host application to a given print job

Example XML Code: <UniqueID>Card ID 00023465</UniqueID>

MagnePrint

This key returns the string that is representative of the MagnePrint data that was read from the magnetic stripe.

Values	Value Description
String “MAGNEPRINT”	Returns the MagnePrint value on the magstripe

Example XML Code: <MagnePrint>[MAGNEPRINT DATA]</MagnePrint>

Track1Data

This key returns the string that is representative of the TK1 data that was encoded on the magnetic stripe.

Values	Value Description
String “TRACK1”	Returns the data encoded on Track 1 magstripe

Example XML Code: <Track1Data>[TRACK1 DATA]</Track1Data>

Track2Data

This key returns the string that is representative of the TK2 data that was encoded on the magnetic stripe.

Values	Value Description
String "TRACK2"	Returns the data encoded on Track 2 magstripe

Example XML Code: <Track2Data>[TRACK2 DATA]</Track2Data>

Track3Data

This key returns the string that is representative of the TK3 data that was encoded on the magnetic stripe.

Values	Value Description
String "TRACK3"	Returns the data encoded on Track 3 magstripe

Example XML Code: <Track3Data>[TRACK3 DATA]</Track3Data>

Location

This key returns a string that is representative of the end location of the card process.
Typically "Eject Bin" or "Reject Bin".

Values	Value Description
String "EJECT BIN"	Card at Eject Bin.
String "REJECT BIN"	Card at Reject Bin.

Example XML Code: <Location>Eject BIN</Location>

SECTION = PrintStatus

The *PrintStatus* section is a combination of the CommandStatus, Device Information, and DeviceStatus sections.

PrintStatus is returned at the end of each print-job process, and should be used by the host application to confirm all final status aspects of a given job. This includes verification of magstripe and Smart Card data, receipt of MagnePrint data, and verification if the card was successfully placed in the Eject Bin or captured in the Reject Bin.

SECTION = SmartCard

The *SmartCard* section includes responses to SmartCard commands.

The following keys are included in the SmartCard section:

- Type
- CommandID
- PropertyID
- PropertyType
- ResultCode
- ReturnCode
- Data
- DataLength
- NumCommands

Type

This key is echoed back as a response to a SmartCard request to identify the Type that was sent.

Values	Value Description
String Contact	Contact SmartCard
String Contactless	Contactless SmartCard (Not supported at this time)

Example XML Code: <Type> Contact </Type>

CommandID

This key is echoed back as a response to a SmartCard request to identify the command that was sent. Please refer to the IntelliStripe 380 Command Reference manual for available supported commands related to SmartCard Application.

Values	Value Description
Numeric 0	Get Property
Numeric 1	Set Property
Numeric XX	XX refers to Command that was requested

Example XML Code: <CommandID>80</CommandID>

PropertyID

This key is echoed back as a response to a SmartCard request to identify the PropertyID that was sent. Please refer to the IntelliStripe 380 Command Reference manual for available supported properties related to SmartCard Application

Values	Value Description
Numeric XX	XX refers to Property that was requested

Example XML Code: <PropertyID>AE</PropertyID>

PropertyType

This key is echoed back as a response to a SmartCard request to identify the PropertyType that was sent.

Values	Value Description
Numeric 0	NONE
Numeric 1	DWORD
Numeric 2	STRING
Numeric 3	BOOLEAN
Numeric 4	BINARY

Example XML Code: <PropertyType>3</PropertyType>

ResultCode

This key indicates the result code that is sent by the device in response to the request.

Values	Value Description
Numeric XX	XX refers to Property that was requested

Example XML Code: <ResultCode>0</ResultCode>

ReturnCode

This key indicates the return code that is sent by the SmartCard driver in response to the request.

Values	Value Description
Numeric XX	XX refers to driver return code

Example XML Code: <ReturnCode>0</ReturnCode>

Data

This key is used to identify the data being sent by the device.

Values	Value Description
String XX…	2 Byte ASCII Hex 0-9 A-F

Example XML Code: <Data>0101ABCD</Data>

DataLength

This key is used to identify the data length being sent by the device.

Values	Value Description
Num XX…	Length of Data

Example XML Code: <DataLength>2</DataLength>

NumCommands

This key is the number of multiple Smart Card commands that were requested to the device. The NumCommands key is optional and, if not included, all other Smart Card keys will not be enumerated.

Values	Value Description
NumValue xx	Number of multiple Smart Cards to be enumerated.

Example XML Code: <NumCommands>3</NumCommands>

SECTION 5. INDEPENDENT AND SMART CARD COMMANDS

Section 5 describes all the Key-Value pairs sent to the device to process independent card processing and Smart Card commands.

All Key-Value pairs belong to a group referred to as a Section. The following are top level sections sent during a job request:

Sections

- CommandInfo
- SmartCard

SECTION = CommandInfo

The *CommandInfo* section includes keys that allow the application to select which independent command to process.

The following keys are included in the *CommandInfo* section:

- CardHopperIndex
- UniqueID
- CommandID
- Description
- ResponseURL
- OwnerID
- SendCancelOnError

CardHopperIndex

This key determines which card hopper will be used for a given card processing operation.

Values	Value Description
Num Value 0	Hopper 1
Num Value 1	Hopper 2
Num Value 2	Toggle Between Hopper 1 and Hopper 2
Num Value 3	Current Hopper
Num Value 4	Exception Feed Slot (Manual Card Insertion)

Example XML Code: <CardHopperIndex>1</CardHopperIndex>

UniqueID

This key allows the host application to present a unique "job number" ID string that can be used to name (tag) a particular card personalization job sequence. This may be useful for auditing purposes to track whether a particular card print job was successful or not.

Values	Value Description
String "Card ID 00023654"	ID string can be any alphanumeric string that the host application defines. This will be used as a "job tag" to track a given card personalization sequence.

Example XML Code: <UniqueID>Card ID 00023654</UniqueID>

CommandID

This key determines which independent command will be used for a given card processing operation.

Values	Value Description
Num Value 1	Move Card to Encoder
Num Value 2	Eject Card
Num Value 3	Abort Card
Num Value 4	Smart Card
Num Value 5	Encode Card
Num Value 6	Read Card
Num Value 7	Process a Card This command indicates that the personalization of a card will be done in one request
Num Value 10	Move Card to Contactless Station
Num Value 11	Move Card to Consume Position
Num Value 14	Move Card to Encoder with Notification

Example XML Code: <CommandID>1</CommandID>

Description

This key accepts a description string which is to be displayed for the queueing version of the ExpressCard 1000 front panel application. This key allows the host application to provide conceptual information to describe a particular card personalization job sequence.

Values	Value Description
String "Gift Card"	Description string can be any alphanumeric string that the host application defines. Where size of string is 0 to 25 characters.
String ""	(Default If Omitted)

Example XML Code: <Description>Gift Card</Description>

ResponseURL

This key defines the network location to which a notification will be posted back informing that the ExpressCard 1000 machine has moved a card into the encoder and is awaiting further independent command instructions. To enable a notification, set the Command ID to 14, define the ResponseURL, and then send the request to the ExpressCard 1000 web page QueueJob.aspx.

After the notification is sent from the ExpressCard 1000 device, the host application may send independent commands to complete the personalization sequence.

Values	Value Description
String "https://address/Page"	ResponseURL string can be any alphanumeric string that the host application defines.
String ""	(Default If Omitted)

Example XML Code: <ResponseURL>https://Client1/GetNotification.aspx</ResponseURL>

Example Request:

```
<?xml version="1.0"?>
<DeviceSettings>
    <CommandInfo>
        <Description>Gift Card</Description>
        <OwnerID>Central Bank</OwnerID>
        <UniqueID>Sample1</UniqueID>
        <CommandID>14</CommandID>
        <CardHopperIndex>0</CardHopperIndex>
        <ResponseURL>https://Client1/GetNotification.aspx</ResponseURL>
     </CommandInfo>
</DeviceSettings>
```

OwnerID

This key allows the host application to present an owner ID string that can be used to indicate the locale from which a particular card personalization job sequence originated.

Values	Value Description
String "Central Branch"	OwnerID string can be any alphanumeric string that the host application defines.
String ""	(Default If Omitted)

Example XML Code: <OwnerID>Central Branch</OwnerID>

SendCancelOnError

This key defines whether a Busy state will automatically be cleared if a card is not fed from the hopper to the encoder. When set to TRUE, a cancel is invoked to place the device into Ready state with a response of "Timeout Moving Card to Smart Card Station". When set to FALSE, the device will remain in its current state until further commands are sent.

Values	Value Description
String TRUE	Auto cancellation is enabled.
String FALSE	Auto cancellation is disabled. (Default If Omitted)

Example XML Code: <SendCancelOnError>TRUE</ SendCancelOnError >

SECTION = SmartCard

The *SmartCard* section includes keys that allow the application to select which independent Smart Card properties and commands to process.

The following keys are included in the SmartCard section:

- Type
- CommandID
- PropertyID
- PropertyType
- Data
- DataLength
- NumCommands

Type

This key is used to identify which type of SmartCard to process.

Values	Value Description
String Contact	Contact Smart Card
String Contactless	Contactless SmartCard (Not supported at this time)

CommandID

This key determines which independent Smart Card command will be used for a given card processing operation.

Given <PropertyType> is Property	
Values	**Value Description**
NumValue 0	Get Property
NumValue 1	Set Property

Given <PropertyType> is Command	
Values	**Value Description**
NumValue 80	ProcessSmartCardPowerOn
NumValue 81	ProcessSmartCardPowerOff
NumValue 82	ProcessSmartCardWarmReset
NumValue 85	ProcessSmartCardAPDU

PropertyID

This key determines which Smart Card property to set or get.

Values	Value Description
NumValue [n] HEX	Where [n] = property number. (0 < n)

PropertyType

This key determines the data type for the property.

Values	Value Description
NumValue 1	Dword
NumValue 2	String
NumValue 3	Byte

Data

This key sets and returns the data for Smart Card property or command operations.

Values	Value Description
Stirng xx	2 Byte ASCII Hex 0-9 A-F

DataLength

This key returns the length of the data returned from a Smart Card property or command operation.

Values	Value Description
NumValue xx	Length of Data

NumCommands

This key sets the number of multiple Smart Card commands to be processed from a single request. The NumCommands key is optional and, if not included, all other Smart Card keys require no enumeration. When NumCommands is included, each of the keys listed below must be terminated with a command count [n].

Key enumeration structure for multiple command processing:

- CommandID[n]
- PropertyID[n]
- PropertyType[n]
- Data[n]

Values	Value Description
NumValue xx	Number of multiple Smart Cards to be enumerated.

Example XML Code: <NumCommands>3</NumCommands>

This example demonstrates three commands (80,0,85) in a single request. First, the smart card is powered on (80). Second, a property value is requested using get property (0). And last, APDU data is sent (85).

Smart Card Request

```
<?xml version="1.0" ?>
<DeviceSettings>
     <CommandInfo>
          <UniqueID>IntelliCAT</UniqueID>
          <CommandID>4</CommandID>
      </CommandInfo>
      <SmartCard>
          <Type>Contact</Type>
          <NumCommands>3</NumCommands>

          <CommandID1>80</CommandID1>

          <CommandID2>0</CommandID2>
          <PropertyID2>AE</PropertyID2>
          <PropertyType2>3</PropertyType2>

          <CommandID3>85</CommandID3>
          <Data3>00A40000023F00</Data3>
      </SmartCard>
</DeviceSettings>
```

Smart Card Response

```
<?xml version="1.0"?>
<PrintStatus>
     <CommandStatus>
          <CommandID>4</CommandID>
         <ReturnCode>0</ReturnCode>
         <ReturnMsg>OK</ReturnMsg>
         <UniqueID>IntelliCAT</UniqueID>
         <Location>Encoder</Location>
     <SmartCard>
         <Type>Contact</Type>
         <NumCommands>3</NumCommands>
         <CommandID1>80</CommandID1>
         <PropertyID1></PropertyID1>
         <PropertyType1></PropertyType1>
         <ResultCode1>0</ResultCode1>
         <ReturnCode1>0</ReturnCode1>
         <Data1>3BBF180080317032535441524
         34F532053323120439000 9C</Data1>
         <DataLength1>48</DataLength1>
         <CommandID2>0</CommandID2>
         <PropertyID2>AE</PropertyID2>
         <PropertyType2>3</PropertyType2>
         <ResultCode2>0</ResultCode2>
         <ReturnCode2>0</ReturnCode2>
         <Data2>00</Data2>
         <DataLength2>2</DataLength2>
         <CommandID3>85</CommandID3>
         <PropertyID3></PropertyID3>
         <PropertyType3></PropertyType3>
         <ResultCode3>0</ResultCode3>
         <ReturnCode3>0</ReturnCode3>
         <Data3>6985</Data3>
         <DataLength3>4</DataLength3>
     </SmartCard>

     </CommandStatus>
     <DeviceInformation>…</DeviceInformation>
     <DeviceStatus>…</DeviceStatus>
</PrintStatus>
```

Example Request and Response for Independent and Smart Card Commands are below:

- MoveCard
- EjectCard
- AbortCard
- EncodeCard
- ReadCard

- ProcessGetProperty
- ProcessSetProperty
- ProcessSmartCardAPDU
- ProcessSmartCardPowerOff
- ProcessSmartCardPowerOn
- ProcessSmartCardPropertyCondRpt
- ProcessSmartCardWarmReset

MoveCard:

MoveCard - Example Request:

```
<?xml version="1.0" ?>
<DeviceSettings>
    <CommandInfo>
        <UniqueID>IntelliCAT</UniqueID>
        <CommandID>1</CommandID>
        <CardHopperIndex>0</CardHopperIndex>
   </CommandInfo>
</DeviceSettings>
```

MoveCard - Example Response:

```
<?xml version="1.0" ?>
<PrintStatus>
    <CommandStatus>
        <CommandID>1</CommandID>
        <ReturnCode>0</ReturnCode>
        <ReturnMsg>OK</ReturnMsg>
        <UniqueID>IntelliCAT</UniqueID>
        <MagnePrint />
        <Track1Data />
        <Track2Data />
        <Track3Data />
        <Location>Encoder</Location>
    </CommandStatus>
    <DeviceInformation>
        <D2T2Serial>A1234567</D2T2Serial>
        <D2T2RibbonPartNumber>86230</D2T2RibbonPartNumber>
        <D2T2RibbonPercentage>56%</D2T2RibbonPercentage>
        <ControllerModel>Express Card 1000 MLB</ControllerModel>
        <ControllerVersion>33050804A0</ControllerVersion>
        <EncoderModel>IntelliStripe 380</EncoderModel>
        <EncoderVersion>16051306B08</EncoderVersion>
        <EjectBin>OK</EjectBin>
         <RejectBin>OK</RejectBin>
     </DeviceInformation>
    <DeviceStatus>
        <D2T2Display>PRINTER STATUS SMART ENCODING CANCEL PAUSE</D2T2Display>
        <D2T2Status>Printing</D2T2Status>
        <D2T2Sensors>CIF=0;CFL=0;CPP=0;CMG=0;HFS=0;SCO=0;PHL=0;
        CSC=1;FLP=4;HHP=8;HVP=6;PHP=0;
        </D2T2Sensors>
        <EmbosserStatus>Ready</EmbosserStatus>
        <EmbosserSensors>PR.CE=0,XY.YP=1,XY.XH=0,XY.YH=0,CL.CD=0,XY.YE=0,EB.TC=0,EB.BC=0,
        EB.CE=1,EB.WH=0,CS.TD=1,CS.AD=1,EX.EB=1,EX.RB=0,EX.T1=1,EX.YT=1,XY.H1=0,XY.H2=0,
        EX.T2=2,EX.T3=2,EX.T4=2,RB.PC=2,RB.RD=2,TP.CD=2,EX.FP=2,EB.TR=2
        </EmbosserSensors>
        <EncoderStatus>Encoder Received Card</EncoderStatus>
        <EncoderStatusCode>0,0</EncoderStatusCode>
        <EncoderSensors>FS=0,MS=0,RS=1,AS=1</EncoderSensors>
   </DeviceStatus>
</PrintStatus>
```

MoveCard:

MoveCard Contactless - Example Request:

```
<?xml version="1.0" ?>
<DeviceSettings>
    <CommandInfo>
        <UniqueID>IntelliCAT</UniqueID>
        <CommandID>10</CommandID>
        <CardHopperIndex>0</CardHopperIndex>
   </CommandInfo>
</DeviceSettings>
```

MoveCard Contactless - Example Response:

```
<?xml version="1.0" ?>
<PrintStatus>
    <CommandStatus>
        <CommandID>10</CommandID>
        <ReturnCode>0</ReturnCode>
        <ReturnMsg>OK</ReturnMsg>
        <UniqueID>IntelliCAT</UniqueID>
        <MagnePrint />
        <Track1Data />
        <Track2Data />
        <Track3Data />
        <Location>Encoder</Location>
    </CommandStatus>
    <DeviceInformation>
        <D2T2Serial>A1234567</D2T2Serial>
        <D2T2RibbonPartNumber>86230</D2T2RibbonPartNumber>
        <D2T2RibbonPercentage>56%</D2T2RibbonPercentage>
        <ControllerModel>Express Card 1000 MLB</ControllerModel>
        <ControllerVersion>33050804A0</ControllerVersion>
        <EncoderModel>IntelliStripe 380</EncoderModel>
        <EncoderVersion>16051306B08</EncoderVersion>
        <EjectBin>OK</EjectBin>
        <RejectBin>OK</RejectBin>
    </DeviceInformation>
    <DeviceStatus>
        <D2T2Display>PRINTER STATUS SMART ENCODING CANCEL PAUSE</D2T2Display>
        <D2T2Status>Printing</D2T2Status>
        <D2T2Sensors>CIF=0;CFL=0;CPP=0;CMG=0;HFS=0;SCO=0;PHL=0;
        CSC=1;FLP=4;HHP=8;HVP=6;PHP=0;
        </D2T2Sensors>
        <EmbosserStatus>Ready</EmbosserStatus>
        <EmbosserSensors>PR.CE=0,XY.YP=1,XY.XH=0,XY.YH=0,CL.CD=0,XY.YE=0,EB.TC=0,EB.BC=0,
        EB.CE=1,EB.WH=0,CS.TD=1,CS.AD=1,EX.EB=1,EX.RB=0,EX.T1=1,EX.YT=1,XY.H1=0,XY.H2=0,
        EX.T2=2,EX.T3=2,EX.T4=2,RB.PC=2,RB.RD=2,TP.CD=2,EX.FP=2,EB.TR=2
        </EmbosserSensors>
        <EncoderStatus>Encoder Received Card</EncoderStatus>
        <EncoderStatusCode>0,0</EncoderStatusCode>
        <EncoderSensors>FS=0,MS=0,RS=1,AS=1</EncoderSensors>
    </DeviceStatus>
</PrintStatus>
```

EjectCard:

EjectCard - Example Request:

```
<?xml version="1.0" ?>
<DeviceSettings>
    <CommandInfo>
        <UniqueID>IntelliCAT</UniqueID>
        <CommandID>2</CommandID>
    </CommandInfo>
</DeviceSettings>
```

EjectCard - Example Response:

```
<?xml version="1.0" ?>
<PrintStatus>
    <CommandStatus>
        <CommandID>2</CommandID>
        <ReturnCode>0</ReturnCode>
        <ReturnMsg>OK</ReturnMsg>
        <UniqueID>IntelliCAT</UniqueID>
         <MagnePrint>010008003994973ED7E3761834094FFC33105FA95EF97F3F18FBEF0A4AC1
        E7F041E0A3EE350D2C3F398ECEFEF5F817613C40887F288C
        </MagnePrint>
        <Track1Data>%B99991234567890^STERLING/JOANNE^99121011445?</Track1Data>
        <Track2Data>;9999991234567890=99121011445?</Track2Data>
        <Track3Data>;019999991234567890=00101220100005095016020000005030001991210123
        3456789?
        </Track3Data>
        <Location>Eject BIN</Location>
    </CommandStatus>
    <DeviceInformation>
        <D2T2Serial>A1234567</D2T2Serial>
        <D2T2RibbonPartNumber>86230</D2T2RibbonPartNumber>
        <D2T2RibbonPercentage>56%</D2T2RibbonPercentage>
        <ControllerModel>Express Card 1000 MLB</ControllerModel>
        <ControllerVersion>33050804A0</ControllerVersion>
        <EncoderModel>IntelliStripe 380</EncoderModel>
        <EncoderVersion>16051306B08</EncoderVersion>
        <EjectBin>OK</EjectBin>
        <RejectBin>OK</RejectBin>
    </DeviceInformation>
    <DeviceStatus>
        <D2T2Display>READY H1 MENU H2</D2T2Display>
        <D2T2Status>Ready</D2T2Status>
        <D2T2Sensors>CIF=0;CFL=0;CPP=0;CMG=0;HFS=0;SCO=0;PHL=0;CSC=0;FLP=2;HHP=8;
        HVP=7;PHP=0;
        </D2T2Sensors>
        <EmbosserStatus>Ready</EmbosserStatus>
        <EmbosserSensors>PR.CE=0,XY.YP=1,XY.XH=1,XY.YH=0,CL.CD=0,XY.YE=0,EB.TC=0,
        EB.BC=0,EB.CE=1,EB.WH=0,CS.TD=1,CS.AD=1,EX.EB=1,EX.RB=0,EX.T1=1,EX.YT=1,
        XY.H1=0,XY.H2=0,EX.T2=2,EX.T3=2,EX.T4=2,RB.PC=2,RB.RD=2,TP.CD=2,
        EX.FP=2,EB.TR=2</EmbosserSensors>
        <EncoderStatus>Ready</EncoderStatus>
        <EncoderStatusCode>0,0</EncoderStatusCode>
        <EncoderSensors>FS=0,MS=0,RS=0,AS=0</EncoderSensors>
    </DeviceStatus>
</PrintStatus>
```

AbortCard:

AbortCard - Example Request:

```
<?xml version="1.0" ?>
<DeviceSettings>
    <CommandInfo>
        <UniqueID>IntelliCAT</UniqueID>
        <CommandID>3</CommandID>
    </CommandInfo>
</DeviceSettings>
```

AbortCard - Example Response:

```
<?xml version="1.0" ?>
<PrintStatus>
    <CommandStatus>
        <CommandID>3</CommandID>
        <ReturnCode>69</ReturnCode>
        <ReturnMsg>Error UserAbort</ReturnMsg>
        <UniqueID>IntelliCAT</UniqueID>
        <MagnePrint />
        <Track1Data />
        <Track2Data />
        <Track3Data />
        <Location>Reject BIN</Location>
    </CommandStatus>
    <DeviceInformation>
        <D2T2Serial>A1234567</D2T2Serial>
        <D2T2RibbonPartNumber>86230</D2T2RibbonPartNumber>
        <D2T2RibbonPercentage>56%</D2T2RibbonPercentage>
        <ControllerModel>Express Card 1000 MLB</ControllerModel>
        <ControllerVersion>33050804A0</ControllerVersion>
        <EncoderModel>IntelliStripe 380</EncoderModel>
        <EncoderVersion>16051306B08</EncoderVersion>
        <EjectBin>OK</EjectBin>
        <RejectBin>OK</RejectBin>
    </DeviceInformation>
    <DeviceStatus>
        <D2T2Display>READY H1 MENU H2</D2T2Display>
        <D2T2Status>Ready</D2T2Status>
        <D2T2Sensors>CIF=0;CFL=0;CPP=0;CMG=0;HFS=0;SCO=0;PHL=0;CSC=0;FLP=2;HHP=8;
        HVP=7;PHP=0;</D2T2Sensors>
        <EmbosserStatus>Ready</EmbosserStatus>
        <EmbosserSensors>PR.CE=0,XY.YP=1,XY.XH=0,XY.YH=0,CL.CD=0,XY.YE=0,EB.TC=0,
        EB.BC=0,EB.CE=1,EB.WH=0,CS.TD=1,CS.AD=1,EX.EB=1,EX.RB=0,EX.T1=1,EX.YT=1,
        XY.H1=0,XY.H2=0,EX.T2=2,EX.T3=2,EX.T4=2,RB.PC=2,RB.RD=2,TP.CD=2,EX.FP=2,EB.TR=2
        </EmbosserSensors>
        <EncoderStatus>Ready</EncoderStatus>
        <EncoderStatusCode>0,0</EncoderStatusCode>
        <EncoderSensors>FS=0,MS=0,RS=0,AS=0</EncoderSensors>
    </DeviceStatus>
</PrintStatus>
```

EncodeCard:

EncodeCard - Example Request:

```
<?xml version="1.0" ?>
<DeviceSettings>
    <CommandInfo>
        <UniqueID>IntelliCAT</UniqueID>
        <CommandID>5</CommandID>
    </CommandInfo>
    <MagEncode>
        <MagEncodeEnable>TRUE</MagEncodeEnable>
        <Coercivity>AUTO</Coercivity>
        <StripeRetry>3</StripeRetry>
        <NumTracks>3</NumTracks>
        <TrackBPI1>210</TrackBPI1>
        <TrackBPC1>7</TrackBPC1>
        <TrackData1>B49991234567890^STERLING/JOANNE^99121011445</TrackData1>
        <TrackBPI2>75</TrackBPI2>
        <TrackBPC2>5</TrackBPC2>
        <TrackData2>4999991234567890=99121011445</TrackData2>
        <TrackBPI3>210</TrackBPI3>
        <TrackBPC3>5</TrackBPC3>
        <TrackData3>019999991234567890=00101220100005095016020000050300019
        91210123456789</TrackData3>
    </MagEncode>
</DeviceSettings>
```

EncodeCard - Example Response:

```
<?xml version="1.0" ?>
<PrintStatus>
    <CommandStatus>
        <CommandID>5</CommandID>
        <ReturnCode>0</ReturnCode>
        <ReturnMsg>OK</ReturnMsg>
        <UniqueID>IntelliCAT</UniqueID>
        <MagnePrint>010008E47B4B5276C03C38C6B8205C51CE8E5046C5E49C5198FD5046CEE
        742E311FB953FB9AFE356905F37EEF84881871F63E93FCE58</MagnePrint>
        <Track1Data>%B49991234567890^STERLING/JOANNE^99121011445?</Track1Data>
        <Track2Data>;4999991234567890=99121011445?</Track2Data>
        <Track3Data>;019999991234567890=00101220100005095016020000005030001991
        210123456789?</Track3Data>
        <Location>Encoder</Location>
    </CommandStatus>
    <DeviceInformation>
        <EncoderModel>IntelliStripe 380</EncoderModel>
        <EncoderVersion>16051306B08</EncoderVersion>
        <EjectBin>OK</EjectBin>
        <RejectBin>OK</RejectBin>
    </DeviceInformation>
    <DeviceStatus>
        <D2T2Display>PRINTER STATUS SMART ENCODING CANCEL PAUSE</D2T2Display>
        <D2T2Status>Printing</D2T2Status>
        <D2T2Sensors>CIF=0;CFL=0;CPP=0;CMG=0;HFS=0;SCO=0;PHL=0;CSC=1;FLP=4;
        HHP=8;HVP=6;PHP=0;
        </D2T2Sensors>
        <EmbosserStatus>Ready</EmbosserStatus>
```

```
        <EmbosserSensors>PR.CE=0,XY.YP=1,XY.XH=0,XY.YH=0,XY.H1=1,XY.H2=1,CL.CD=0,
        XY.YE=0,EB.TC=1,EB.BC=1,EB.CE=1,EB.WH=0,CS.TD=1,CS.AD=1,EX.EB=0,EX.RB=0,
        EX.T1=1,EX.YT=1</EmbosserSensors>
        <EncoderStatus>Read Card Completed</EncoderStatus>
        <EncoderStatusCode>0,0</EncoderStatusCode>
        <EncoderSensors>FS=0,MS=0,RS=1,AS=1</EncoderSensors>
    </DeviceStatus>
</PrintStatus>
```

ReadCard:

ReadCard - Example Request:

```
<?xml version="1.0" ?>
<DeviceSettings>
    <CommandInfo>
        <UniqueID>IntelliCAT</UniqueID>
        <CommandID>6</CommandID>
    </CommandInfo>
</DeviceSettings>
```

ReadCard - Example Response:

```
<?xml version="1.0" ?>
<PrintStatus>
    <CommandStatus>
        <CommandID>6</CommandID>
        <ReturnCode>0</ReturnCode>
        <ReturnMsg>OK</ReturnMsg>
        <UniqueID>IntelliCAT</UniqueID>
        <MagnePrint>010008A48779C20C3930D034969E8F85503D40C6F80FD871170A15A8D5C894A1
        B6733105DEE401831127623FC18185C111D560FAD857</MagnePrint>
        <Track1Data>%B49991234567890^STERLING/JOANNE^99121011445?</Track1Data>
        <Track2Data>;4999991234567890=99121011445?</Track2Data>
        <Track3Data>;019999991234567890=00101220100005095016020000005030001991
        210123456789?</Track3Data>
        <Location>Encoder</Location>
    </CommandStatus>
    <DeviceInformation>
        <EncoderModel>IntelliStripe 380</EncoderModel>
        <EncoderVersion>16051306B08</EncoderVersion>
        <EjectBin>OK</EjectBin>
        <RejectBin>OK</RejectBin>
    </DeviceInformation>
    <DeviceStatus>
        <D2T2Display>PRINTER STATUS SMART ENCODING CANCEL PAUSE</D2T2Display>
        <D2T2Status>Printing</D2T2Status>
        <D2T2Sensors>CIF=0;CFL=0;CPP=0;CMG=0;HFS=0;SCO=0;PHL=0;CSC=1;FLP=4;
        HHP=8;HVP=6;PHP=0;</D2T2Sensors>
        <EmbosserStatus>Ready</EmbosserStatus>
        <EmbosserSensors>PR.CE=0,XY.YP=1,XY.XH=0,XY.YH=0,XY.H1=1,XY.H2=1,CL.CD=0,
        XY.YE=0,EB.TC=1,EB.BC=1,EB.CE=1,EB.WH=0,CS.TD=1,CS.AD=1,EX.EB=0,
        EX.RB=0,EX.T1=1,EX.YT=1</EmbosserSensors>
        <EncoderStatus>Read Card Completed</EncoderStatus>
        <EncoderStatusCode>0,0</EncoderStatusCode>
        <EncoderSensors>FS=0,MS=0,RS=1,AS=1</EncoderSensors>
    </DeviceStatus>
</PrintStatus>
```

ProcessGetProperty:

ProcessGetProperty - Example Request:

```
<?xml version="1.0" ?>
<DeviceSettings>
    <CommandInfo>
        <UniqueID>IntelliCAT</UniqueID>
        <CommandID>4</CommandID>
    </CommandInfo>
    <SmartCard>
        <Type>Contact</Type>
        <CommandID>0</CommandID>
        <PropertyID>AE</PropertyID>
        <PropertyType>3</PropertyType>
    </SmartCard>
</DeviceSettings>
```

ProcessGetProperty - Example Response:

```
<?xml version="1.0" ?>
<PrintStatus>
    <CommandStatus>
        <CommandID>4</CommandID>
        <SmartCard>
            <Type>Contact</Type>
            <CommandID>0</CommandID>
            <PropertyID>AE</PropertyID>
            <PropertyType>3</PropertyType>
            <ResultCode>0</ResultCode>
            <ReturnCode>0</ReturnCode>
            <Data>00</Data>
            <DataLength>2</DataLength>
        </SmartCard>
        <ReturnCode>0</ReturnCode>
        <ReturnMsg>OK</ReturnMsg>
        <UniqueID>IntelliCAT</UniqueID>
        <Location>Encoder</Location>
    </CommandStatus>
    <DeviceInformation>
        <EncoderModel>IntelliStripe 380</EncoderModel>
        <EncoderVersion>16051306B08</EncoderVersion>
        <EjectBin>OK</EjectBin>
        <RejectBin>OK</RejectBin>
    </DeviceInformation>
    <DeviceStatus>
        <D2T2Display>PRINTER STATUS SMART ENCODING CANCEL PAUSE</D2T2Display>
        <D2T2Status>Printing</D2T2Status>
        <D2T2Sensors>CIF=0;CFL=0;CPP=0;CMG=0;HFS=0;SCO=0;PHL=0;CSC=1;FLP=4;
        HHP=8;HVP=6;PHP=0;</D2T2Sensors>
        <EmbosserStatus>Ready</EmbosserStatus>
        <EmbosserSensors>PR.CE=0,XY.YP=1,XY.XH=0,XY.YH=0,CL.CD=0,XY.YE=0,EB.TC=0,
        EB.BC=0,EB.CE=1,EB.WH=0,CS.TD=1,CS.AD=1,EX.EB=1,EX.RB=0,EX.T1=1,EX.YT=1,
        XY.H1=0,XY.H2=0,EX.T2=2,EX.T3=2,EX.T4=2,RB.PC=2,RB.RD=2,TP.CD=2,EX.FP=2,EB.TR=2
        </EmbosserSensors>
        <EncoderStatus>Encoder Received Card</EncoderStatus>
        <EncoderStatusCode>0,0</EncoderStatusCode>
        <EncoderSensors>FS=0,MS=0,RS=1,AS=1</EncoderSensors>
    </DeviceStatus>
```

```
</PrintStatus>
```

ProcessSetProperty:

ProcessSetProperty - Example Request:

```
<?xml version="1.0" ?>
<DeviceSettings>
    <CommandInfo>
          <UniqueID>IntelliCAT</UniqueID>
          <CommandID>4</CommandID>
    </CommandInfo>
    <SmartCard>
          <Type>Contact</Type>
          <CommandID>1</CommandID>
          <PropertyID>AE</PropertyID>
          <PropertyType>3</PropertyType>
          <Data>01</Data>
    </SmartCard>
</DeviceSettings>
```

ProcessSetProperty - Example Response:

```
<?xml version="1.0" ?>
<PrintStatus>
    <CommandStatus>
        <CommandID>4</CommandID>
        <SmartCard>
            <Type>Contact</Type>
            <CommandID>1</CommandID>
            <PropertyID>AE</PropertyID>
            <PropertyType>3</PropertyType>
            <ResultCode>0</ResultCode>
            <ReturnCode>0</ReturnCode>
            <Data />
            <DataLength>0</DataLength>
        </SmartCard>
        <ReturnCode>0</ReturnCode>
        <ReturnMsg>OK</ReturnMsg>
        <UniqueID>IntelliCAT</UniqueID>
        <Location>Encoder</Location>
    </CommandStatus>
    <DeviceInformation>
        <EncoderModel>IntelliStripe 380</EncoderModel>
        <EncoderVersion>16051306B08</EncoderVersion>
        <EjectBin>OK</EjectBin>
        <RejectBin>OK</RejectBin>
    </DeviceInformation>
    <DeviceStatus>
        <D2T2Display>PRINTER STATUS SMART ENCODING CANCEL PAUSE</D2T2Display>
        <D2T2Status>Printing</D2T2Status>
        <D2T2Sensors>CIF=0;CFL=0;CPP=0;CMG=0;HFS=0;SCO=0;PHL=0;CSC=1;FLP=4;
        HHP=8;HVP=6;PHP=0;</D2T2Sensors>
        <EmbosserStatus>Ready</EmbosserStatus>
        <EmbosserSensors>PR.CE=0,XY.YP=1,XY.XH=0,XY.YH=0,CL.CD=0,XY.YE=0,EB.TC=0,
        EB.BC=0,EB.CE=1,EB.WH=0,CS.TD=1,CS.AD=1,EX.EB=1,EX.RB=0,EX.T1=1,EX.YT=1,
        XY.H1=0,XY.H2=0,EX.T2=2,EX.T3=2,EX.T4=2,RB.PC=2,RB.RD=2,TP.CD=2,EX.FP=2,EB.TR=2
        </EmbosserSensors>
        <EncoderStatus>Encoder Received Card</EncoderStatus>
        <EncoderStatusCode>0,0</EncoderStatusCode>
        <EncoderSensors>FS=0,MS=0,RS=1,AS=1</EncoderSensors>
```

```
    </DeviceStatus>
</PrintStatus>
```

ProcessSmartCardAPDU:

ProcessSmartCardAPDU - Example Request:

```
<?xml version="1.0" ?>
<DeviceSettings>
    <CommandInfo>
        <UniqueID>IntelliCAT</UniqueID>
        <CommandID>4</CommandID>
    </CommandInfo>
    <SmartCard>
        <Type>Contact</Type>
        <CommandID>85</CommandID>
        <Data>00A40000023F00</Data>
    </SmartCard>
</DeviceSettings>
```

ProcessSmartCardAPDU - Example Response

```
<?xml version="1.0" ?>
<PrintStatus>
    <CommandStatus>
        <CommandID>4</CommandID>
        <SmartCard>
            <Type>Contact</Type>
            <CommandID>85</CommandID>
            <PropertyID />
            <PropertyType />
            <ResultCode>0</ResultCode>
            <ReturnCode>0</ReturnCode>
            <Data>6985</Data>
            <DataLength>4</DataLength>
        </SmartCard>
        <ReturnCode>0</ReturnCode>
        <ReturnMsg>OK</ReturnMsg>
        <UniqueID>IntelliCAT</UniqueID>
        <Location>Encoder</Location>
    </CommandStatus>
    <DeviceInformation>
        <EncoderModel>IntelliStripe 380</EncoderModel>
        <EncoderVersion>16051306B08</EncoderVersion>
        <EjectBin>OK</EjectBin>
        <RejectBin>OK</RejectBin>
    </DeviceInformation>
    <DeviceStatus>
        <D2T2Display>PRINTER STATUS SMART ENCODING CANCEL PAUSE</D2T2Display>
        <D2T2Status>Printing</D2T2Status>
        <D2T2Sensors>CIF=0;CFL=0;CPP=0;CMG=0;HFS=0;SCO=0;PHL=0;CSC=1;FLP=4;
        HHP=8;HVP=6;PHP=0;</D2T2Sensors>
        <EmbosserStatus>Ready</EmbosserStatus>
        <EmbosserSensors>PR.CE=0,XY.YP=1,XY.XH=0,XY.YH=0,CL.CD=0,XY.YE=0,EB.TC=0,
        EB.BC=0,EB.CE=1,EB.WH=0,CS.TD=1,CS.AD=1,EX.EB=1,EX.RB=0,EX.T1=1,EX.YT=1,
        XY.H1=0,XY.H2=0,EX.T2=2,EX.T3=2,EX.T4=2,RB.PC=2,RB.RD=2,TP.CD=2,EX.FP=2,EB.TR=2
        </EmbosserSensors>
        <EncoderStatus>Encoder Received Card</EncoderStatus>
        <EncoderStatusCode>0,0</EncoderStatusCode>
        <EncoderSensors>FS=0,MS=0,RS=1,AS=1</EncoderSensors>
    </DeviceStatus>
```

```
</PrintStatus>
```

ProcessSmartCardPowerOff:

ProcessSmartCardPowerOff - Example Request:

```
<?xml version="1.0" ?>
<DeviceSettings>
    <CommandInfo>
        <UniqueID>IntelliCAT</UniqueID>
        <CommandID>4</CommandID>
    </CommandInfo>
    <SmartCard>
        <Type>Contact</Type>
        <CommandID>81</CommandID>
    </SmartCard>
</DeviceSettings>
```

ProcessSmartCardPowerOff - Example Response:

```
<?xml version="1.0" ?>
<PrintStatus>
    <CommandStatus>
        <CommandID>4</CommandID>
        <SmartCard>
            <Type>Contact</Type>
            <CommandID>81</CommandID>
            <PropertyID />
            <PropertyType />
            <ResultCode>0</ResultCode>
            <ReturnCode>0</ReturnCode>
            <Data />
            <DataLength>0</DataLength>
        </SmartCard>
        <ReturnCode>0</ReturnCode>
        <ReturnMsg>OK</ReturnMsg>
        <UniqueID>IntelliCAT</UniqueID>
        <Location>Encoder</Location>
    </CommandStatus>
    <DeviceInformation>
        <EncoderModel>IntelliStripe 380</EncoderModel>
        <EncoderVersion>16051306B08</EncoderVersion>
        <EjectBin>OK</EjectBin>
        <RejectBin>OK</RejectBin>
    </DeviceInformation>
    <DeviceStatus>
        <D2T2Display>PRINTER STATUS SMART ENCODING CANCEL PAUSE</D2T2Display>
        <D2T2Status>Printing</D2T2Status>
        <D2T2Sensors>CIF=0;CFL=0;CPP=0;CMG=0;HFS=0;SCO=0;PHL=0;CSC=1;FLP=4;
        HHP=8;HVP=6;PHP=0;</D2T2Sensors>
        <EmbosserStatus>Ready</EmbosserStatus>
        <EmbosserSensors>PR.CE=0,XY.YP=1,XY.XH=0,XY.YH=0,CL.CD=0,XY.YE=0,EB.TC=0,
        EB.BC=0,EB.CE=1,EB.WH=0,CS.TD=1,CS.AD=1,EX.EB=1,EX.RB=0,EX.T1=1,EX.YT=1,
        XY.H1=0,XY.H2=0,EX.T2=2,EX.T3=2,EX.T4=2,RB.PC=2,RB.RD=2,TP.CD=2,EX.FP=2,EB.TR=2
        </EmbosserSensors>
        <EncoderStatus>Encoder Received Card</EncoderStatus>
        <EncoderStatusCode>0,0</EncoderStatusCode>
       <EncoderSensors>FS=0,MS=0,RS=1,AS=1</EncoderSensors>
    </DeviceStatus>
</PrintStatus>
```

ProcessSmartCardPowerOn:

ProcessSmartCardPowerOn - Example Request:

```
<?xml version="1.0" ?>
<DeviceSettings>
    <CommandInfo>
          <UniqueID>IntelliCAT</UniqueID>
          <CommandID>4</CommandID>
    </CommandInfo>
    <SmartCard>
          <Type>Contact</Type>
          <CommandID>80</CommandID>
    </SmartCard>
</DeviceSettings>
```

ProcessSmartCardPowerOn - Example Response:

```
<?xml version="1.0" ?>
<PrintStatus>
    <CommandStatus>
        <CommandID>4</CommandID>
        <SmartCard>
            <Type>Contact</Type>
            <CommandID>80</CommandID>
            <PropertyID />
            <PropertyType />
            <ResultCode>0</ResultCode>
            <ReturnCode>0</ReturnCode>
            <Data>3BBF1800C02031705553544152434F5320533231204390009B</Data>
            <DataLength>50</DataLength>
        </SmartCard>
        <ReturnCode>0</ReturnCode>
        <ReturnMsg>OK</ReturnMsg>
        <UniqueID>IntelliCAT</UniqueID>
        <Location>Encoder</Location>
    </CommandStatus>
    <DeviceInformation>
        <EncoderModel>IntelliStripe 380</EncoderModel>
        <EncoderVersion>16051306B08</EncoderVersion>
        <EjectBin>OK</EjectBin>
        <RejectBin>OK</RejectBin>
    </DeviceInformation>
    <DeviceStatus>
        <D2T2Display>PRINTER STATUS SMART ENCODING CANCEL PAUSE</D2T2Display>
        <D2T2Status>Printing</D2T2Status>
        <D2T2Sensors>CIF=0;CFL=0;CPP=0;CMG=0;HFS=0;SCO=0;PHL=0;CSC=1;FLP=4;
        HHP=8;HVP=6;PHP=0;</D2T2Sensors>
        <EmbosserStatus>Ready</EmbosserStatus>
        <EmbosserSensors>PR.CE=0,XY.YP=1,XY.XH=0,XY.YH=0,CL.CD=0,XY.YE=0,EB.TC=0,
        EB.BC=0,EB.CE=1,EB.WH=0,CS.TD=1,CS.AD=1,EX.EB=1,EX.RB=0,EX.T1=1,EX.YT=1,
        XY.H1=0,XY.H2=0,EX.T2=2,EX.T3=2,EX.T4=2,RB.PC=2,RB.RD=2,TP.CD=2,EX.FP=2,EB.TR=2
        </EmbosserSensors>
        <EncoderStatus>Encoder Received Card</EncoderStatus>
        <EncoderStatusCode>0,0</EncoderStatusCode>
        <EncoderSensors>FS=0,MS=0,RS=1,AS=1</EncoderSensors>
    </DeviceStatus>
</PrintStatus>
```

ProcessSmartCardPropertyCondRpt:

ProcessSmartCardProperytCondRpt - Example Request:

```
<?xml version="1.0" ?>
<DeviceSettings>
    <CommandInfo>
        <UniqueID>IntelliCAT</UniqueID>
        <CommandID>4</CommandID>
    </CommandInfo>
    <SmartCard>
        <Type>Contact</Type>
        <CommandID>0</CommandID>
        <PropertyID>0</PropertyID>
        <PropertyType>4</PropertyType>
    </SmartCard>
</DeviceSettings>
```

ProcessSmartCardProperytCondRpt - Example Response:

```
<?xml version="1.0" ?>
<PrintStatus>
    <CommandStatus>
        <CommandID>4</CommandID>
        <SmartCard>
            <Type>Contact</Type>
            <CommandID>0</CommandID>
            <PropertyID>0</PropertyID>
            <PropertyType>4</PropertyType>
            <ResultCode>0</ResultCode>
            <ReturnCode>0</ReturnCode>
            <Data>808001000000000C10B60033EF49E00</Data>
            <DataLength>30</DataLength>
        </SmartCard>
       <ReturnCode>0</ReturnCode>
       <ReturnMsg>OK</ReturnMsg>
       <UniqueID>IntelliCAT</UniqueID>
       <Location>Encoder</Location>
    </CommandStatus>
    <DeviceInformation>
        <EncoderModel>IntelliStripe 380</EncoderModel>
        <EncoderVersion>16051306B08</EncoderVersion>
        <EjectBin>OK</EjectBin>
        <RejectBin>OK</RejectBin>
    </DeviceInformation>
    <DeviceStatus>
        <D2T2Display>PRINTER STATUS SMART ENCODING CANCEL PAUSE</D2T2Display>
        <D2T2Status>Printing</D2T2Status>
        <D2T2Sensors>CIF=0;CFL=0;CPP=0;CMG=0;HFS=0;SCO=0;PHL=0;CSC=1;FLP=4;
        HHP=8;HVP=6;PHP=0;</D2T2Sensors>
        <EmbosserStatus>Ready</EmbosserStatus>
        <EmbosserSensors>PR.CE=0,XY.YP=1,XY.XH=0,XY.YH=0,CL.CD=0,XY.YE=0,EB.TC=0,
        EB.BC=0,EB.CE=1,EB.WH=0,CS.TD=1,CS.AD=1,EX.EB=1,EX.RB=0,EX.T1=1,EX.YT=1,
        XY.H1=0,XY.H2=0,EX.T2=2,EX.T3=2,EX.T4=2,RB.PC=2,RB.RD=2,TP.CD=2,EX.FP=2,EB.TR=2
        </EmbosserSensors>
       <EncoderStatus>Encoder Received Card</EncoderStatus>
       <EncoderStatusCode>0,0</EncoderStatusCode>
       <EncoderSensors>FS=0,MS=0,RS=1,AS=1</EncoderSensors>
    </DeviceStatus>
</PrintStatus>
```

ProcessSmartCardWarmReset:

ProcessSmartCardWarmReset - Example Request:

```
<?xml version="1.0" ?>
<DeviceSettings>
    <CommandInfo>
        <UniqueID>IntelliCAT</UniqueID>
        <CommandID>4</CommandID>
    </CommandInfo>
    <SmartCard>
        <Type>Contact</Type>
        <CommandID>82</CommandID>
    </SmartCard>
</DeviceSettings>
```

ProcessSmartCardWarmReset - Example Response

```
<?xml version="1.0" ?>
<PrintStatus>
    <CommandStatus>
        <CommandID>4</CommandID>
        <SmartCard>
            <Type>Contact</Type>
            <CommandID>82</CommandID>
            <PropertyID />
            <PropertyType />
            <ResultCode>0</ResultCode>
            <ReturnCode>0</ReturnCode>
            <Data>3BBF1800C02031705553544152434F5320533231204390009B</Data>
            <DataLength>50</DataLength>
        </SmartCard>
        <ReturnCode>0</ReturnCode>
        <ReturnMsg>OK</ReturnMsg>
        <UniqueID>IntelliCAT</UniqueID>
        <Location>Encoder</Location>
    </CommandStatus>
    <DeviceInformation>
        <EncoderModel>IntelliStripe 380</EncoderModel>
        <EncoderVersion>16051306B08</EncoderVersion>
        <EjectBin>OK</EjectBin>
        <RejectBin>OK</RejectBin>
    </DeviceInformation>
    <DeviceStatus>
        <D2T2Display>PRINTER STATUS SMART ENCODING CANCEL PAUSE</D2T2Display>
        <D2T2Status>Printing</D2T2Status>
        <D2T2Sensors>CIF=0;CFL=0;CPP=0;CMG=0;HFS=0;SCO=0;PHL=0;CSC=1;FLP=4;
        HHP=8;HVP=6;PHP=0;</D2T2Sensors>
        <EmbosserStatus>Ready</EmbosserStatus>
        <EmbosserSensors>PR.CE=0,XY.YP=1,XY.XH=0,XY.YH=0,CL.CD=0,XY.YE=0,EB.TC=0,
        EB.BC=0,EB.CE=1,EB.WH=0,CS.TD=1,CS.AD=1,EX.EB=1,EX.RB=0,EX.T1=1,EX.YT=1,
        XY.H1=0,XY.H2=0,EX.T2=2,EX.T3=2,EX.T4=2,RB.PC=2,RB.RD=2,TP.CD=2,EX.FP=2,EB.TR=2
        </EmbosserSensors>
       <EncoderStatus>Encoder Received Card</EncoderStatus>
       <EncoderStatusCode>0,0</EncoderStatusCode>
       <EncoderSensors>FS=0,MS=0,RS=1,AS=1</EncoderSensors>
    </DeviceStatus>
</PrintStatus>
```

APPENDIX A. XML CODE SAMPLES

The following are code samples of XML key-value pair commands that are representative of a typical "job file" sent to the EC1000. Operations related to card-pick, encoding, printing, and embossing are provided. Used in conjunction with Key-Value definitions in sections 3 and 5 of this document, these code samples should assist a programmer in understanding the basic use and sequence of the key-value pair commands that are utilized when sending a job file to the device.

Any process involving printing to a card requires the addition of the D2T2 section with the following keys as the minimum:

```
<D2T2>
       <SplitRibbon>FALSE</SplitRibbon>
       <Dither>Optimized For Graphics</Dither>
       <ColorManagement>System Color Management</ColorManagement>
       <RotateFront>FALSE</RotateFront>
       <RotateBack>FALSE</RotateBack>
       <Orientation>LANDSCAPE</Orientation>
       <PrintTextEnabled>FALSE</PrintTextEnabled>
       <PrintImageEnabled>FALSE</PrintImageEnabled>
</D2T2>
```

This usage table describes which sections are required to perform a particular process.

Usage Table for Sections

Operation	Required Sections
Print Image & Text	CommandInfo, D2T2
Emboss Data	CommandInfo, D2T2, Emboss
Encode Data	CommandInfo, MagEncode
Encode Print Emboss	CommandInfo, D2T2, Emboss, MagEncode
Smart Card	CommandInfo, SmartCard

EXAMPLE: PRINT IMAGE AND TEXT

The following are the minimum required keys to process a single line of printed text and a single image:

```
<?xml version="1.0"?>
<DeviceSettings>
     <CommandInfo>
            <UniqueID>0123456</UniqueID>
            <CardHopperIndex>0</CardHopperIndex>
            <CommandID>7</CommandID>
     </CommandInfo>
     <D2T2>
            <SplitRibbon>FALSE</SplitRibbon>
            <Dither>Optimized For Graphics</Dither>
            <ColorManagement>System Color Management</ColorManagement>
            <RotateFront>FALSE</RotateFront>
            <RotateBack>FALSE</RotateBack>
            <Orientation>LANDSCAPE</Orientation>

            <PrintImageEnabled>TRUE</PrintImageEnabled>
            <NumImages>1</NumImages>
            <ImageDataEncodeType1>64</ImageDataEncodeType1>
            <ImageSide1>FRONT</ImageSide1>
            <ImageScaleXYWH1>0,0,336,212</ImageScaleXYWH1>
            <ImageSize1>408278</ImageSize1>
            <ImageType1>JPG</ImageType1>
            <ImageData1>Base64 Data Removed From Display</ImageData1>

            <PrintTextEnabled>TRUE</PrintTextEnabled>
            <NumTextDataLines>1</NumTextDataLines>
            <TextData1>Test Print Front</TextData1>
            <TextPositionXY1>10,10</TextPositionXY1>
            <TextSide1>FRONT</TextSide1>
            <TextFontName1>Courier New</TextFontName1>
            <TextFontSize1>18</TextFontSize1>
            <TextFontColor1>000000</TextFontColor1>
            <TextFontBold1>TRUE</TextFontBold1>
            <TextFontItalic1>TRUE</TextFontItalic1>
            <TextFontStrikeThru1>TRUE</TextFontStrikeThru1>
     </D2T2>
</DeviceSettings>
```

EXAMPLE: PRINT TEXT ONLY

The following are the minimum required keys to process a single line of printed text:

```
<?xml version="1.0"?>
<DeviceSettings>
     <CommandInfo>
            <UniqueID>0123456</UniqueID>
            <CardHopperIndex>0</CardHopperIndex>
            <CommandID>7</CommandID>
     </CommandInfo>
     <D2T2>
            <SplitRibbon>FALSE</SplitRibbon>
            <Dither>Optimized For Graphics</Dither>
            <ColorManagement>System Color Management</ColorManagement>
            <RotateFront>FALSE</RotateFront>
            <RotateBack>FALSE</RotateBack>
            <Orientation>LANDSCAPE</Orientation>
            <PrintImageEnabled>FALSE</PrintImageEnabled>

            <PrintTextEnabled>TRUE</PrintTextEnabled>
            <NumTextDataLines>1</NumTextDataLines>
            <TextData1>Test Print Front</TextData1>
            <TextPositionXY1>10,10</TextPositionXY1>
            <TextSide1>FRONT</TextSide1>
            <TextFontName1>Courier New</TextFontName1>
            <TextFontSize1>18</TextFontSize1>
            <TextFontColor1>000000</TextFontColor1>
            <TextFontBold1>TRUE</TextFontBold1>
            <TextFontItalic1>TRUE</TextFontItalic1>
            <TextFontStrikeThru1>FALSE</TextFontStrikeThru1>
     </D2T2>
</DeviceSettings>
```

EXAMPLE: PRINT TEXT ONLY BASE 64 ENCODING

The following are the minimum required keys to process a single line of printed Base 64 encoded text:

```
<?xml version="1.0"?>
<DeviceSettings>
     <CommandInfo>
            <UniqueID>0123456</UniqueID>
            <CardHopperIndex>0</CardHopperIndex>
            <CommandID>7</CommandID>
     </CommandInfo>
     <D2T2>
            <SplitRibbon>FALSE</SplitRibbon>
            <Dither>Optimized For Graphics</Dither>
            <ColorManagement>System Color Management</ColorManagement>
            <RotateFront>FALSE</RotateFront>
            <RotateBack>FALSE</RotateBack>
            <Orientation>LANDSCAPE</Orientation>
            <PrintImageEnabled>FALSE</PrintImageEnabled>

            <PrintTextEnabled>TRUE</PrintTextEnabled>
            <NumTextDataLines>1</NumTextDataLines>

            <TextData1>VGVzdCBQcmludCBGcm9udA==</TextData1>
            <TextDataEncoded1>TRUE</TextDataEncoded1>

            <TextPositionXY1>10,10</TextPositionXY1>
            <TextSide1>FRONT</TextSide1>
            <TextFontName1>Courier New</TextFontName1>
            <TextFontSize1>18</TextFontSize1>
            <TextFontColor1>000000</TextFontColor1>
            <TextFontBold1>TRUE</TextFontBold1>
            <TextFontItalic1>TRUE</TextFontItalic1>
            <TextFontStrikeThru1>FALSE</TextFontStrikeThru1>
     </D2T2>
</DeviceSettings>
```

EXAMPLE: PRINT TEXT & EMBOSS WITH KEYLINE

The following are the minimum required keys to process a single keyline print text and emboss:

```
<?xml version="1.0"?>
<DeviceSettings>
     <CommandInfo>
            <UniqueID>0123456</UniqueID>
            <CardHopperIndex>0</CardHopperIndex>
            <CommandID>7</CommandID>
     </CommandInfo>
     <D2T2>
            <SplitRibbon>FALSE</SplitRibbon>
            <Dither>Optimized For Graphics</Dither>
            <ColorManagement>System Color Management</ColorManagement>
            <RotateFront>FALSE</RotateFront>
            <RotateBack>FALSE</RotateBack>
            <Orientation>LANDSCAPE</Orientation>
            <PrintImageEnabled>FALSE</PrintImageEnabled>
            <PrintTextEnabled>TRUE</PrintTextEnabled>
            <NumTextDataLines>1</NumTextDataLines>

            <TextData1>1234</TextData1>
            <TextPositionXY1>10,10</TextPositionXY1>
            <TextSide1>FRONT</TextSide1>
            <TextFontName1>Courier New</TextFontName1>
            <TextFontSize1>18</TextFontSize1>
            <TextFontColor1>000000</TextFontColor1>
            <TextFontBold1>TRUE</TextFontBold1>
            <TextFontItalic1>TRUE</TextFontItalic1>
            <TextFontStrikeThru1>FALSE</TextFontStrikeThru1>

            <TextKeyLine1>TRUE</TextKeyLine1>
            <TextKeyLineColor1>FF0000</TextKeyLineColor1>
            <TextKeyLineTop1>-1.0</TextKeyLineTop1>
            <TextKeyLineLeft1>-1.0</TextKeyLineLeft1>
            <TextKeyLineBottom1>1.0</TextKeyLineBottom1>
            <TextKeyLineRight1>1.0</TextKeyLineRight1>
     </D2T2>
     <Emboss>
            <EmbossEnabled>TRUE</EmbossEnabled>
            <EmbossDataLines>1</EmbossDataLines>
            <PrintEmboss1>TRUE</PrintEmboss1>
            <EmbossData1>1234 5678 9012 1234</EmbossData1>
            <EmbossDataPosXY1>40,120</EmbossDataPosXY1>
            <EmbossFontID1>1</EmbossFontID1>
            <EmbossFontColor1>000000</EmbossFontColor1>

            <EmbossKeyLine1>TRUE</EmbossKeyLine1>
            <EmbossKeyLineColor1>FF0000</EmbossKeyLineColor1>
            <EmbossKeyLineTop1>-1.0</EmbossKeyLineTop1>
            <EmbossKeyLineLeft1>-1.0</EmbossKeyLineLeft1>
            <EmbossKeyLineBottom1>1.0</EmbossKeyLineBottom1>
            <EmbossKeyLineRight1>1.0</EmbossKeyLineRight1>
     </Emboss>
```

```
</DeviceSettings>
```

EXAMPLE: EMBOSS DATA

The following are the minimum required keys to process four lines of embossed data:

```
<?xml version="1.0"?>
<DeviceSettings>
     <CommandInfo>
            <UniqueID>0123456</UniqueID>
            <CardHopperIndex>0</CardHopperIndex>
            <CommandID>7</CommandID>
     </CommandInfo>
     <D2T2>
            <SplitRibbon>FALSE</SplitRibbon>
            <Dither>Optimized For Graphics</Dither>
            <ColorManagement>System Color Management</ColorManagement>
            <RotateFront>FALSE</RotateFront>
            <RotateBack>FALSE</RotateBack>
            <Orientation>LANDSCAPE</Orientation>
            <PrintTextEnabled>FALSE</PrintTextEnabled>
            <PrintImageEnabled>FALSE</PrintImageEnabled>
     </D2T2>
     <Emboss>
            <EmbossEnabled>TRUE</EmbossEnabled>
            <EmbossDataLines>4</EmbossDataLines>
            <PrintEmboss1>TRUE</PrintEmboss1>
            <EmbossData1>1234 5678 9012 1234</EmbossData1>
            <EmbossDataPosXY1>40,120</EmbossDataPosXY1>
            <EmbossFontID1>1</EmbossFontID1>
            <EmbossFontColor1>000000</EmbossFontColor1>
            <PrintEmboss2>TRUE</PrintEmboss2>
            <EmbossData2>11/06</EmbossData2>
            <EmbossDataPosXY2>180,147</EmbossDataPosXY2>
            <EmbossFontID2>2</EmbossFontID2>
            <EmbossFontColor2>000000</EmbossFontColor2>
            <PrintEmboss3>TRUE</PrintEmboss3>
            <EmbossData3>JOHN Q SMITH</EmbossData3>
            <EmbossDataPosXY3>30,180</EmbossDataPosXY3>
            <EmbossFontID3>2</EmbossFontID3>
            <EmbossFontColor3>000000</EmbossFontColor3>
            <PrintEmboss4>TRUE</PrintEmboss4>
            <EmbossData4>1234 122</EmbossData4>
            <EmbossDataPosXY4>206,92</EmbossDataPosXY4>
            <EmbossFontID4>4</EmbossFontID4>
            <EmbossFontColor4>000000</EmbossFontColor4>
     </Emboss>
</DeviceSettings>
```

EXAMPLE: ENCODE DATA

The following are the minimum required keys to process magnetic stripe encoded data:

```
<?xml version="1.0"?>
<DeviceSettings>
     <CommandInfo>
            <UniqueID>0123456</UniqueID>
            <CardHopperIndex>0</CardHopperIndex>
            <CommandID>7</CommandID>
     </CommandInfo>
     <MagEncode>
            <MagEncodeEnable>TRUE</MagEncodeEnable>
            <NumTracks>2</NumTracks>
            <Coercivity>HIGH</Coercivity>
            <StripeRetry>3</StripeRetry>
            <TrackData1>B1234567890121234^SMITH/JOHN Q^06111010498400997000000</TrackData1>
            <TrackData2>1234567890121234=06111010498499700000</TrackData2>
            <TrackData3></TrackData3>
     </MagEncode>
</DeviceSettings>
```

EXAMPLE: ENCODE PRINT EMBOSS REQUEST

The following are the minimum required keys to process a card for encode, print, and emboss in a single request:

```
<?xml version="1.0" ?>
<DeviceSettings>
    <CommandInfo>
        <UniqueID>0123456</UniqueID>
        <CardHopperIndex>0</CardHopperIndex>
         <CommandID>7</CommandID>
    </CommandInfo>
    <D2T2>
        <SplitRibbon>FALSE</SplitRibbon>
        <Dither>Optimized For Graphics</Dither>
        <ColorManagement>System Color Management</ColorManagement>
        <RotateFront>FALSE</RotateFront>
        <RotateBack>FALSE</RotateBack>
        <Orientation>LANDSCAPE</Orientation>
        <PrintTextEnabled>TRUE</PrintTextEnabled>
        <PrintImageEnabled>TRUE</PrintImageEnabled>
        <NumImages>2</NumImages>
        <ImageDataEncodeType1>64</ImageDataEncodeType1>
        <ImageSide1>FRONT</ImageSide1>
        <ImageScaleXYWH1>0,0,336,212</ImageScaleXYWH1>
        <ImageSize1>408278</ImageSize1>
        <ImageType1>JPG</ImageType1>
        <ImageData1>/9j/4(ABBREVIATED...)AAQSkZJRgABAgEBLAEsAAD/4Qt5RXhpZgA</ImageData1>
        <ImageDataEncodeType2>64</ImageDataEncodeType2>
        <ImageSide2>BACK</ImageSide2>
        <ImageScaleXYWH2>0,0,336,212</ImageScaleXYWH2>
        <ImageSize2>194614</ImageSize2>
        <ImageType2>JPG</ImageType2>
        <ImageData2>/9j/4A(ABBREVIATED…)AQSkZJRgABAgEBLAEsAAD/4Q</ImageData2>
        <NumTextDataLines>2</NumTextDataLines>
        <TextData1>Test Print Front</TextData1>
        <TextPositionXY1>10,10</TextPositionXY1>
        <TextSide1>FRONT</TextSide1>
        <TextFontName1>Courier New</TextFontName1>
        <TextFontSize1>18</TextFontSize1>
        <TextFontColor1>000000</TextFontColor1>
        <TextFontBold1>TRUE</TextFontBold1>
        <TextFontItalic1>TRUE</TextFontItalic1>
        <TextFontStrikeThru1>TRUE</TextFontStrikeThru1>
        <TextData2>Test Print Back</TextData2>
        <TextPositionXY2>50,100</TextPositionXY2>
        <TextSide2>BACK</TextSide2>
        <TextFontName2>Courier New</TextFontName2>
        <TextFontSize2>9</TextFontSize2>
        <TextFontColor2>000000</TextFontColor2>
        <TextFontBold2>TRUE</TextFontBold2>
        <TextFontItalic2>TRUE</TextFontItalic2>
        <TextFontStrikeThru2>FALSE</TextFontStrikeThru2>
    </D2T2>
    <Emboss>
        <EmbossEnabled>TRUE</EmbossEnabled>
```

```
        <EmbossDataLines>4</EmbossDataLines>
        <PrintEmboss1>TRUE</PrintEmboss1>
        <EmbossData1>1234 5678 9012 1234</EmbossData1>
        <EmbossDataPosXY1>30,120</EmbossDataPosXY1>
        <EmbossFontID1>1</EmbossFontID1>
        <EmbossFontColor1>000000</EmbossFontColor1>
        <PrintEmboss2>TRUE</PrintEmboss2>
        <EmbossData2>JOHN Q SMITH</EmbossData2>
        <EmbossDataPosXY2>30,177</EmbossDataPosXY2>
        <EmbossFontID2>2</EmbossFontID2>
        <EmbossFontColor2>000000</EmbossFontColor2>
        <PrintEmboss3>TRUE</PrintEmboss3>
        <EmbossData3>11/06</EmbossData3>
        <EmbossDataPosXY3>180,147</EmbossDataPosXY3>
        <EmbossFontID3>2</EmbossFontID3>
        <EmbossFontColor3>000000</EmbossFontColor3>
        <PrintEmboss4>TRUE</PrintEmboss4>
        <EmbossData4>1234 122</EmbossData4>
        <EmbossDataPosXY4>206,92</EmbossDataPosXY4>
        <EmbossFontID4>4</EmbossFontID4>
        <EmbossFontColor4>000000</EmbossFontColor4>
    </Emboss>
    <MagEncode>
        <MagEncodeEnable>TRUE</MagEncodeEnable>
        <NumTracks>3</NumTracks>
        <Coercivity>HIGH</Coercivity>
        <StripeRetry>3</StripeRetry>
        <TrackData1>B1234567890121234^SMITH/JOHN Q^06111010498400997000000</TrackData1>
        <TrackBPI1>210</TrackBPI1>
        <TrackBPC1>7</TrackBPC1>
        <TrackData2>1234567890121234=06111010498499700000</TrackData2>
        <TrackBPI2>75</TrackBPI2>
        <TrackBPC2>5</TrackBPC2>
        <TrackData3 />
        <TrackBPI3>210</TrackBPI3>
        <TrackBPC3>5</TrackBPC3>
    </MagEncode>
</DeviceSettings>
```

EXAMPLE: ENCODE PRINT EMBOSS RESPONSE

```
<?xml version="1.0" ?>
<PrintStatus>
    <CommandStatus>
        <CommandID>7</CommandID>
        <ReturnCode>0</ReturnCode>
        <ReturnMsg>OK</ReturnMsg>
        <UniqueID>0123456</UniqueID>
        <MagnePrint>0100087846BA06AAA47D970BF546475977CC15F2873CC671A8F8A7582EF0
        28F6277BD60D0075F0161C5F91A39FD58F5052A11E06E2DD</MagnePrint>
        <Track1Data>%B1234567890121234^SMITH/JOHN Q^06111010498400997000000?</Track1Data>
        <Track2Data>;1234567890121234=06111010498499700000?</Track2Data>
        <Track3Data />
        <Location>Eject BIN</Location>
    </CommandStatus>
    <DeviceInformation>
        <D2T2Serial>A1234567</D2T2Serial>
        <D2T2RibbonPartNumber>86231</D2T2RibbonPartNumber>
        <D2T2RibbonPercentage>93%</D2T2RibbonPercentage>
        <ControllerModel>Express Card 1000 MLB</ControllerModel>
        <ControllerVersion>33050804A03</ControllerVersion>
        <EncoderModel>IntelliStripe 380</EncoderModel>
        <EncoderVersion>16051306B08</EncoderVersion>
        <EjectBin>OK</EjectBin>
        <RejectBin>OK</RejectBin>
    </DeviceInformation>
    <DeviceStatus>
        <D2T2Display>READY H1 MENU H2</D2T2Display>
        <D2T2Status>Ready</D2T2Status>
        <D2T2Sensors>CIF=0;CFL=0;CPP=0;CMG=0;HFS=1;SCO=0;PHL=0;CSC=0;FLP=2;
        HHP=8;HVP=7;PHP=0;</D2T2Sensors>
        <EmbosserStatus>Ready</EmbosserStatus>
        <EmbosserSensors>PR.CE=0,XY.YP=1,XY.XH=1,XY.YH=0,XY.H1=1,XY.H2=1,CL.CD=0,
        XY.YE=0,EB.TC=1,EB.BC=1,EB.CE=1,EB.WH=0,CS.TD=1,CS.AD=1,EX.EB=0,EX.RB=0,
        EX.T1=1,EX.YT=1</EmbosserSensors>
        <EncoderStatus>Ready</EncoderStatus>
       <EncoderStatusCode>0,0</EncoderStatusCode>
       <EncoderSensors>FS=0,MS=0,RS=0,AS=0</EncoderSensors>
    </DeviceStatus>
</PrintStatus>
```

EXAMPLE: ENCRYPTED PRINT TEXT

The following are the minimum required keys to process a single line of encrypted print text:

```
<?xml version="1.0"?>
<DeviceSettings>
     <CommandInfo>
            <UniqueID>0123456</UniqueID>
            <CardHopperIndex>0</CardHopperIndex>
            <CommandID>7</CommandID>
     </CommandInfo>
     <D2T2>
            <SplitRibbon>FALSE</SplitRibbon>
            <Dither>Optimized For Graphics</Dither>
            <ColorManagement>System Color Management</ColorManagement>
            <RotateFront>FALSE</RotateFront>
            <RotateBack>FALSE</RotateBack>
            <Orientation>LANDSCAPE</Orientation>
            <PrintImageEnabled>FALSE</PrintImageEnabled>
            <PrintTextEnabled>TRUE</PrintTextEnabled>
            <NumTextDataLines>1</NumTextDataLines>

            <EncryptedText>TRUE</EncryptedText>
            <TextData1>bMfQMUtMKDY=</TextData1>
            <TextDataLength1>4</TextDataLength1>
            <TextPositionXY1>40,120</TextPositionXY1>
            <TextSide1>FRONT</TextSide1>
            <TextFontName1>Courier New</TextFontName1>
            <TextFontSize1>18</TextFontSize1>
            <TextFontColor1>000000</TextFontColor1>
            <TextFontBold1>TRUE</TextFontBold1>
            <TextFontItalic1>TRUE</TextFontItalic1>
            <TextFontStrikeThru1>FALSE</TextFontStrikeThru1>
     </D2T2>
</DeviceSettings>
```

The following is the process to encrypt BASE 64 data lines:

- Encode the clear text data in UTF-8 format.
- Encrypt the clear text data with Data Transport Key using:
  - Triple DES
  - 16-byte Key
  - Initialization Vector “0000000000000000”
  - Cipher mode of CBC
  - Padding mode of Zero
- BASE 64 encode the encrypted text data.
- Assign the resulting data to the applicable TextData[n] tag.
- Assign the length of the clear text data to the applicable TextDataLength[n] tag.

- Set the <EncryptedText> to TRUE for the D2T2 section.

C# .Net Sample code:

```
// Encode the cleart text data.
 string cleartextdata = "1234";
 string DataTransportKeyHex = "11111111222222223333333344444444";
 string cipher_mode = "CBC";
 Encoding buffEncoder = System.Text.Encoding.UTF8;
 byte[] buffer = buffEncoder.GetBytes(cleartextdata);

// Encrypt clear text data with Data Transport Key using CBC mode.
 byte[] encBuffer = TDES_EncByte(DataTransportKeyHex, buffer, cipher_mode, PAD_ZERO);

// BASE 64 encode the encrypted text data.
 string resultData = System.Convert.ToBase64String(encBuffer);

--------------------------------------------------------------------------------
Inputs:
Encoding UTF-8    = 31323334(hex)
Encrypted         = 6CC7D0314B4C2836(hex)

Output:
Base 64 Encoded   = bMfQMUtMKDY=
```

Visual Basic .Net Sample code:

```
‘ Encode the cleart text data.
 Dim cleartextdata as String = "1234"
 Dim DataTransportKeyHex as String = "11111111222222223333333344444444"
 Dim cipher_mode as String = "CBC"
 Dim buffEncoder As Encoding = System.Text.Encoding.UTF8

 Dim buffer as Byte()
 Byte = buffEncoder.GetBytes(cleartextdata)

‘ Encrypt clear text data with Data Transport Key using CBC mode.
 Dim encBuffer as Byte()
 encBuffer = TDES_EncByte(DataTransportKeyHex, buffer, cipher_mode, PAD_ZERO)

‘ BASE 64 encode the encrypted text data.
 Dim resultData as String
 resultData = System.Convert.ToBase64String(encBuffer)

--------------------------------------------------------------------------------
Inputs:
Encoding UTF-8    = 31323334(hex)
Encrypted         = 6CC7D0314B4C2836(hex)

Output:
Base 64 Encoded   = bMfQMUtMKDY=
```

EXAMPLE: ENCRYPTED EMBOSS & ENCODE

The following are the minimum required keys to process a single line of encrypted emboss and magstripe encode data:

```
<?xml version="1.0"?>
<DeviceSettings>
     <CommandInfo>
            <UniqueID>0123456</UniqueID>
            <CardHopperIndex>0</CardHopperIndex>
            <CommandID>7</CommandID>
     </CommandInfo>
     <D2T2>
            <SplitRibbon>FALSE</SplitRibbon>
            <Dither>Optimized For Graphics</Dither>
            <ColorManagement>System Color Management</ColorManagement>
            <RotateFront>FALSE</RotateFront>
            <RotateBack>FALSE</RotateBack>
            <Orientation>LANDSCAPE</Orientation>
            <PrintTextEnabled>FALSE</PrintTextEnabled>
            <PrintImageEnabled>FALSE</PrintImageEnabled>
     </D2T2>
     <Emboss>
            <Encrypted>TRUE</Encrypted>
            <EmbossEnabled>TRUE</EmbossEnabled>
            <EmbossDataLines>1</EmbossDataLines>
            <PrintEmboss1>TRUE</PrintEmboss1>
            <EmbossData1>bMfQMUtMKDY=</EmbossData1>
            <EmbossDataLength1>4</EmbossDataLength1>
            <EmbossDataPosXY1>40,120</EmbossDataPosXY1>
            <EmbossFontID1>1</EmbossFontID1>
            <EmbossFontColor1>000000</EmbossFontColor1>
     </Emboss>
     <MagEncode>
            <Encrypted>TRUE</Encrypted>
            <MagEncodeEnable>TRUE</MagEncodeEnable>
            <NumTracks>2</NumTracks>
            <Coercivity>HIGH</Coercivity>
            <StripeRetry>3</StripeRetry>
            <TrackData1>
            DHiqnEbKcEbVVH+tvN7n6QWTjZsw4Fpv+2IUBHwhSrTv/bSmOwBQS23lP+9xITB6Zp3QdvN2/G8=
            </TrackData1>
            <TrackData2>
            3B4Hz/KOLLi2B+w/3jnNFFL27EcraT0w9IOlB9lK7LTGy3Hyd5SsgQ==
            </TrackData2>
            <TrackData3></TrackData3>
            <TrackData1Length>53</TrackData1Length>
            <TrackData2Length>37</TrackData2Length>
            <TrackData3Length>0</TrackData3Length>
     </MagEncode>
</DeviceSettings>
```

The following is the process to encrypt BASE 64 data lines.

- Encode the clear text data in UTF-8 format.
- Encrypt the clear text data with Data Transport Key using:
  - Triple DES
  - 16-byte Key
  - Initialization Vector “0000000000000000”
  - Cipher mode of CBC
  - Padding mode of Zero
- BASE 64 encode the encrypted emboss and magencode data.
- Assign the resulting data to the applicable EmbossData[n] and TrackData[n] tags.
- Assign the length of the clear text data to the applicable EmbossDataLength[n] and TrackData[n]Length tags.
- Set the <Encrypted> to TRUE for the Emboss and MagEncode sections.

C# .Net Sample code:

```
// Encode the cleart text data.
 string cleartextdata = "1234567890121234=06111010498499700000";
 string DataTransportKeyHex = "11111111222222223333333344444444";
 string cipher_mode = "CBC";
 Encoding buffEncoder = System.Text.Encoding.UTF8;
 byte[] buffer = buffEncoder.GetBytes(cleartextdata);

// Encrypt clear text data with Data Transport Key using CBC mode.
 byte[] encBuffer = TDES_EncByte(DataTransportKeyHex, buffer, cipher_mode, PAD_ZERO);

// BASE 64 encode the encrypted text data.
 string resultData = System.Convert.ToBase64String(encBuffer);

--------------------------------------------------------------------------------
Inputs:
Encoding UTF-8    =
313233343536373839303132313233343D303631313130313034393834393937303030303030(hex)
Encrypted         =
DC1E07CFF28E2CB8B607EC3FDE39CD1452F6EC472B693D30F483A507D94AECB4C6CB71F27794AC81(hex)

Output:
Base 64 Encoded   =
3B4Hz/KOLLi2B+w/3jnNFFL27EcraT0w9IOlB9lK7LTGy3Hyd5SsgQ==
```

Visual Basic .Net Sample code:

```
‘ Encode the cleart text data.
 Dim cleartextdata as String = "1234567890121234=06111010498499700000"
 Dim DataTransportKeyHex as String = "11111111222222223333333344444444"
 Dim cipher_mode as String = "CBC"
 Dim buffEncoder As Encoding = System.Text.Encoding.UTF8

 Dim buffer as Byte()
 Byte = buffEncoder.GetBytes(cleartextdata)

‘ Encrypt clear text data with Data Transport Key using CBC mode.
 Dim encBuffer as Byte()
 encBuffer = TDES_EncByte(DataTransportKeyHex, buffer, cipher_mode, PAD_ZERO)

‘ BASE 64 encode the encrypted text data.
 Dim resultData as String
 resultData = System.Convert.ToBase64String(encBuffer)

--------------------------------------------------------------------------------
Inputs:
Encoding UTF-8    =
313233343536373839303132313233343D303631313130313034393834393937303030303030(hex)
Encrypted         =
DC1E07CFF28E2CB8B607EC3FDE39CD1452F6EC472B693D30F483A507D94AECB4C6CB71F27794AC81(hex)

Output:
Base 64 Encoded   =
3B4Hz/KOLLi2B+w/3jnNFFL27EcraT0w9IOlB9lK7LTGy3Hyd5SsgQ==
```

EXAMPLE: QUEUE PROCESS SEQUENCE

1. Send a job to QueueJob.aspx.
2. Receive the queue status with a Unique ID.
3. The host application polls for the status using the Unique ID provided from step number 2.
4. Receive the queue status.

1. Send Queue Job XML:

Send an XML post to web page QueueJob.aspx.

```
<?xml version="1.0"?>
<DeviceSettings>
     <CommandInfo>
            <UniqueID>Sample1</UniqueID>
            <OwnerID>Central Bank</OwnerID>
            <CardHopperIndex>0</CardHopperIndex>
            <CommandID>7</CommandID>
            <Description>Entry Card</Description>
     </CommandInfo>
     <MagEncode>
            <MagEncodeEnable>TRUE</MagEncodeEnable>
            <NumTracks>3</NumTracks>
            <Coercivity>HIGH</Coercivity>
            <StripeRetry>3</StripeRetry>
            <TrackData1>B1234567890121234^SMITH/JOHN Q^06111010498400997000000</TrackData1>
            <TrackData2>1234567890121234=06111010498499700000</TrackData2>
            <TrackData3></TrackData3>
     </MagEncode>
</DeviceSettings>
```

2. Queue Job response XML:

```
<?xml version="1.0"?>
<QueueStatus>
       <CommandStatus>
            <UniqueID>Sample1_7f0c8daf-f9ab-46c1-8aa6-5cb57bd72ca6</UniqueID>
            <OwnerID>Central Bank</OwnerID>
            <ReturnCode>0</ReturnCode>
            <ReturnMsg>OK</ReturnMsg>
            </CommandStatus>
</QueueStatus>
```

3. Queue Polling:

Post a request to web page QueueStatus.aspx with a query string of:

QueueStatus.aspx?UniqueID=[*UniqueID*]

```
https://192.168.10.100/QueueStatus.aspx?UniqueID= Sample1_7f0c8daf-f9ab-46c1-8aa6-
5cb57bd72ca6
```

4. Queue Polling Response: - Pending

```
<?xml version="1.0"?>
<QueueStatus>
     <CommandStatus>
            <UniqueID>0123456_1796e62a-0c38-4f8b-b78e-54579227a735</UniqueID>
            <OwnerID>Central Bank</OwnerID>
            <CommandID>7</CommandID>
            <CardType>0</CardType>
            <ReturnCode>68</ReturnCode>
            <ReturnMsg>Pending</ReturnMsg>
            <Submitted>6/8/2009 8:26:24 AM</Submitted>
            <Completed></Completed>
            <Priority>1</Priority>
            <Data></Data>
            <Description>Entry Card</Description>
</CommandStatus>
</QueueStatus>
```

Queue Polling Response: - Completed

```
<?xml version="1.0"?>
<QueueStatus>
     <CommandStatus>
            <UniqueID> Sample1_7f0c8daf-f9ab-46c1-8aa6-5cb57bd72ca6 </UniqueID>
            <OwnerID>Central Bank</OwnerID>
            <CommandID>7</CommandID>
            <CardType>0</CardType>
            <ReturnCode>0</ReturnCode>
            <ReturnMsg>OK</ReturnMsg>
            <Submitted>6/3/2009 3:00:37 PM</Submitted>
            <Completed>6/3/2009 3:00:58 PM</Completed>
            <Priority></Priority>
            <Data>
               <MagnePrint>0100087846BA06AAA47D970BF546475977CC15F2873CC671A8F8A7582EF0
               28F6277BD60D0075F0161C5F91A39FD58F5052A11E06E2DD
               </MagnePrint>
            </Data>
            <Description>Entry Card</Description>
     </CommandStatus>
</QueueStatus>
```

EXAMPLE: QUEUE NOTIFICATION SEQUENCE

1. Send a Move To Encoder Notification command including a Response URL.
2. Receive the queue status with a Unique ID.
3. Wait for a Notification to where the Unique ID notified matches the Unique ID received from step number 2. Host application may continue the personalization job after a match is found.

1. Move to Encoder with Notification Queue Job request XML:

Send an XML post to web page QueueJob.aspx.

```
<?xml version="1.0"?>
<DeviceSettings>
     <CommandInfo>
            <Description>Gift Card</Description>
            <OwnerID>Central Bank</OwnerID>
            <UniqueID>Sample1</UniqueID>
            <CommandID>14</CommandID>
            <CardHopperIndex>0</CardHopperIndex>
            <ResponseURL>https://Client1/GetNotification.aspx</ResponseURL>
     </CommandInfo>
</DeviceSettings>
```

2. Move to Encoder Queue Job response XML:

```
<?xml version="1.0"?>
<QueueStatus>
       <CommandStatus>
            <UniqueID>Sample1_7f0c8daf-f9ab-46c1-8aa6-5cb57bd72ca6</UniqueID>
            <OwnerID>Central Bank</OwnerID>
            <ReturnCode>0</ReturnCode>
            <ReturnMsg>OK</ReturnMsg>
            </CommandStatus>
</QueueStatus>
```

3. Queue Notification: - Success

```
<?xml version="1.0" ?>
<QueueStatus>
     <CommandStatus>
            <UniqueID>Sample1_7f0c8daf-f9ab-46c1-8aa6-5cb57bd72ca6</UniqueID>
            <CommandID>14</CommandID>
            <ReturnCode>109</ReturnCode>
            <ReturnMsg>Ready</ReturnMsg>
     </CommandStatus>
</QueueStatus>
```

EXAMPLE: QUEUE STATUS SEQUENCE

To view the status of a queued job, a GET command is posted to the QueueStatus web page. The Queue web page interface then returns an XML response of the status contained in the following key paths:

- QueueStatus//CommandStatus//UniqueID
- QueueStatus//CommandStatus//OwnerID
- QueueStatus//CommandStatus//CommandID
- QueueStatus//CommandStatus//CardType
- QueueStatus//CommandStatus//ReturnCode
- QueueStatus//CommandStatus//ReturnMsg
- QueueStatus//CommandStatus//Submitted
- QueueStatus//CommandStatus//Completed
- QueueStatus//CommandStatus//Priority
- QueueStatus//CommandStatus//Data
- QueueStatus//CommandStatus//Description

The Status of a queue initially begins with Pending. Once in process, the status is changed to Busy. After set to Busy, the status may change to any status associated with card production.

1. Queue Status Request:
Post a request to web page QueueStatus.aspx with a query string of:

QueueStatus.aspx?UniqueID=[*UniqueID*]

```
https://ec1000/QueueStatus.aspx?UniqueID=123_d023a200-5c90-447b-8154-803a8de771b5
```

Example Queue Status Response : - Pending

```
<?xml version="1.0" ?>
<QueueStatus>
     <CommandStatus>
            <UniqueID>Sample1_7f0c8daf-f9ab-46c1-8aa6-5cb57bd72ca6</UniqueID>
            <OwnerID>Central Bank</OwnerID>
            <CommandID>7</CommandID>
            <CardType>0</CardType>
            <ReturnCode>68</ReturnCode>
            <ReturnMsg>Pending</ReturnMsg>
            <Submitted>1/01/2009 8:07:53 AM</Submitted>
            <Completed />
            <Priority>1</Priority>
            <Data />
            <Description>Gift Card</Description>
     </CommandStatus>
```

```
 </QueueStatus>
```

Example Queue Status Response: - Busy

```
<?xml version="1.0"?>
<QueueStatus>
     <CommandStatus>
            <UniqueID>Sample1_7f0c8daf-f9ab-46c1-8aa6-5cb57bd72ca6</UniqueID>
            <OwnerID>Central Bank</OwnerID>
            <CommandID>7</CommandID>
            <CardType>0</CardType>
            <ReturnCode>3</ReturnCode>
            <ReturnMsg>Busy</ReturnMsg>
            <Submitted>1/01/2009 8:07:53 AM</Submitted>
            <Completed />
            <Priority>1</Priority>
            <Data />
            <Description>Gift Card</Description>
     </CommandStatus>
</QueueStatus>
```

Example Queue Status Response: - Complete

```
<?xml version="1.0"?>
<QueueStatus>
     <CommandStatus>
            <UniqueID>Sample1_7f0c8daf-f9ab-46c1-8aa6-5cb57bd72ca6</UniqueID>
            <OwnerID>Central Bank</OwnerID>
            <CommandID>7</CommandID>
            <CardType>0</CardType>
            <ReturnCode>0</ReturnCode>
            <ReturnMsg>OK</ReturnMsg>
            <Submitted>1/01/2009 8:07:53 AM</Submitted>
            <Completed>1/01/2009 8:09:53 AM</Completed>
            <Priority />
            <Data>
                  <MagnePrint>010108041BBBBADA420C51C685B4F7B702AEF9F789590C87B
                   83DC795E6015AC4CC4F67991DBFCC6181D72D42686C67090C54E2DE49BB
                   </MagnePrint>
            </Data>
            <Description>Gift Card</Description>
     </CommandStatus>
</QueueStatus>
```

EXAMPLE: QUEUE COUNT SEQUENCE

To attain the number of queued jobs, a GET command is posted to the QueueCount web page. The Queue web page interface then returns an XML response of the status contained in the following key paths:

- QueueCount//Count
- QueueCount//CommandStatus//ReturnCode
- QueueCount//CommandStatus//ReturnMsg

1. Post a request to the QueueCount web page.
2. Receive the response.

1. Queue Count Request :

```
https://ec1000/QueueCount.aspx
```

2. Queue Count Response :

```
<?xml version="1.0" ?>
<QueueCount>
          <Count>2</Count>
          <CommandStatus>
                    <ReturnCode>0</ReturnCode>
                    <ReturnMsg>OK</ReturnMsg>
          </CommandStatus>
</QueueCount>
```

EXAMPLE: QUEUE DELETE SEQUENCE

To delete a queued job, a GET command is posted to the QueueDel web page. The Queue web page interface then returns an XML response of the status contained in the following key paths:

- QueueStatus//CommandStatus//UniqueID
- QueueStatus//CommandStatus//OwnerID
- QueueStatus//CommandStatus//ReturnCode
- QueueStatus//CommandStatus//ReturnMsg

1. Post a request to the QueueDel web page.
2. Receive the response.

1. Queue Deletion Request:

Post a request to web page QueueDel.aspx with a query string of:

QueueDel.aspx?UniqueID=[*UniqueID*]

```
https://ec1000/QueueDel.aspx?UniqueID=Sample1_7f0c8daf-f9ab-46c1-8aa6-5cb57bd72ca6
```

2. Queue Deletion Response: - Success

```
<?xml version="1.0" ?>
<QueueDelete>
     <CommandStatus>
            <UniqueID>Sample1_7f0c8daf-f9ab-46c1-8aa6-5cb57bd72ca6</UniqueID>
            <OwnerID></OwnerID>
            <ReturnCode>0</ReturnCode>
            <ReturnMsg>OK</ReturnMsg>
     </CommandStatus>
 </QueueDelete>
```

3. Queue Deletion Response: - No Job Found

```
<?xml version="1.0" ?>
<QueueDelete>
     <CommandStatus>
            <UniqueID>Sample1_7f0c8daf-f9ab-46c1-8aa6-5cb57bd72ca6</UniqueID>
            <OwnerID></OwnerID>
            <ReturnCode>2</ReturnCode>
            <ReturnMsg>No Job Found</ReturnMsg>
     </CommandStatus>
 </QueueDelete>
```

APPENDIX B. RETURN CODES AND MESSAGES

The following are the return codes and job state messages returned by the device:

RETURN CODES AND MESSAGES

RET CODE NUMBER	RETURN CODE	MSG CODE
0	JOBERR_SUCCESS	
1	JOBERR_DONE	
2	JOBERR_NO_JOB	
3	JOBERR_BUSY	
4	JOBERR_MGR_MOVE_CARD_TO_ENCODER	
5	JOBERR_MGR_TIMEOUT_MOVE_CARD_TO_ENCODER	
6	JOBERR_MGR_MOVE_CARD_TO_D2T2	
7	JOBERR_MGR_TIMEOUT_MOVE_CARD_TO_D2T2	
8	JOBERR_MGR_MOVE_CARD_TO_EMBOSSER	
9	JOBERR_MGR_TIMEOUT_MOVE_CARD_TO_EMBOSSER	
10	JOBERR_MGR_SCANNING_CARD	
11	JOBERR_MGR_ALLOC_RESOURCE	
12	JOBERR_MGR_ALLOC_RESPONSE	
13	JOBERR_MGR_ALLOC_D2T2	
14	JOBERR_MGR_ALLOC_CONTROLLER	
15	JOBERR_MGR_PARSING_XML	
16	JOBERR_MGR_MISSING_XML_DATA	
17	JOBERR_MGR_SETTING_RESOURCES	
18	JOBERR_MGR_INIT	
19	JOBERR_MGR_INIT_CONTROLLER	
20	JOBERR_MGR_INIT_PRINTER	
21	JOBERR_MGR_INIT_ENCODER	
22	JOBERR_MGR_INIT_SCANNER	
23	JOBERR_MGR_INIT_EMBOSSER	
24	JOBERR_MGR_MOVETO_TIPPER	
25	JOBERR_MGR_TIP_CARD	
26	JOBERR_MGR_EJECT_CARD	
27	JOBERR_MGR_SELECTHOPPER	
28	JOBERR_MGR_EJECTBIN_FULL	
29	JOBERR_MGR_REJECTBIN_FULL	
30	JOBERR_MGR_GET_CONFIGURATION	
31	JOBERR_MGR_NO_CARD_TO_PROCESS	
32	JOBERR_D2T2_CREATE_IMAGE	
33	JOBERR_D2T2_CREATE_TEXT	
34	JOBERR_D2T2_PRINTCARD	
35	JOBERR_D2T2_CARD_JAM	
36	JOBERR_D2T2_EJECT_CARD	
37	JOBERR_D2T2_TIMEOUT	
38	JOBERR_D2T2_SELECT_HOPPER	
39	JOBERR_D2T2_HOPPER_EMPTY	
40	JOBERR_D2T2_INVALID_HOPPER_STATE	

RET CODE NUMBER	RETURN CODE	MSG CODE
41	JOBERR_D2T2_CARDWAIT_TIMEOUT	
42	JOBERR_ENCODER_CARDJAM	
43	JOBERR_ENCODER_OPEN	
44	JOBERR_ENCODER_EJECT	
45	JOBERR_ENCODER_ENCODE	
46	JOBERR_ENCODER_ENCODE_TIMEOUT	
47	JOBERR_ENCODER_READCARD	
48	JOBERR_ENCODER_SET_MSR_DIRECTION	
49	JOBERR_ENCODER_SET_MSR_COERCIVITY	
50	JOBERR_ENCODER_SET_RETRY_PROPERTY	
51	JOBERR_ENCODER_SET_AUTOCONSUME_PROPERTY	
52	JOBERR_ENCODER_ENCODE_PREPARE	
53	JOBERR_ENCODER_ENCODE_VERIFY	
54	JOBERR_ENCODER_CLEAR_TRACKS	
55	JOBERR_XY_MOVETO_HOME	
56	JOBERR_XY_MOVETO_PRINTERHANDOFF	
57	JOBERR_XY_MOVETO_EMBOSSERHANDOFF	
58	JOBERR_XY_MOVETO_CARDHOME	
59	JOBERR_XY_MOVETO_EJECTHANDOFF	
60	JOBERR_XY_CLAMPCARD	
61	JOBERR_PRINTER_BUSY	
62	JOBERR_SCANNER_SCAN	
63	JOBERR_EMBOSSSER_EMBOSS	
64	JOBERR_INVALID_CONFIG	
65	JOBERR_CONTROLLER_OPEN	
66	JOBERR_CONTROLLER_GET_MODELNUMBER	
67	JOBERR_CONTROLLER_GET_SENSORS	
68	JOBERR_PENDING	
69	JOBERR_USERABORT	
70	JOBERR_ENCODER_SC_FAILED	
71	JOBERR_MGR_INIT_PROCESS	
72	JOBERR_ENCODER_NOCARD	
73	JOBERR_ENCODER_CLEAR_ENCODE	
74	JOBERR_CONTROLLER_TOPDOOROPEN	
75	JOBERR_CONTROLLER_ACCESSDOOROPEN	
76	JOBERR_CONTROLLER_DOORSTATUS_FAILED	
77	JOBERR_MGR_ALLBIN_FULL	
78	JOBERR_ENCODER_SC_INVALID_DATA	
79	JOBERR_MGR_NOT_ACTIVATED	
80	JOBERR_MGR_NOT_QUEUED	
81	JOBERR_MGR_INVALID_COMMAND	
82	JOBERR_MGR_QUEUE_NOT_SUPPORTED	
83	JOBERR_MGR_OFFLINE	
84	JOBERR_MGR_METHOD_NOT_SUPPORTED	
85	JOBERR_GENERIC	
86	JOBERR_UNKNOWN	
87	JOBERR_TIMEOUT	

RET CODE NUMBER	RETURN CODE	MSG CODE
88	JOBERR_INVALID_XML	
89	JOBERR_ENCODER_MOVECARD	
90	JOBERR_MGR_INVALID_PARAMS	
91	JOBERR_ENCODER_BUSY	
92	JOBERR_ENCODER_SC_INITIALIZE_PERSO	
93	JOBERR_ENCODER_SC_SELECT	
94	JOBERR_ENCODER_SC_GETKEYDATA	
95	JOBERR_ENCODER_SC_VALIDATEKEYDATA	
96	JOBERR_ENCODER_SC_DERIVEKDPERSO	
97	JOBERR_ENCODER_SC_VALIDATETK1DATA	
98	JOBERR_ENCODER_SC_VALIDATETK2DATA	
99	JOBERR_ENCODER_SC_DERIVEKDCVC3	
100	JOBERR_ENCODER_SC_STOREDATA	
101	JOBERR_MGR_ENCRYPTION_NOT_SUPPORTED	
102	JOBERR_MGR_SC_INVALID_TYPE	
103	JOBERR_MGR_SC_METHOD_NOT_SUPPORTED	
104	JOBERR_ENCODER_COMMUNICATION	
105	JOBERR_CONTROLLER_GET_VERSION	
106	JOBERR_MGR_NOT_AUTHORIZED	
107	JOBERR_MGR_QUEUE_DELETE_FAILED	
108	JOBERR_MGR_INVALID_QUEUE_ID	
109	JOBERR_READY	
110	JOBERR_MGR_IMAGE_ENCRYPTION_NOT_SUPPORTED	
111	JOBERR_MGR_ENCRYPTION_REQUIRED	
112	JOBERR_MGR_TRANSPORT_KEY_NOT_LOADED	
113	JOBERR_MGR_ENCODE_NOT_ENABLED	
114	JOBERR_MGR_EMBOSS_NOT_ENABLED	
115	JOBERR_MGR_REAR_INDENT_NOT_ENABLED	
116	JOBERR_MGR_FRONT_INDENT_NOT_ENABLED	
117	JOBERR_MGR_PRINT_TEXT_NOT_ENABLED	
118	JOBERR_MGR_PRINT_IMAGE_NOT_ENABLED	
119	JOBERR_MGR_CONTACTLESS_SMARTCARD_NOT_ENABLED	
120	JOBERR_MGR_CONTACT_SMARTCARD_NOT_ENABLED	

JOB STATES- MESSAGES

REF NUMBER	JOB STATES – MESSAGES	COMMENTS
X	"BUSY"	
X	"CARD ARRIVED AT SMARTCARD STATION"	
X	"CARD EJECTED FROM ENCODER"	
X	"EJECT CARD COMPLETED"	
X	"EJECTING CARD FROM ENCODER"	
X	"EJECTING CARD FROM PRINTER"	
X	"EJECTING CARD"	
X	"EMBOSS CARD COMPLETED"	
X	"EMBOSSING CARD"	
X	"ENCODE CARD COMPLETED"	
X	"ENCODE VERIFY COMPLETED"	
X	"ENCODER ERROR, CARD JAM"	
X	"ENCODER RECEIVED CARD"	
X	"ENCODING CARD"	
X	"ERROR DURING CARD TRANSFER"	
X	"ERROR EMBOSSING CARD"	
X	"ERROR ENCODING CARD"	
X	"ERROR PRINTING CARD"	
X	"ERROR PROCESSING JOB"	
X	"ERROR SCANNING CARD"	
X	"INITIALIZING CONTROLLER"	
X	"INITIALIZING EMBOSSER"	
X	"INITIALIZING ENCODER"	
X	"INITIALIZING PRINTER"	
X	"INITIALIZING SCANNER"	
X	"INITIALIZING XY TRANSPORT"	
X	"INITIALIZING"	
X	"MOVE TO CARD HOME POSITION COMPLETED"	
X	"MOVE TO EMBOSSER HANDOFF POSITION COMPLETED"	
X	"MOVE TO HOME POSITION COMPLETED"	
X	"MOVE TO PRINTER HANDOFF POSITION COMPLETED	
X	"MOVING CARD TO ENCODER"	
X	"MOVING CARD TO SMARTCARD STATION"	
X	"MOVING TO CARD HOME POSITION"	
X	"MOVING TO EMBOSSER HANDOFF POSITION"	
X	"MOVING TO HOME POSITION"	
X	"MOVING TO PRINTER HANDOFF POSITION"	
X	"MOVING XY TRANSPORT TO CARD RECEIVE POSITION"	
X	"PLEASE INSERT CARD"	
X	"PREPARING TO EJECT CARD"	
X	"PREPARING TO EMBOSS CARD"	
X	"PREPARING TO ENCODE CARD"	
X	"PREPARING TO PRINT"	
X	"PRINT CARD COMPLETED"	

X	**“PRINTING CARD”**
X	**“READ CARD COMPLETED”**
X	**“READING CARD”**
X	**“READY”**
X	**“SCAN CARD COMPLETED”**
X	**“SCANNING CARD”**
X	**“SELECTING HOPPER 1”**
X	**“SELECTING HOPPER 2”**
X	**“SELECTING HOPPER CURRENT”**
X	**“SELECTING HOPPER EXCEPTION”**
X	**“SELECTING HOPPER TOGGLE”**
X	**“UNKNOWN”**
X	**“VERIFYING ENCODE COMPLETED"**
X	**“VERIFYING ENCODE”**
X	**“WAITING FOR CARD TO ARRIVE AT SMARTCARD STATION”**
X	**“WAITING FOR CARD TO EJECT”**
X	**“WAITING FOR CARD"**
X	**“WAITING FOR ENCODE COMPLETION"**
X	**“WAITING TO ENCODE CARD"**
X	**“XY TRANSPORT HAS BEEN INITIALIZED”**
X	**“XY TRANSPORT READY TO RECEIVE CARD”**
X	**“TIMEOUT MOVING CARD TO SMART CARD STATION**