

# **USB MagneSafe V5**

## **COMMUNICATION REFERENCE MANUAL**

**PART NUMBER 99875475-1**

**JANUARY 2010**

**MAGTEK<sup>®</sup>**  
**REGISTERED TO ISO 9001:2008**  
1710 Apollo Court  
Seal Beach, CA 90740  
Phone: (562) 546-6400  
FAX: (562) 546-6301  
Technical Support: (651) 415-6800  
[\*\*www.magtek.com\*\*](http://www.magtek.com)

Information in this document is subject to change without notice. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of MagTek, Inc.

MagTek is a registered trademark of MagTek, Inc.  
MagnePrint is a registered trademark of MagTek, Inc.  
MagneSafe™ is a trademark of MagTek, Inc.  
Magensa™ is a trademark of MagTek, Inc.

USB (Universal Serial Bus) Specification is Copyright© 1998 by Compaq Computer Corporation, Intel Corporation, Microsoft Corporation, NEC Corporation.

Appendix A is taken from Universal Serial Bus HID Usage Tables, Version 1.12, Section 10, Keyboard/Keypad Page (0x07) ©1996-2005 USB Implementers' Forum

Appendix B is taken from Section 8.3 Report Format for Array Items, Device Class Definition for Human Interface Devices (HID) Version 1.11, ©1996-2001 USB Implementers' Forum, *hidcomments@usb.org*

#### REVISIONS

Rev Number	Date	Notes
1.01	08 Jan 10	Initial Release (previously part of 99875388)

## LIMITED WARRANTY

MagTek warrants that the products sold pursuant to this Agreement will perform in accordance with MagTek's published specifications. This warranty shall be provided only for a period of one year from the date of the shipment of the product from MagTek (the "Warranty Period"). This warranty shall apply only to the "Buyer" (the original purchaser, unless that entity resells the product as authorized by MagTek, in which event this warranty shall apply only to the first repurchaser).

During the Warranty Period, should this product fail to conform to MagTek's specifications, MagTek will, at its option, repair or replace this product at no additional charge except as set forth below. Repair parts and replacement products will be furnished on an exchange basis and will be either reconditioned or new. All replaced parts and products become the property of MagTek. This limited warranty does not include service to repair damage to the product resulting from accident, disaster, unreasonable use, misuse, abuse, negligence, or modification of the product not authorized by MagTek. MagTek reserves the right to examine the alleged defective goods to determine whether the warranty is applicable.

Without limiting the generality of the foregoing, MagTek specifically disclaims any liability or warranty for goods resold in other than MagTek's original packages, and for goods modified, altered, or treated without authorization by MagTek.

Service may be obtained by delivering the product during the warranty period to MagTek (1710 Apollo Court, Seal Beach, CA 90740). If this product is delivered by mail or by an equivalent shipping carrier, the customer agrees to insure the product or assume the risk of loss or damage in transit, to prepay shipping charges to the warranty service location, and to use the original shipping container or equivalent. MagTek will return the product, prepaid, via a three (3) day shipping service. A Return Material Authorization ("RMA") number must accompany all returns. Buyers may obtain an RMA number by contacting Technical Support at (888) 624-8350.

**EACH BUYER UNDERSTANDS THAT THIS MAGTEK PRODUCT IS OFFERED AS IS. MAGTEK MAKES NO OTHER WARRANTY, EXPRESS OR IMPLIED, AND MAGTEK DISCLAIMS ANY WARRANTY OF ANY OTHER KIND, INCLUDING ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.**

IF THIS PRODUCT DOES NOT CONFORM TO MAGTEK'S SPECIFICATIONS, THE SOLE REMEDY SHALL BE REPAIR OR REPLACEMENT AS PROVIDED ABOVE. MAGTEK'S LIABILITY, IF ANY, SHALL IN NO EVENT EXCEED THE TOTAL AMOUNT PAID TO MAGTEK UNDER THIS AGREEMENT. IN NO EVENT WILL MAGTEK BE LIABLE TO THE BUYER FOR ANY DAMAGES, INCLUDING ANY LOST PROFITS, LOST SAVINGS, OR OTHER INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OF, OR INABILITY TO USE, SUCH PRODUCT, EVEN IF MAGTEK HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, OR FOR ANY CLAIM BY ANY OTHER PARTY.

### **LIMITATION ON LIABILITY**

EXCEPT AS PROVIDED IN THE SECTIONS RELATING TO MAGTEK'S LIMITED WARRANTY, MAGTEK'S LIABILITY UNDER THIS AGREEMENT IS LIMITED TO THE CONTRACT PRICE OF THIS PRODUCT.

MAGTEK MAKES NO OTHER WARRANTIES WITH RESPECT TO THE PRODUCT, EXPRESSED OR IMPLIED, EXCEPT AS MAY BE STATED IN THIS AGREEMENT, AND MAGTEK DISCLAIMS ANY IMPLIED WARRANTY, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

MAGTEK SHALL NOT BE LIABLE FOR CONTINGENT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES TO PERSONS OR PROPERTY. MAGTEK FURTHER LIMITS ITS LIABILITY OF ANY KIND WITH RESPECT TO THE PRODUCT, INCLUDING ANY NEGLIGENCE ON ITS PART, TO THE CONTRACT PRICE FOR THE GOODS.

MAGTEK'S SOLE LIABILITY AND BUYER'S EXCLUSIVE REMEDIES ARE STATED IN THIS SECTION AND IN THE SECTION RELATING TO MAGTEK'S LIMITED WARRANTY.

### **FCC WARNING STATEMENT**

This equipment has been tested and was found to comply with the limits for a Class B digital device pursuant to Part 15 of FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a residential environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference with radio communications. However, there is no guarantee that interference will not occur in a particular installation.

### **FCC COMPLIANCE STATEMENT**

This device complies with Part 15 of the FCC Rules. Operation of this device is subject to the following two conditions: (1) this device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.

### **CANADIAN DOC STATEMENT**

This digital apparatus does not exceed the Class B limits for radio noise from digital apparatus set out in the Radio Interference Regulations of the Canadian Department of Communications.

Le présent appareil numérique n'émet pas de bruits radioélectriques dépassant les limites applicables aux appareils numériques de la classe B prescrites dans le Règlement sur le brouillage radioélectrique édicté par le ministère des Communications du Canada.

This Class B digital apparatus complies with Canadian ICES-003.

Cet appareil numérisé de la classe B est conforme à la norme NMB-003 du Canada.


### **CE STANDARDS**

Testing for compliance with CE requirements was performed by an independent laboratory. The unit under test was found compliant with standards established for Class B devices.

### **UL/CSA**

This product is recognized per Underwriter Laboratories and Canadian Underwriter Laboratories 1950.

### **RoHS STATEMENT**

When ordered as RoHS compliant, this product meets the Electrical and Electronic Equipment (EEE) Reduction of Hazardous Substances (RoHS) European Directive 2002/95/EC. The marking is clearly recognizable, either as written words like "Pb-free", "lead-free", or as another clear symbol (.

# TABLE OF CONTENTS

<b>SECTION 1. SECURITY</b> .....	<b>1</b>
SECURITY LEVEL 0 .....	1
SECURITY LEVEL 1 .....	1
SECURITY LEVEL 2 .....	1
SECURITY LEVEL 3 .....	1
SECURITY LEVEL 4 .....	2
COMMANDS AND SECURITY LEVELS.....	2
<b>SECTION 2. USB COMMUNICATIONS</b> .....	<b>3</b>
HID USAGES.....	3
MAGNETIC STRIPE READER USAGE PAGE (HID) .....	4
REPORT DESCRIPTOR (HID) .....	5
MAGNETIC STRIPE READER USAGE PAGE (KB).....	7
REPORT DESCRIPTOR (KB) .....	7
CARD DATA (HID) .....	8
Track 1 Decode Status.....	10
Track 2 Decode Status.....	10
Track 3 Decode Status.....	10
Track 1 Encrypted Data Length.....	10
Track 2 Encrypted Data Length.....	10
Track 3 Encrypted Data Length.....	11
Track 1 Absolute Data Length.....	11
Track 2 Absolute Data Length.....	11
Track 3 Absolute Data Length.....	11
Card Encode Type.....	11
Encrypted Track Data.....	11
Track 1 Encrypted Data.....	12
Track 2 Encrypted Data.....	12
Track 3 Encrypted Data.....	12
Card Status.....	12
MagnePrint Status .....	13
MagnePrint Data Length.....	13
MagnePrint Absolute Data Length .....	13
Encrypted MagnePrint Data .....	14
Device Serial Number.....	14
Track 1 Masked Data Length .....	14
Track 2 Masked Data Length .....	14
Track 3 Masked Data Length .....	14
Masked Track Data .....	14
Track 1 Masked Data .....	15
Track 2 Masked Data .....	15
Track 3 Masked Data .....	16
Encrypted Session ID .....	16
DUKPT Key Serial Number .....	16
Encryption Counter.....	16
MagneSafe Version Number .....	17
Hashed Track 2 Data.....	17
Clear Text CRC .....	17
Encrypted CRC.....	17
CARD DATA (KB).....	17
Reader Encryption Status.....	19
Format Code.....	20
PROGRAMMABLE CONFIGURATION OPTIONS .....	20
Low Level Communications.....	20

COMMANDS .....	21
PRIVILEGED COMMANDS.....	21
COMMAND NUMBER .....	22
DATA LENGTH.....	22
DATA .....	22
RESULT CODE .....	23
GET AND SET PROPERTY COMMANDS .....	23
Property ID .....	24
SOFTWARE ID PROPERTY .....	25
USB SERIAL NUM PROPERTY .....	26
POLLING INTERVAL PROPERTY.....	26
DEVICE SERIAL NUM PROPERTY .....	27
MAGNESAFE VERSION NUMBER .....	28
TRACK ID ENABLE PROPERTY.....	29
ISO TRACK MASK PROPERTY .....	29
AAMVA TRACK MASK PROPERTY.....	30
MAX PACKET SIZE PROPERTY (HID).....	31
ACTIVITY TIMEOUT PERIOD PROPERTY (WIRELESS READER ONLY) .....	32
STAY POWERED AFTER SWIPE PROPERTY (WIRELESS READER ONLY) .....	32
RF ADDRESS PROPERTY (WIRELESS READER ONLY).....	33
INTERFACE TYPE PROPERTY .....	34
TRACK DATA SEND FLAGS PROPERTY (KB).....	34
MP FLAGS PROPERTY (KB) .....	35
ACTIVE KEYMAP PROPERTY (KB).....	36
ASCII TO KEYPRESS CONVERSION TYPE PROPERTY (KB).....	37
CRC FLAG PROPERTY (KB) .....	38
SURESWIPE FLAG PROPERTY (KB).....	39
PRE CARD STRING PROPERTY (KB) .....	39
POST CARD STRING PROPERTY (KB) .....	40
PRE TRACK STRING PROPERTY (KB) .....	41
POST TRACK STRING PROPERTY (KB).....	41
TERMINATION STRING PROPERTY (KB) .....	42
FS PROPERTY (KB) .....	42
SS TK1 ISO ABA PROPERTY (KB).....	43
SS TK2 ISO ABA PROPERTY (KB).....	43
SS TK3 ISO ABA PROPERTY (KB).....	43
SS TK3 AAMVA PROPERTY (KB).....	44
SS TK2 7BITS PROPERTY (KB) .....	44
SS TK3 7BITS PROPERTY (KB) .....	44
ES PROPERTY (KB).....	45
FORMAT CODE PROPERTY (KB).....	45
ES TRACK 1 PROPERTY.....	46
ES TRACK 2 PROPERTY .....	46
ES TRACK 3 PROPERTY .....	46
SEND ENCRYPTION COUNTER (KB) .....	47
MASK OTHER CARDS .....	47
MSR DIRECTION PROPERTY (INSERT READER ONLY) .....	48
CARD INSERTED PROPERTY (INSERT READER ONLY).....	49
SEND CLEAR AAMVA CARD DATA PROPERTY .....	49
HID SURESWIPE FLAG PROPERTY (HID).....	50
SOFTWARE ID 2 PROPERTY (WIRELESS READER ONLY).....	51
RESET DEVICE COMMAND .....	51
GET KEYMAP ITEM COMMAND (KB) .....	52
SET KEYMAP ITEM COMMAND (KB).....	54
SAVE CUSTOM KEYMAP COMMAND (KB) .....	56
DUKPT OPERATION .....	56

Get DUKPT KSN and Counter .....	56
SET SESSION ID COMMAND .....	57
ACTIVATE AUTHENTICATED MODE COMMAND .....	57
ACTIVATION CHALLENGE REPLY COMMAND .....	59
DEACTIVATE AUTHENTICATED MODE COMMAND .....	60
GET READER STATE COMMAND .....	61
SET SECURITY LEVEL COMMAND .....	62
GET ENCRYPTION COUNTER COMMAND .....	63
POWER DOWN COMMAND (WIRELESS READER ONLY) .....	64
GET BATTERY STATUS COMMAND (WIRELESS READER ONLY) .....	64
<b>SECTION 3. DEMO PROGRAM .....</b>	<b>67</b>
INSTALLATION .....	67
OPERATION .....	67
SOURCE CODE .....	68
<b>APPENDIX A. KEYBOARD USAGE ID DEFINITIONS .....</b>	<b>69</b>
KEYBOARD/KEYPAD PAGE (0X07) .....	69
<b>APPENDIX B. MODIFIER BYTE DEFINITIONS .....</b>	<b>77</b>
<b>APPENDIX C. GUIDE ON DECRYPTING DATA .....</b>	<b>79</b>
<b>APPENDIX D. COMMAND EXAMPLES .....</b>	<b>81</b>
<b>APPENDIX E. IDENTIFYING ISO/ABA AND AAMVA CARDS .....</b>	<b>117</b>
ISO/ABA FINANCIAL CARDS .....	117
AAMVA DRIVER LICENSES .....	117





# SECTION 1. SECURITY

This reader is intended to be a secure reader. Security features include:

- Supplies 54 byte MagnePrint value
- Includes Device Serial Number
- Encrypts all track data and the MagnePrint value
- Provides clear text confirmation data including card holder's name, expiration date, and a portion of the PAN as part of the Masked Track Data
- Supports Mutual Authentication Mode for use with Magensa.net
- Offers selectable levels of Security

The reader supports five Security Levels. The Security Level can be increased by command, but can never be decreased:

## SECURITY LEVEL 0

Security Level 0 is a special case. It signifies that all DUKPT keys have been used. In this case the unit is at the end of its useful life. This level is set automatically by the reader when it runs out of DUKPT keys.

## SECURITY LEVEL 1

Security Level 1 is a factory test level only.

## SECURITY LEVEL 2

Security Level 2 is the least secure user mode. In this mode, keys are loaded but not used for most operations (only used to load new keys or move to Security Level 3 or 4). All other properties and commands are freely usable.

In the keyboard emulation mode, the reader sends data in the *SureSwipe* format as defined in MagTek document 99875206. The default SureSwipe mode can be changed to allow the reader to send data in the V5 format as described in this document, but the MagnePrint data will not be sent.

In the HID mode, the reader sends track data but does not send MagnePrint data.

## SECURITY LEVEL 3

Security Level 3 enables encryption of track data, MagnePrint data, and the Session ID. MagnePrint data is always included and it is always encrypted. The format for the data is detailed later in this document. At Security Level 3, many commands require security—most notably, the **Set Property** command. Transition to Security Level 4 requires security.

## SECURITY LEVEL 4

When the reader is at Security Level 4, a correctly executed Authentication Sequence is required before the reader will emit data from a card swipe. Correctly executing the Authentication Sequence also causes the Green LED to blink, alerting the user to the fact that the reader is being controlled by a Host with knowledge of the keys—that is, an Authentic Host.

Commands that require security must be sent with a four byte Message Authentication Code (MAC) appended to the end. The MAC is calculated as specified in ANSI X9.24 Part 1 – 2004, Annex A. Note that data supplied to the MAC algorithm should NOT be converted to the ASCII-Hex, rather it should be supplied in its raw binary form. The MAC key to be used is as specified in the same document (“Request PIN Entry 2” bullet 2). Calculating the MAC requires knowledge of the current DUKPT KSN, this could be retrieved using the **Get DUKPT KSN and Counter** command. For each command processed successfully, the DUKPT Key is advanced.

## COMMANDS AND SECURITY LEVELS

The following table shows how security levels affect the various commands. “Y” means the command can run. “N” means the command is prohibited. “S” means the command is protected (requires MACing). “X” means other (notes to follow).

Command	Level 2	Level 3	Level 4
Get Property	Y	Y	Y
Set Property	Y	S	S
Reset	Y	X*	X*
Get Keymap Item	Y	Y	Y
Set Keymap Item	Y	S	S
Save Keymap	Y	S	S
Get DUKPT SN and Counter	Y	Y	Y
Set Session ID	Y	Y	Y
Activate Authenticated Mode	N	Y	Y
Activation Challenge Reply	N	Y	Y
Deactivate Authenticated Mode	N	Y	Y
Get Reader State	Y	Y	Y
Set Security Level	S	S	S
Get Encryption Counter	Y	Y	Y
Power Down (wireless reader only)	Y	Y	Y
Get Battery Status (wireless reader only)	Y	Y	Y

- \* The Reset command has special behavior. When an Authentication sequence has failed, only a correctly MACd Reset command can be used to reset the reader. This is to prevent a dictionary attack on the keys and to minimize a denial of service attack.

## SECTION 2. USB COMMUNICATIONS

This reader conforms to the USB specification revision 1.1. This reader also conforms to the Human Interface Device (HID) class specification version 1.1. The reader communicates to the host either as a vendor-defined HID device or as a HID Keyboard Emulation device. (Refer to **Interface Type** Property for information on how to change modes.) The latest versions of the Windows operating system come with standard Windows USB drivers that will support both modes.

The reader has an adjustable endpoint descriptor polling interval value that can be set to any value in the range of 1ms to 255ms. This property can be used to speed up or slow down the card data transfer rate. The reader also has an adjustable serial number descriptor. More details about these properties can be found later in this document in the command section.

The reader will go into suspend mode when directed to do so by the host. The reader will wake up from suspend mode when directed to do so by the host. The reader does not support remote wakeup.

This is a full speed USB device. It is powered from the USB bus. The vendor ID (VID) is 0x0801. The product ID (PID) when in the HID mode is 0x0011 for the swipe reader and 0x0013 for the insert reader. The product ID (PID) when in the Keyboard Emulation mode is 0x0001 for both the swipe and insert reader. The wireless reader dongle uses the same PID as previously mentioned for the wired USB swipe reader. However, the wireless reader can also be directly connected to the host with a USB cable for updating firmware and charging the battery. When the wireless reader is directly connected, the PID is 0x0014.

Since there are two modes of operation, there are some properties and commands that are exclusive to one of the two modes. Where a property or command is unique, it will be identified with either **HID** or **KB**. Properties and commands that are common to both modes do not include any modifier.

### HID USAGES

HID devices send data in reports. Elements of data in a report are identified by unique identifiers called usages. The structure of the reader's reports and the reader's capabilities are reported to the host in a report descriptor. The host usually gets the report descriptor only once, right after the reader is plugged in. The report descriptor usages identify the reader's capabilities and report structures. Usages are four byte integers. The most significant two bytes are called the usage page and the least significant two bytes are called usage IDs. Usages that are related can share a common usage page. Usages can be standardized or they can be vendor defined. Standardized usages such as usages for mice and keyboards can be found in the HID Usage Tables document and can be downloaded free at [www.usb.org](http://www.usb.org). Vendor-defined usages must have a usage page in the range 0xFF00 – 0xFFFF. All usages for this reader use vendor-defined magnetic stripe reader usage page 0xFF00. The usage IDs for this reader are defined in the following tables. The usage types are also listed. These usage types are defined in the HID Usage Tables document.

**MAGNETIC STRIPE READER USAGE PAGE (HID)**

Magnetic Stripe Reader usage page 0xFF00:

Usage ID (Hex)	Usage Name	Usage Type	Report Type
1	Decoding reader device	Collection	None
20	Track 1 decode status	Data	Input
21	Track 2 decode status	Data	Input
22	Track 3 decode status	Data	Input
23	MagnePrint status	Data	Input
28	Track 1 encrypted data length	Data	Input
29	Track 2 encrypted data length	Data	Input
2A	Track 3 encrypted data length	Data	Input
2B	MagnePrint data length	Data	Input
30	Track 1 encrypted data	Data	Input
31	Track 2 encrypted data	Data	Input
32	Track 3 encrypted data	Data	Input
33	Encrypted MagnePrint data	Data	Input
38	Card encode type	Data	Input
39	Card status	Data	Input
40	Device Serial Number	Data	Input
42	Reader Encryption Status	Data	Input
46	DUKPT serial number/counter	Data	Input
47	Track 1 Masked data length	Data	Input
48	Track 2 Masked data length	Data	Input
49	Track 3 Masked data length	Data	Input
4A	Track 1 Masked Data	Data	Input
4B	Track 2 Masked Data	Data	Input
4C	Track 3 Masked Data	Data	Input
50	Encrypted Session ID	Data	Input
51	Track 1 Data Absolute Length	Data	Input
52	Track 2 Data Absolute Length	Data	Input
53	Track 3 Data Absolute Length	Data	Input
54	MagnePrint Data Absolute Length	Data	Input
55	Encryption Counter	Data	Input
56	MagneSafe Version Number	Data	Input
57	Hashed Track 2 Data	Data	Input
20	Command message	Data	Feature

## REPORT DESCRIPTOR (HID)

The Report Descriptor is made available to the hosting system during USB enumeration. The descriptor is shown here for completeness. Typically the hosting operating system will provide the ability to parse HID Reports based on the actual Report Descriptor, using the assigned Usage IDs. We strongly CAUTION the programmer to avoid depending on this specific structure as it may change in future versions. The Report Descriptor is structured as follows:

Item	Value (Hex)
Usage Page (Magnetic Stripe Reader)	06 00 FF
Usage (Decoding reader device)	09 01
Collection (Application)	A1 01
Logical Minimum (0)	15 00
Logical Maximum (255)	26 FF 00
Report Size (8)	75 08
Usage (Track 1 decode status)	09 20
Usage (Track 2 decode status)	09 21
Usage (Track 3 decode status)	09 22
Usage (Track 1 encrypted data length)	09 28
Usage (Track 2 encrypted data length)	09 29
Usage (Track 3 encrypted data length)	09 2A
Usage (Card encode type)	09 38
Report Count (7)	95 07
Input (Data, Variable, Absolute, Bit Field)	81 02
Usage (Track 1 encrypted data)	09 30
Report Count (112)	95 70
Input (Data, Variable, Absolute, Buffered Bytes)	82 02 01
Usage (Track 2 encrypted data)	09 31
Report Count (112)	95 70
Input (Data, Variable, Absolute, Buffered Bytes)	82 02 01
Usage (Track 3 encrypted data)	09 32
Report Count (112)	95 70
Input (Data, Variable, Absolute, Buffered Bytes)	82 02 01
Usage (Card status)	09 39
Report Count (1)	95 01
Input (Data, Variable, Absolute, Bit Field)	81 02
Report Size (32)	75 20
Usage (MagnePrint status)	09 23
Report Count (1)	95 01
Input (Data, Variable, Absolute, Bit Field)	81 02

<b>Item</b>	<b>Value (Hex)</b>
Report Size (8)	75 08
Usage (MagnePrint data length)	09 2B
Report Count (1)	95 01
Input (Data, Variable, Absolute, Bit Field)	81 02
Usage (MagnePrint data)	09 33
Report Count (128)	95 80
Input (Data, Variable, Absolute, Buffered Bytes)	82 02 01
Usage (Device serial number)	09 40
Report Count (16)	95 10
Input (Data, Variable, Absolute, Buffered Bytes)	82 02 01
Usage (Reader Encryption Status)	09 42
Report Count (2)	95 02
Input (Data, Variable, Absolute, Buffered Bytes)	82 02 01
Usage (DUKPT Serial Number/Counter)	09 46
Report Count (10)	95 0A
Input (Data, Variable, Absolute, Buffered Bytes)	82 02 01
Usage (Track 1 Masked data length)	09 47
Usage (Track 2 Masked data length)	09 48
Usage (Track 3 Masked data length)	09 49
Report Count (3)	95 03
Input (Data, Variable, Absolute, Bit Field)	81 02
Usage (Track 1 Masked data)	09 4A
Report Count (112)	95 70
Input (Data, Variable, Absolute, Buffered Bytes)	82 02 01
Usage (Track 2 Masked data)	09 4B
Report Count (112)	95 70
Input (Data, Variable, Absolute, Buffered Bytes)	82 02 01
Usage (Track 3 Masked data)	09 4C
Report Count (112)	95 70
Input (Data, Variable, Absolute, Buffered Bytes)	82 02 01
Usage (Encrypted Session ID)	09 50
Report Count (8)	95 08
Input (Data, Variable, Absolute, Buffered Bytes)	82 02 01
Usage (Track 1 Absolute data length)	09 51

Item	Value (Hex)
Usage (Track 2 Masked data length)	09 52
Usage (Track 3 Masked data length)	09 53
Usage (MagnePrint Absolute data length)	09 54
Report Count (4)	95 04
Input (Data, Variable, Absolute, Bit Field)	81 02
Usage (Encryption Counter)	09 55
Report Count (3)	95 03
Input (Data, Variable, Absolute, Buffered Bytes)	82 02 01
Usage (MagneSafe Version Number)	09 56
Report Count (8)	95 08
Input (Data, Variable, Absolute, Buffered Bytes)	82 02 01
Usage (Hashed Track 2 Data)	09 57
Report Count (20)	95 14
Input (Data, Variable, Absolute, Buffered Bytes)	82 02 01
Usage (Command Message)	09 20
Report Count (60)	95 3C
Feature (Data, Variable, Absolute, Buffered Bytes)	B2 02 01
End Collection	C0

### MAGNETIC STRIPE READER USAGE PAGE (KB)

Magnetic Stripe Reader usage page 0xFF00:

Usage ID (Hex)	Usage Name	Usage Type	Report Type
20	Command message	Data	Feature

### REPORT DESCRIPTOR (KB)

The Report Descriptor is structured as follows:

Item	Value(Hex)
Usage Page (Generic Desktop)	05 01
Usage (Keyboard)	09 06
Collection (Application)	A1 01
Usage Page (Key Codes)	05 07
Usage Minimum (224)	19 E0
Usage Maximum (231)	29 E7
Logical Minimum (0)	15 00
Logical Maximum (1)	25 01
Report Size (1)	75 01

Item	Value(Hex)
Report Count (8)	95 08
Input (Data, Variable, Absolute)	81 02
Report Count (1)	95 01
Report Size (8)	75 08
Input (Constant)	81 03
Report Count (5)	95 05
Report Size (1)	75 01
Usage Page (LEDs)	05 08
Usage Minimum (1)	19 01
Usage Maximum (5)	29 05
Output (Data, Variable, Absolute)	91 02
Report Count (1)	95 01
Report Size (3)	75 03
Output (Constant)	91 03
Report Count (6)	95 06
Report Size (8)	75 08
Logical Minimum (0)	15 00
Logical Maximum (101)	25 66
Usage Page (Key Codes)	05 07
Usage Minimum (0)	19 00
Usage Maximum (101)	29 66
Input (Data, Array)	81 00
Logical Maximum (255)	26 FF 00
Usage Page (vendor defined (MSR))	06 00 FF
Usage (command data)	09 20
Report Count	95 18
Feature (Data, Variable, Absolute, Buffered Bytes)	B2 02 01
End Collection	C0

## CARD DATA (HID)

The details about how the card data and commands are structured into HID reports follow later in this section. Windows applications that communicate to this reader can be easily developed. These applications can communicate to the reader using standard windows API calls that communicate to the reader using the standard Windows USB HID driver. These applications can be easily developed using compilers such as Microsoft's Visual Basic or Visual C++. A demonstration program and its source code, written in Visual Basic, that communicates with this reader is available. This demo program can be used to test the reader and it can be used as a guide for developing other applications. More details about the demo program follow later in this document.

It is recommended that application software developers become familiar with the HID specification the USB specification before attempting to communicate with this reader. This document assumes that the reader is familiar with these specifications. These specifications can be downloaded free from [www.usb.org](http://www.usb.org).



Card data is only sent to the host on the Interrupt In pipe using an Input Report. The reader will send only one Input Report per card swipe. If the host requests data from the reader when no data is available, the reader will send a NAK to the host to indicate that it has nothing to send. When a card is swiped, the Input Report will be sent even if the data is not decodable. The following table shows how the input report is structured.

<b>Offset</b>	<b>Usage Name</b>
0	Track 1 decode status
1	Track 2 decode status
2	Track 3 decode status
3	Track 1 encrypted data length
4	Track 2 encrypted data length
5	Track 3 encrypted data length
6	Card encode type
7 – 118	Track 1 encrypted data
119 – 230	Track 2 encrypted data
231 - 342	Track 3 encrypted data
343	Card status
344 – 347	MagnePrint status
348	MagnePrint data length
349 - 476	MagnePrint data
477 – 492	Device serial number
493-494	Reader Encryption Status
495 – 504	DUKPT serial number/counter
505	Track 1 Masked data length
506	Track 2 Masked data length
507	Track 3 Masked data length
508 – 619	Track 1 Masked data
620 – 731	Track 2 Masked data
732 – 843	Track 3 Masked data
844 – 851	Encrypted Session ID
852	Track 1 Absolute data length
853	Track 2 Absolute data length
854	Track 3 Absolute data length
855	MagnePrint Absolute data length
856-858	Encryption Counter
859-866	MagneSafe Version Number
867-886	Hashed Track 2 Data

### Track 1 Decode Status

Bits	7-1	0
Value	Reserved	Error

This is a one-byte value, which indicates the status of decoding track 1. Bit position zero indicates if there was an error decoding track 1 if the bit is set to one. If it is zero, then no error occurred. If a track has data on it that is not noise, and it is not decodable, then a decode error is indicated. If a decode error is indicated, the corresponding track data length value for the track that has the error will be set to zero and no valid track data will be supplied.

### Track 2 Decode Status

Bits	7-1	0
Value	Reserved	Error

This is a one-byte value, which indicates the status of decoding track 2. Bit position zero indicates if there was an error decoding track 2 if this bit is set to one. If it is zero, then no error occurred. If a track has data on it that is not noise, and it is not decodable, then a decode error is indicated. If a decode error is indicated, the corresponding track data length value for the track that has the error will be set to zero and no valid track data will be supplied.

### Track 3 Decode Status

Bits	7-1	0
Value	Reserved	Error

This is a one-byte value, which indicates the status of decoding track 3. Bit position zero indicates if there was an error decoding track 3 if this bit is set to one. If it is zero, then no error occurred. If a track has data on it that is not noise, and it is not decodable, then a decode error is indicated. If a decode error is indicated, the corresponding track data length value for the track that has the error will be set to zero and no valid track data will be supplied.

### Track 1 Encrypted Data Length

This one-byte value indicates the number of bytes in the Track 1 encrypted data field. The field is always a multiple of 8 bytes in length. This value will be zero if there was no data on the track or if there was an error decoding the track. Once the encrypted data is decrypted, there may be fewer bytes of decoded track data than indicated by this field. The number of bytes of decoded track data is indicated by the Track 1 Absolute Data Length field.

### Track 2 Encrypted Data Length

This one-byte value indicates the number of bytes in the Track 2 encrypted data field. The field is always a multiple of 8 bytes in length. This value will be zero if there was no data on the track or if there was an error decoding the track. Once the encrypted data is decrypted, there may be fewer bytes of decoded track data than indicated by this field. The number of bytes of decoded track data is indicated by the Track 2 Absolute Data Length field.

### Track 3 Encrypted Data Length

This one-byte value indicates the number of bytes in the Track 3 encrypted data field. The field is always a multiple of 8 bytes in length. This value will be zero if there was no data on the track or if there was an error decoding the track. Once the encrypted data is decrypted, there may be fewer bytes of decoded track data than indicated by this field. The number of bytes of decoded track data is indicated by the Track 3 Absolute Data Length field.

### Track 1 Absolute Data Length

This one-byte value indicates the number of useable bytes in the Track 1 Encrypted Data field after decryption.

### Track 2 Absolute Data Length

This one-byte value indicates the number of useable bytes in the Track 2 Encrypted Data field after decryption.

### Track 3 Absolute Data Length

This one-byte value indicates the number of useable bytes in the Track 3 Encrypted Data field after decryption.

### Card Encode Type

This one-byte value indicates the type of encoding that was found on the card. The following table defines the possible values.

Value	Encode Type	Description
0	ISO/ABA	ISO/ABA encode format (see Appendix E for ISO/ABA description)
1	AAMVA	AAMVA encode format (see Appendix E for AAMVA description)
2	Reserved	
3	Blank	The card is blank.
4	Other	The card has a non-standard encode format. For example, ISO/ABA track 1 format on track 2.
5	Undetermined	The card encode type could not be determined because no tracks could be decoded.
6	None	No decode has occurred. This type occurs if no magnetic stripe data has been acquired since the data has been cleared or since the reader was powered on. This reader only sends an Input report when a card has been swiped so this value will never occur.

### Encrypted Track Data

If decodable track data exists for a given track, it is located in the *Track x Encrypted Data* field that corresponds to the track number. The length of each *Encrypted Data* field is fixed at 112 bytes, but the length of valid data in each field is determined by the corresponding *Encrypted Data Length* field that corresponds to the track number. Data located in positions greater than the encrypted track data length field indicates are undefined and should be ignored. The HID specification requires that reports be fixed in size, but the number of bytes encoded on a card may vary. Therefore, the Input Report always contains the maximum amount of bytes that can

be encoded on the card and the number of valid bytes in each track is indicated by the *Encrypted Data Length* field.

The encrypted data from each track is decoded and converted to ASCII, and then it is encrypted. The encrypted track data includes all data starting with the start sentinel and ending with the end sentinel. The encryption begins with the first 8 bytes of the clear text track data. The 8-byte result of this encryption is placed in the *Encrypted Data* buffer for the corresponding track. The process continues using the CBC (Cipher Block Chaining) method with the encrypted 8 bytes XORed with the next 8 bytes of clear text. That result is placed in next 8 bytes of the *Encrypted Data* buffer and the process continues until all clear text bytes have been encrypted. If the final block of clear text contains fewer than 8 bytes, it is padded with binary zeros to fill up the 8 bytes. After this final clear text block is XORed with the prior 8 bytes of encrypted data, it is encrypted and placed in the *Encrypted Data* buffer. No Initial Vector is used in the process.

Decrypting the data must be done in 8 byte blocks, ignoring any final unused bytes in the last block. See Appendix C for more information.

### **Track 1 Encrypted Data**

This field contains the encrypted track data for track 1.

### **Track 2 Encrypted Data**

This field contains the encrypted track data for track 2.

### **Track 3 Encrypted Data**

This field contains the encrypted track data for track 3.

### **Card Status**

For the swipe reader, this one byte field is reserved for future use. It is currently not used on the swipe reader.

For the insert reader this field is defined as follows:

Bits	7-1	0
Value	Reserved	Card Inserted

This is a one-byte value, which indicates the card status. Bit position zero indicates that the card was swiped in the insertion direction if it is set to 1. If it is set to 0, then the card was swiped in the withdrawal direction. All other bit positions are reserved.

## MagnePrint Status

This Binary field represents 32 bits of MagnePrint status information. Each character represents 4 bits (hexadecimal notation). For example, suppose the characters are: “A1050000”:

Nibble	1	2	3	4	5	6	7	8
Value	A	1	0	5	0	0	0	0
Bit	7 6 5 4 3 2 1 0	15 14 13 12 11 10 9 8	23 22 21 20 19 18 17 16	31 30 29 28 27 26 25 24				
Value	1 0 1 0 0 0 0 1	0 0 0 0 0 0 1 0	1 0 1 0 0 0 0 0	0 0 0 0 0 0 0 0				
Usage*	R R R R R R R M	R R R R R R R R	0 0 D 0 F L N S	0 0 0 0 0 0 0 0				

\* Usage Legend:

- D = Direction
- F = Too Fast
- L = Too Slow
- M = MagnePrint capable
- N = Too Noisy
- R = Revision

This four-byte field contains the MagnePrint status. The MagnePrint status is in little endian byte order. Byte 1 is the least significant byte. Byte 1 LSB is status bit 0. Byte 4 MSB is status bit 31. MagnePrint status is defined as follows:

Bit 0	=	This is a MagnePrint-capable product (usage M)
Bits 1-15	=	Product revision & mode (usage R)
Bit 16	=	STATUS-only state (usage S)
Bit 17	=	Noise too high or “move me” away from the noise source (used only in STATUS) (usage N)
Bit 18	=	Swipe too slow (usage L)
Bit 19	=	Swipe too fast (usage F)
Bit 20	=	Unassigned (always set to Zero)
Bit 21	=	Actual Card Swipe Direction (0 = Forward, 1 = Reverse) (usage D)
Bits 22-31	=	Unassigned (always set to Zero)

If the Enable/Disable MagnePrint property is set to disable MagnePrint, this field will not be sent.

## MagnePrint Data Length

This one-byte value indicates the number of bytes in the Encrypted MagnePrint Data field. The field is always a multiple of 8 bytes in length. This value will be zero if there was no MagnePrint data. Once the encrypted data is decrypted, there may be fewer bytes of decoded MagnePrint data than indicated by this field. The number of bytes of decoded MagnePrint data is indicated by the MagnePrint Absolute Data Length field.

## MagnePrint Absolute Data Length

This one-byte value indicates the number of useable bytes in the Encrypted MagnePrint Data field after decryption.

### **Encrypted MagnePrint Data**

This 128 byte Binary field contains the MagnePrint data. Only the number of bytes specified in the MagnePrint data length field are valid. The least significant bit of the first byte of data in this field corresponds to the first bit of MagnePrint data. If the Enable/Disable MagnePrint property is set to disable MagnePrint, this field will not be sent.

### **Device Serial Number**

This 16-byte ASCII field contains the device serial number. The device serial number is a NUL (zero) terminated string. So the maximum length of the device serial number, not including the null terminator, is 15 bytes. This device serial number can also be retrieved and set with the device serial number property explained in the property section of this document. This field is stored in non-volatile memory, so it will persist when the unit is power cycled.

### **Track 1 Masked Data Length**

This one-byte value indicates how many bytes of decoded card data are in the track 1 Masked Data field. This value will be zero if there was no data on the track or if there was an error decoding the track.

### **Track 2 Masked Data Length**

This one-byte value indicates how many bytes of decoded card data are in the track 2 Masked Data field. This value will be zero if there was no data on the track or if there was an error decoding the track.

### **Track 3 Masked Data Length**

This one-byte value indicates how many bytes of decoded card data are in the track 3 Masked Data field. This value will be zero if there was no data on the track or if there was an error decoding the track.

### **Masked Track Data**

If decodable track data exists for a given track, it is located in the Masked Track Data field that corresponds to the track number. The length of each Masked Track Data field is fixed at 112 bytes, but the length of valid data in each field is determined by the Masked Track Data Length field that corresponds to the track number. Masked Track Data located in positions greater than indicated in the Masked Track Data Length field are undefined and should be ignored. The HID specification requires that reports be fixed in size but the number of bytes encoded on a card may vary. Therefore, the Input Report always contains the maximum amount of bytes that can be encoded on the card and the number of valid bytes in each track is indicated by the Masked Track Data Length field.

The Masked Track Data is decoded and converted to ASCII and then it is “masked.” The Masked Track Data includes all data starting with the start sentinel and ending with the end sentinel. Much of the data is “masked;” a specified mask character is sent instead of the actual character read from the track. Which characters are masked depends on the format of the card. Only ISO/ABA (Financial Cards with Format Code B) and AAMVA cards are selectively

masked; all other card types are either wholly masked or wholly clear. There is a separate masking property for ISO/ABA cards and AAMVA cards. See the ISO Track Masking property and the AAMVA Track Masking property for more information. (Refer to Appendix E for a description on how ISO/ABA and AAMVA cards are identified.)

Each of these properties allows the application to specify details of masking for Primary Account Number and Driver's License / ID Number (DL/ID#), the masking character to be used, and whether a correction should be applied to make the Mod 10 9 (Luhn algorithm) digit at the end of the number be correct.

### **Track 1 Masked Data**

This field contains the Masked Track Data for track 1. All characters are transmitted.

For an ISO/ABA card, the PAN is masked as follows:

- The specified number of initial characters is sent unmasked. The specified number of trailing characters is sent unmasked. If Mod 10 correction is specified, all but one of the intermediate characters of the PAN are set to zero; one of them will be set such that last digit of the PAN calculates an accurate Mod 10 check of the rest of the PAN *as transmitted*. If the Mod 10 correction is not specified, all of the intermediate characters of the PAN are set to the specified mask character.
- The Card Holder's name and the Expiration Date are transmitted unmasked.
- All Field Separators are sent unmasked.
- All other characters are set to the specified mask character.

For an AAMVA card, the specified mask character is substituted for each of the characters read from the card.

### **Track 2 Masked Data**

This field contains the Masked Track Data for track 2.

For an ISO/ABA card, the PAN is masked as follows:

- The specified number of initial characters are sent unmasked. The specified number of trailing characters are sent unmasked. If Mod 10 correction is specified, all but one of the intermediate characters of the PAN are set to zero; one of them will be set such that last digit of the PAN calculates an accurate Mod 10 check of the rest of the PAN as transmitted. If the Mod 10 correction is not specified, all of the intermediate characters of the PAN are set to the specified mask character.
- The Expiration Date is transmitted unmasked.
- All Field Separators are sent unmasked.
- All other characters are set to the specified mask character.

For an AAMVA card, the DL/ID# is masked as follows:

- The specified number of initial characters are sent unmasked. The specified number of trailing characters are sent unmasked. If Mod 10 correction is specified, all but one of the intermediate characters of the DL/ID#PAN are set to zero; one of them will be set such that last digit of the DL/ID# calculates an accurate Mod 10 check of the rest of the DL/ID# as transmitted. If the Mod 10 correction is not specified, all of the intermediate characters of the DL/ID# are set to the specified mask character.
- The Expiration Date and Birth Date are transmitted unmasked.
- All other characters are set to the specified mask character.

### Track 3 Masked Data

This field contains the Masked Track Data for track 3.

For an ISO/ABA card, the PAN is masked as follows:

- The specified number of initial characters are sent unmasked. The specified number of trailing characters are sent unmasked. If Mod 10 correction is specified, all but one of the intermediate characters of the PAN are set to zero; one of them will be set such that last digit of the PAN calculates an accurate Mod 10 check of the rest of the PAN as transmitted. If the Mod 10 correction is not specified, all of the intermediate characters of the PAN are set to the specified mask character.
- All Field Separators are sent unmasked.
- All other characters are set to the specified mask character.

For an AAMVA card, the specified mask character is substituted for each of the characters read from the card.

### Encrypted Session ID

This 8-byte Binary field contains the encrypted version of the current Session ID. Its primary purpose is to prevent replays. After a card is read, this property will be encrypted, along with the card data, and supplied as part of the transaction message. The clear text version of this will never be transmitted. To avoid replay, the application sets the Session ID property before a transaction and verifies that the Encrypted Session ID returned with card data decrypts to the value set.

### DUKPT Key Serial Number

This 10 byte Binary field contains the DUKPT Key Serial Number used to encrypt the encrypted fields in this message. This 80-bit field includes the Initial Key Serial Number in the leftmost 59 bits and a value for the Encryption Counter in the rightmost 21 bits. If no keys are loaded, all bytes will have the value 0x00.

### Encryption Counter

This 3-byte field contains the value of the Encryption Counter at the end of this transaction. See the **Get Encryption Counter** command for more information.



### MagneSafe Version Number

This eight byte field contains the MagneSafe Version Number with at least one terminating byte of zero to make string manipulation convenient. See the **MagneSafe Version Number** Property for more information.

### Hashed Track 2 Data

This twenty (20) byte field contains the hashed track 2 data with SHA1 algorithm.

### Clear Text CRC

This 2-byte Binary field contains a clear text version of a Cyclical Redundancy Check (CRC-16 CCITT, polynomial 0x1021) (least significant byte sent first). It provides a CRC of all characters sent prior to this CRC. The CRC is converted to four characters of ASCII before being sent. The application may calculate a CRC from the data received prior to this CRC and compare it to the CRC received. If they are the same, the application can have high confidence that all the data was received correctly. The **CRC Flag** property controls whether this field is sent.

### Encrypted CRC

This 8-byte Binary field contains an encrypted version of a Cyclical Redundancy Check (CRC). It provides a CRC of all characters sent prior to this CRC. The CRC is converted to 16 characters of ASCII before being sent. After the receiver decrypts the message, the CRC is contained in the first 2 bytes of the message, all other bytes are meaningless. The application may calculate a CRC from the data received prior to this CRC and compare it to the CRC received. If they are the same, the application can have high confidence that all the data was received correctly. The **CRC Flag** property controls whether this field is sent.

### CARD DATA (KB)

The card data is converted to ASCII and transmitted to the host as if it had been typed on a keyboard. Any data with ASCII values 0 – 31 or 127 will be transmitted as their equivalent control code combination. For example a carriage return value 13 (0x0D) will be sent as (^M) where ^ represents the Ctrl key on the keyboard.

#### *Caution*

*If another keyboard is connected to the same host as this reader and a key is pressed on the other keyboard while this reader is transmitting, then the data transmitted by this reader may get corrupted.*

The reader's programmable configuration options affect the format of the card data. During normal reader operation, the reader acts like a USB HID keyboard so the host operating system takes care of all low level communications with the reader so that the application developer is not burdened with these low level details.

All data will be sent in upper case regardless of the state of the caps lock key on the keyboard. If no data is detected on a track then nothing will be transmitted for that track. If an error is detected on a track, the ASCII character “E” will be sent in place of the track data to indicate an error.

For the encrypted fields, the original binary bytes are encrypted using the DES CBC mode with an Initialization Vector starting at all binary zeroes and the PIN Encryption Key associated with the current DUKPT KSN. This is done in segments of 8 bytes. If the last segment of the original data is less than eight bytes long (track data only), the last bytes of the block will be set to binary zeroes before encrypting. When decrypting track data, the End Sentinel can be used to find the actual end of the data (ignoring the final zeroes). Each byte of encrypted data is then converted to *two* bytes of ASCII data representing the Hexadecimal value of the encrypted byte (many of the encrypted bytes will not have values in the ASCII character range).

When the reader is in Security Level 2, the factory default settings cause the data to be transmitted in the SureSwipe format (see MagTek manual 99875206).

The card data format for all programmable configuration options is as follows:

- [P30]
- [P32] [Tk1 SS] [Tk1 Masked Data] [ES] [P33]
- [P32] [Tk2 SS] [Tk2 Masked Data] [ES] [P33]
- [P32] [Tk3 SS] [Tk3 Masked Data] [ES] [P33]
- [P31]
- [P35] [Reader Encryption Status]
- [P35] [Tk1 Encrypted Data (including TK1 SS and ES)]
- [P35] [Tk2 Encrypted Data (including TK2 SS and ES)]
- [P35] [Tk3 Encrypted Data (including TK3 SS and ES)]
- [P35] [MagnePrint Status]
- [P35] [Encrypted MagnePrint data]
- [P35] [Device serial number]
- [P35] [Encrypted Session ID]
- [P35] [DUKPT serial number/counter]
- [P35] [Encryption Counter] (optional, off by default)
- [P35] [Clear Text CRC]
- [P35] [Encrypted CRC]
- [P35] [Format Code]
- [P34]

The characters and fields are described in the list below. The Property ID (e.g., P13) is the decimal value of the property ID in the command list (see **Pre Card String**).

Label	Property ID	Description	Default
	0x1E	Pre card string	0 (0x00)
	0x1F	Post card string	0 (0x00)
	0x20	Pre track string	0 (0x00)
	0x21	Post track string	0 (0x00)
	0x22	Terminating string	C/R (0x0D)

Label	Property ID	Description	Default
	0x23	Programmable field separator (this key is never found in track data or the default programmable field separators)	" " (0x7C)
Tk1 SS	0x24	ISO/ABA start sentinel	"%" (0x25)
Tk2-SS	0x25	ISO/ABA 5-bit start sentinel	"," (0x3B)
Tk3-SS	0x26	ISO/ABA start sentinel	"+" (0x2B)
Tk3-SS AAMVA	0x27	AAMVA start sentinel	"#" (0x23)
Tk2-SS 7 bit	0x28	7 bit start sentinel (ISO/ABA Track 1 start sentinel)	"@" (0x40)
Tk3-SS 7 bit	0x29	7 bit start sentinel (ISO/ABA Track 1 start sentinel)	"&" (0x26)
ES	0x2B	End Sentinel	"?" (0x3F)
	0x2D	Track 1 Specific End Sentinel	"?" (0x3F)
	0x2E	Track 2 Specific End Sentinel	"?" (0x3F)
	0x2F	Track 3 Specific End Sentinel	"?" (0x3F)

Track 1, Track 2 and Track 3 Encrypted Data includes the Start and End Sentinel that were decoded from the card.

All fields with the format P# are programmable configuration property numbers. They are described in detail later in this document.

### Reader Encryption Status

This two byte Binary field contains the Encryption Status. The Reader Encryption Status is sent in big endian byte order. Byte 1 is the least significant byte. Byte 1 LSB is status bit 0. Byte 2 MSB is status bit 15. The Reader Encryption Status is defined as follows:

Bit 0	=	DUKPT Keys exhausted
Bit 1	=	Initial DUKPT key Injected
Bit 2	=	Encryption Enabled
Bit 3	=	Authentication Required
Bit 4	=	Timed Out waiting for user to swipe card
Bit 5	=	Encrypting IntelliHead Communication Error – Message too Short
Bit 6	=	Encrypting IntelliHead Communication Error – Wrong Message Type
Bit 7	=	Key Synchronization Error (only on units equipped with Encrypting IntelliHead)
Bit 8	=	Encryption Counter Expired
Bits 9–15	=	Unassigned (always set to Zero)

Notes:

- (1) Encryption will only be performed when Encryption Enabled and Initial DUKPT key Injected are set. Otherwise, data that are normally encrypted are sent in the clear in ASCII HEX format; the DUKPT Serial Number/counter will not be sent.
- (2) When DUKPT Keys Exhausted is set, the reader will no longer read cards and after a card swipe, the reader response will be sent as follows:
  - [P30]
  - [P31]
  - [P35] [Reader Encryption Status]
  - [P35]

[P35]  
[P35]  
[P35]  
[P35]  
[P35] [Device serial number]  
[P35] [Encrypted Session ID]  
[P35] [DUKPT serial number/counter]  
[P35] [Encryption Counter] (optional, off by default)  
[P35] [Clear Text CRC]  
[P35] [Encrypted CRC]  
[P35] [Format Code]  
[P34]

### **Format Code**

This 4-character ASCII field contains the Format Code. The purpose of the Format Code is to allow the receiver of this message to know how to find the different fields in the message. The default Format Code for this reader is “0000”. If any of the properties that affect the format of the message are changed, the first character of the Format Code will automatically change to a “1”. The application may change the final three characters, but making such a change will automatically cause the first character to a “1”.

### **PROGRAMMABLE CONFIGURATION OPTIONS**

This reader has a number of programmable configuration properties. Most of the programmable properties deal with the Keyboard Emulation mode but some of the properties deal with the reader regardless of the mode. These properties are stored in non-volatile memory. These properties can be configured at the factory or by the end user using a program supplied by MagTek. Programming these parameters requires low level communications with the reader. Details on how to communicate with the reader to change programmable configuration properties follows in the next few sections. These details are included as a reference only. Most users will not need to know these details because the reader will be configured at the factory or by a program supplied by MagTek. Most users may want to skip over the next few sections on low level communications and continue with the details of the configuration properties.

### **Low Level Communications**

It is strongly recommended that application software developers become familiar with the HID specification the USB specification before attempting to communicate directly with this reader. This document assumes that the reader is familiar with these specifications. These specifications can be downloaded free from [www.usb.org](http://www.usb.org).

## COMMANDS

Most host applications do not need to send commands to the reader. Most host applications only need to obtain card data from the reader as described previously in this section. This section of the manual can be ignored by anyone who does not need to send commands to the reader.

Commands sent to the wireless reader model will only succeed if the reader is both on and within range of the dongle.

Command requests and responses are sent to and received from the reader using feature reports. Command requests are sent to the reader using the HID class specific request **Set Report**. The response to a command is retrieved from the reader using the HID class specific request **Get Report**. These requests are sent over the default control pipe. When a command request is sent, the reader will NAK the Status stage of the **Set Report** request until the command is completed. This insures that, as soon as the **Set Report** request is completed, the **Get Report** request can be sent to get the command response. The usage ID for the command message was shown previously in the Usage Table.

The following table shows how the feature report is structured for command requests:

Offset	Field Name
0	Command Number
1	Data Length
2 – 49	Data

The following table shows how the feature report is structured for command responses.

Offset	Field Name
0	Result Code
1	Data Length
2 – 49	Data

## PRIVILEGED COMMANDS

Some commands are, for security purposes, privileged. These commands are:

1. Set Property
2. Reset Device\*
3. Set Key Map Item
4. Save Custom Key Map
5. Set Security Level†

\* The Reset Device command is usually not Privileged. The exception is during a sequence to Activate the Authenticated Mode. During this sequence the Reset Device command is Privileged to avoid a hacker using this sequence to exhaust DUKPT keys rendering the reader unusable.

† The Set Security Level command is Privileged when it is being used to set the Security Level. It is not Privileged when it is being used to Get the Security Level.

When the Security Level is set to higher than 2 (see the Security section), the privileged commands must be MACed in order to be accepted. If a MAC is required but not present or incorrect, RC = 07 will be returned.

**COMMAND NUMBER**

This one-byte field contains the value of the requested command number. The following table lists all the existing commands.

<b>Value (Hex)</b>	<b>Command Number</b>	<b>Description</b>
00	Get Property	Gets a property from the reader
01	Set Property	Sets a property in the reader
02	Reset Device	Resets the reader
03	Get Keymap Item	Gets a key map item (KB only)
04	Set Keymap Item	Sets a key map item (KB only)
05	Save Custom Keymap	Saves the custom key map (KB only)
09	Get DUKPT KSN	Reports DUKPT KSN and Counter
0A	Set Session ID	Sets the current Session ID
10	Activate Authenticated Mode	Starts Activation of Authenticated Mode of secure operation
11	Activation Challenge Reply Command	Completes the Activation of Authenticated Mode of secure operation
12	Deactivate Authenticated Mode	Deactivates the Authenticated Mode of secure operation.
13	Reserved	
14	Get Reader State	Gets the current state of the reader.
15	Set Security Level	Sets or gets the current Security Level
28	Power Down MSR (wireless reader only)	Powers down the MSR circuits (if running on battery turns reader off).
29	Get Battery Status (wireless reader only)	Gets Charge Status of battery

**DATA LENGTH**

This one-byte field contains the length of the valid data contained in the Data field. For example, a command with one byte of data would send 01 for this byte; a command with 18 bytes of data would send 12 for this byte.

**DATA**

This multi-byte field contains command data if any. Note that the length of this field is fixed at 60 bytes. Valid data should be placed in the field starting at offset 2. Any remaining data after the valid data should be set to zero. This entire field must always be set even if there is no valid data. The HID specification requires that Reports be fixed in length. Command data may vary in length. Therefore, the Report should be filled with zeros after the valid data.

## RESULT CODE

This one-byte field contains the value of the result code. There are two types of result codes: generic result codes and command-specific result codes. Generic result codes always have the most significant bit set to zero. Generic result codes have the same meaning for all commands and can be used by any command. Command-specific result codes always have the most significant bit set to one. Command-specific result codes are defined by the command that uses them. The same code can have different meanings for different commands. Command-specific result codes are defined in the documentation for the command that uses them. Generic result codes are defined in the following table.

Value (Hex)	Result Code	Description
00	Success	The command completed successfully.
01	Failure	The command failed.
02	Bad Parameter	The command failed due to a bad parameter or command syntax error.
05	Delayed	The request is refused due to anti-hacking mode
07	Invalid Operation	Depends on context of command

## GET AND SET PROPERTY COMMANDS

The **Get Property** command gets a property from the reader. The **Get Property** command number is 00.

The **Set Property** command sets a property in the reader. The **Set Property** command number is 01. For security purposes, this command is privileged. When the Security Level is set to higher than 2 (see the Security section), this commands must be MACed in order to be accepted.

The **Get** and **Set Property** command data fields for the requests and responses are structured as follows.

**Get Property Request Data:**

Data Offset	Value
0	Property ID

**Get Property Response Data:**

Data Offset	Value
0 – n	Property Value

**Set Property Request Data:**

Data Offset	Value
0	Property ID
1 – n	Property Value

**Set Property Response Data:**

None

The result codes for the **Get** and **Set Property** commands can be any of the codes listed in the generic result code table. If the **Set Property** command gets a result code of 0x07, it means the required MAC was absent or incorrect.

## Property ID

Property ID is a one-byte field that contains a value that identifies the property. The following table lists all the current property ID values:

Value (hex) HID mode	Value (hex) KB mode	Property	Description
00	00	Software ID	The reader's software identifier (dongle in wireless reader)
01	01	USB Serial Num	The reader's USB serial number
02	02	Polling Interval	The interrupt pipe's polling interval
03	03	Device Serial Num	Device serial number
04	04	MagneSafe Version Number	Version number of MagneSafe feature set
05	05	Track ID Enable	Track enable / ID enable
06	06	Reserved for future use	
07	07	ISO Track Mask	Specifies Masking factors for ISO cards
08	08	AAMVA Track Mask	Specifies Masking factors for AAMVA cards
09	09	Reserved for future use	
0A	-	Max Packet Size	The interrupt pipe's packet size
0B	0B	Activity Timeout Period (wireless reader only)	Specifies minimum time reader will operate in the absence of activity (used to conserve battery life)
0C-0D	0C-0D	Reserved for future use	
0E	0E	Stay Powered After Swipe (wireless reader only)	Allows reader to stay powered after a good swipe
0F	0F	RF Address (wireless reader only)	Specifies the RF address used to pair the wireless dongle and reader together.
10	10	Interface Type	Type of USB interface
11-14	-	Reserved for future use	
-	14	Track Data Send Flags	Track data send flags
15	15	MP Flags	Enables sending of MagnePrint data
-	16	Active Keymap	Selects which key map to use
-	17	ASCII To Keypress Conversion Type	Type of conversion performed when converting ASCII data to key strokes
18	18	Reserved for future use	
-	19	CRC Flag	Enables/disables sending CRC
-	1A	SureSwipe Flag	Enables/disables SureSwipe Emulation for Security Level 2.
	1B-1D	Reserved for future use	
-	1E	Pre Card String	Pre card string
-	1F	Post Card String	Post card string
-	20	Pre TK String	Pre track string
-	21	Post TK String	Post track string
-	22	Termination String	Terminating string
-	23	FS	Field Separator for additional data
-	24	SS TK1 ISO ABA	Start sentinel char for track 1 – ISO/ABA
-	25	SS TK2 ISO ABA	Start sentinel char for track 2 – ISO/ABA



Value (hex) HID mode	Value (hex) KB mode	Property	Description
-	26	SS TK3 ISO ABA	Start sentinel char for track 3 – ISO/ABA
-	27	SS TK3 AAMVA	Start sentinel char for track 3 - AAMVA
-	28	SS TK2 7BITS	Start sentinel char for track 2 – 7 bit data
-	29	SS TK3 7BITS	Start sentinel char for track 3 – 7 bit data
-	2A	Reserved for future use	
-	2B	ES	End sentinel char for all tracks/formats
-	2C	Format Code	Defines Format Code to be sent with the message
-	2D	ES Track 1	End sentinel char for track 1
-	2E	ES Track 2	End sentinel char for track 2
-	2F	ES Track 3	End sentinel char for track 3
-	30	Send Encryption Counter	Enables/disables sending of Encryption Counter
31	31	Mask “Other” Cards	Enables/disables masking of cards that don’t meet the ISO Financial or the AAMVA formats.
34	34	Send clear AAMVA card data flag	Enables/disables sending of clear AAMVA card data
3A	00	Software ID 2 (wireless reader only)	The wireless reader’s software identifier (not the dongle’s)

The Property Value is a multiple-byte field that contains the value of the property. The number of bytes in this field depends on the type of property and the length of the property. The following table lists all of the property types and describes them.

Property Type	Description
Byte	This is a one-byte value. The valid values depend on the property.
String	This is a multiple byte ASCII string. Its length can be zero to a maximum length that depends on the property. The value and length of the string does not include a terminating NUL character.

### SOFTWARE ID PROPERTY

Property ID: 0x00  
 Property Type: String  
 Length: Fixed at 11 bytes  
 Get Property: Yes  
 Set Property: No  
 Description: This is an 11 byte read only property that identifies the software part number and version for the reader. The first 8 bytes represent the part number and the last 3 bytes represent the version. For example, this string might be “21042812D01”.

For the wireless reader, two software IDs are available, one in the dongle and one in the wireless reader. This property returns the software ID from the dongle. To get the software ID from the wireless reader use the “SOFTWARE ID 2” property. When the wireless reader is directly connected to the host using a USB cable, as is the case when doing firmware updates, this property will return the software ID of the wireless reader.

Examples follow:

Example Get **Software ID** property Request (Hex):

Cmd Num	Data Len	Prp ID
00	01	00

Example Get **Software ID** property Response (Hex):

Result Code	Data Len	Prp Value
00	01	32 31 30 34 32 38 31 32 44 30 31

## USB SERIAL NUM PROPERTY

Property ID: 0x01  
 Property Type: String  
 Length: 0 – 15 bytes  
 Get Property: Yes  
 Set Property: Yes (Once only)  
 Default Value: The default value is no string with a length of zero.  
 Description: The value is an ASCII string that represents the USB serial number. This string can be 0 – 15 bytes long. The value of this property, if any, will be sent to the host when the host requests the USB string descriptor.

This property is stored in non-volatile memory, so it will persist when the unit is power cycled. When this property is changed, the unit must be reset (see Command Number 2) or power cycled to have these changes take effect. This reader must be unplugged for at least 30 seconds to properly power cycle it.

Example Set **USB Serial Num** property Request (Hex):

Cmd Num	Data Len	Prp ID	Prp Value
01	04	01	31 32 33

Example Set **USB Serial Num** property Response (Hex):

Result Code	Data Len	Data
00	00	

Example Get **USB Serial Num** property Request (Hex):

Cmd Num	Data Len	Prp ID
00	01	01

Example Get **USB Serial Num** property Response (Hex):

Result Code	Data Len	Prp Value
00	03	31 32 33

## POLLING INTERVAL PROPERTY

Property ID: 0x02  
 Property Type: Byte  
 Length: 1 byte  
 Get Property: Yes

Set Property: Yes  
 Default Value: 1  
 Description: The value is a byte that represents the reader's polling interval for the Interrupt In Endpoint. The value can be set in the range of 1 – 255 and has units of milliseconds. The polling interval tells the host how often to poll the reader for card data packets. For example, if the polling interval is set to 10, the host will poll the reader for card data packets every 10ms. This property can be used to speed up or slow down the time it takes to send card data to the host. The trade-off is that speeding up the card data transfer rate increases the USB bus bandwidth used by the reader, and slowing down the card data transfer rate decreases the USB bus bandwidth used by the reader. The value of this property, if any, will be sent to the host when the host requests the reader's USB endpoint descriptor.

This property is stored in non-volatile memory, so it will persist when the unit is power cycled. When this property is changed, the unit must be reset (see Command Number 2) or power cycled to have these changes take effect. This reader must be unplugged for at least 30 seconds to properly power cycle it.

Example Set **Polling Interval** property to 10ms Request (Hex):

Cmd Num	Data Len	Prp ID	Prp Value
01	02	02	0A

Example Set **Polling Interval** property Response (Hex):

Result Code	Data Len	Data
00	00	

Example Get **Polling Interval** property Request (Hex):

Cmd Num	Data Len	Prp ID
00	01	02

Example Get **Polling Interval** property Response (Hex):

Result Code	Data Len	Prp Value
00	01	01

## DEVICE SERIAL NUM PROPERTY

Property ID: 0x03  
 Property Type: String  
 Length: 0 – 15 bytes  
 Get Property: Yes  
 Set Property: Yes (Once only)  
 Default Value: The default value is no string with a length of zero.  
 Description: The value is an ASCII string that represents the reader serial number. This string can be 0 – 15 bytes long. The value of this property, if any, will be sent to the host in the device serial number field of the USB input report when a card is swiped. This is explained in the card data section of this document.

This property may be Set once only. Attempts to Set the property again will fail with RC = 0x07 (Sequence Error).

This property is stored in non-volatile memory, so it will persist when the unit is power cycled. When this property is changed, the unit must be reset (see Command Number 2) or power cycled to have these changes take effect. This reader must be unplugged for at least 30 seconds to properly power cycle it.

Example Set **Device Serial Num** property Request (Hex):

Cmd Num	Data Len	Prp ID	Prp Value
01	04	03	31 32 33

Example Set **Device Serial Num** property Response (Hex):

Result Code	Data Len	Data
00	00	

Example Get **Device Serial Num** property Request (Hex):

Cmd Num	Data Len	Prp ID
00	01	03

Example Get **Device Serial Num** property Response (Hex):

Result Code	Data Len	Prp Value
00	03	31 32 33

## MAGNESAFE VERSION NUMBER

Property ID: 0x04  
 Property Type: String  
 Length: 0 – 7 bytes  
 Get Property: Yes  
 Set Property: No  
 Default Value: “V05” (may change later)  
 Description: This is a max 7 byte **read only** property that identifies the MagneSafe Feature Level supported on this reader. Attempts to set this property will fail with RC=01.

Example Get **MagneSafe Version Number** property Request (Hex):

Cmd Num	Data Len	Prp ID
00	01	04

Example Get **MagneSafe Version Number** property Response (Hex):

Result Code	Data Len	Prp Value
00	02	56 30 35

**TRACK ID ENABLE PROPERTY**

Property ID: 0x05  
 Property Type: Byte  
 Length: 1 byte  
 Get Property: Yes  
 Set Property: Yes  
 Default Value: 0x95

Description: This property is defined as follows:

id	0	T <sub>3</sub>	T <sub>3</sub>	T <sub>2</sub>	T <sub>2</sub>	T <sub>1</sub>	T <sub>1</sub>
----	---	----------------	----------------	----------------	----------------	----------------	----------------

Id 0 – Decodes standard ISO/ABA cards only  
 1 – Decodes AAMVA and 7-bit cards also

If this flag is set to 0, only tracks that conform to the ISO format allowed for that track will be decoded. If the track cannot be decoded by the ISO method it will be considered to be in error.

T<sub>#</sub> 00 – Track Disabled  
 01 – Track Enabled  
 10 – Track Enabled/Required (Error if blank)

This property is stored in non-volatile memory, so it will persist when the unit is power cycled. When this property is changed, the unit must be reset (see Command Number 2) or power cycled to have these changes take effect. This reader must be unplugged for at least 30 seconds to properly power cycle it.

Example Set **Track ID Enable** property Request (Hex):

Cmd Num	Data Len	Prp ID	Prp Value
01	02	05	95

Example Set **Track ID Enable** property Response (Hex):

Result Code	Data Len	Data
00	00	

Example Get **Track ID Enable** property Request (Hex):

Cmd Num	Data Len	Prp ID
00	01	05

Example Get **Track ID Enable** property Response (Hex):

Result Code	Data Len	Prp Value
00	01	95

**ISO TRACK MASK PROPERTY**

Property ID: 0x07  
 Property Type: String  
 Length: 6 bytes  
 Get Property: Yes

Set Property: Yes  
Default Value: "04040Y"  
Description: This property specifies the factors for masking data on ISO/ABA type cards:

- The first two bytes specify how many of the leading characters of the PAN should be sent unmasked. The range of masking is from "00" to "99."
- The next two bytes specify how many of the trailing characters of the PAN should be sent unmasked. The range of masking is from "00" to "99."
- The fifth byte specifies which character should be used for masking. If this byte contains the uppercase letter 'V', the following rules apply:
  - The character used for masking the PAN will be '0'
  - All data after the PAN will be sent without masking
- The sixth byte specifies whether the Mod 10 Correction should be applied to the PAN. "Y" means Yes, the Mod 10 Correction will be applied. "N" means No, the Mod 10 will not be applied. (This option is only effective if the masking character is "0".)

This property is stored in non-volatile memory, so it will persist when the unit is power cycled. When this property is changed, the unit must be reset (see Command Number 2) or power cycled to have these changes take effect.

### **AAMVA TRACK MASK PROPERTY**

Property ID: 0x08  
Property Type: String  
Length: 6 bytes  
Get Property: Yes  
Set Property: Yes  
Default Value: "04040Y"  
Description: This property specifies the factors for masking data on AAMVA type cards:

- The first two bytes specify how many of the leading characters of the Driver's License/ID Number (DL/ID#) should be sent unmasked. The range of masking is from "00" to "99."
- The next two bytes specify how many of the trailing characters of the DL/ID# should be sent unmasked. The range of masking is from "00" to "99."
- The fifth byte specifies which character should be used for masking. If this byte contains the uppercase letter 'V', the following rules apply:
  - The PAN will be masked according to the rules of this property (the Send Clear AAMVA Card Data property is ignored)
  - The character used for masking the PAN will be '0'
  - All data after the PAN will be sent without masking
- The sixth byte specifies whether the Mod 10 Correction should be applied to the DL/ID#. "Y" means Yes, the Mod 10 Correction will be applied. "N" means No, the Mod 10 will not be applied. (This option is only effective if the masking character is "0".)

This property is stored in non-volatile memory, so it will persist when the unit is power cycled. When this property is changed, the unit must be reset (see Command Number 2) or power cycled to have these changes take effect.

### MAX PACKET SIZE PROPERTY (HID)

Property ID: 0x0A  
 Property Type: Byte  
 Length: 1 byte  
 Get Property: Yes  
 Set Property: Yes  
 Default Value: 8  
 Description: The value is a byte that represents the reader's maximum packet size for the Interrupt In Endpoint. The value can be set in the range of 1 – 64 and has units of bytes. The maximum packet size tells the host the maximum size of the Interrupt In Endpoint packets. For example, if the maximum packet size is set to 8, the reader will send HID reports in multiple packets of 8 bytes each or less for the last packet of the report. This property can be used to speed up or slow down the time it takes to send card data to the host. Larger packet sizes speed up communications and smaller packet sizes slow down communications. The trade-off is that speeding up the card data transfer rate increases the USB bus bandwidth used by the reader, and slowing down the card data transfer rate decreases the USB bus bandwidth used by the reader. The value of this property will be sent to the host when the host requests the reader's USB endpoint descriptor.

This property is stored in non-volatile memory, so it will persist when the unit is power cycled. When this property is changed, the unit must be reset (see Command Number 2) or power cycled to have these changes take effect. This reader must be unplugged for at least 30 seconds to properly power cycle it.

Example Set **Max Packet Size** property Request (Hex):

Cmd Num	Data Len	Prp ID	Prp Value
01	02	0A	08

Example Set **Max Packet Size** property Response (Hex):

Result Code	Data Len	Data
00	00	

Example Get **Max Packet Size** property Request (Hex):

Cmd Num	Data Len	Prp ID
00	01	0A

Example Get **Max Packet Size** property Response (Hex):

Result Code	Data Len	Prp Value
00	01	08

### ACTIVITY TIMEOUT PERIOD PROPERTY (WIRELESS READER ONLY)

Property ID:	0x0B
Property Type:	Byte
Length:	1 byte
Get Property:	Yes
Set Property:	Yes
Default Value:	120 (0x78) seconds
Description:	This property specifies, in seconds, the minimum amount of time a wireless reader will operate in the absence of activity. Activity is defined as: <ol style="list-style-type: none"><li>Swiping and processing of a card.</li><li>Receipt and processing of commands from a Host.</li><li>Briefly pressing the User Switch.</li></ol>

When the specified time passes without activity, the reader is powered down.

This property is stored in non-volatile memory, so it will persist when the unit is power cycled. When this property is changed, the unit must be reset (see Command Number 2) or power cycled to have these changes take effect.

### STAY POWERED AFTER SWIPE PROPERTY (WIRELESS READER ONLY)

Property ID:	0x0E
Property Type:	Byte
Length:	1 byte
Get Property:	Yes
Set Property:	Yes
Default Value:	0x00 (Don't Stay Powered)
Description:	This property controls whether the wireless reader stays powered after a good swipe. If the property value is 0x00 (the default), the reader powers down after a good swipe.

If the property value is 0x01, the reader stays powered after a good swipe. In this case, the reader may be powered down by pressing and holding the User Switch or it will go off after the activity timeout.

This property is stored in non-volatile memory, so it will persist when the unit is power cycled. When this property is changed, the unit must be reset (see Command Number 2) or power cycled to have these changes take effect.



**RF ADDRESS PROPERTY (WIRELESS READER ONLY)**

Property ID: 0x0F  
 Property Type: Binary  
 Length: 4 bytes  
 Get Property: Yes  
 Set Property: Yes  
 Default Value: 0x12345678 (hex)  
 Description: The value is a 4-byte number that represents the wireless reader's and dongle's RF address. The RF address is used to pair a wireless reader to a dongle. All reader/dongle pairs should have a different RF address from all other reader/dongle pairs. This unique address is what allows more than one pair to be in range of another pair without having RF communications conflict with each other.

This property should be the first property changed so that all other communications will not conflict with other pairs that may be in range. After this property is changed, the reader should be reset (see Command Number 2) before changing any other properties. While this property is being set, make sure that there are no other wireless readers in range that are on and that have the same RF address as could possibly be the case if two new readers that are using the default RF address are on at the same time and in range.

This property should be set during the manufacturing process at the time the reader and dongle are paired together. Additionally, the read and dongle pair should be labeled with a unique matching identifier so that they can be visually distinguished from any other pair.

This property is stored in non-volatile memory, so it will persist when the unit is power cycled. When this property is changed, the unit must be reset (see Command Number 2) or power cycled to have these changes take effect.

Example Set **RF Address** property to 0x9ABCDEF0 (Hex):

Cmd Num	Data Len	Prp ID	Prp Value
01	05	0F	9ABCDEF0

Example Set **RF Address** property Response (Hex):

Result Code	Data Len	Data
00	00	

Example Get **RF Address** property Request (Hex):

Cmd Num	Data Len	Prp ID
00	01	0F

Example Get **RF Address** property Response (Hex) (when in HID type):

Result Code	Data Len	Prp Value
00	04	9ABCDEF0

## INTERFACE TYPE PROPERTY

Property ID: 0x10  
 Property Type: Byte  
 Length: 1 byte  
 Get Property: Yes  
 Set Property: Yes  
 Default Value: 0 (HID)  
 Description: The value is a byte that represents the reader's interface type. The value can be set to 0 for the HID interface or to 1 for the Keyboard Emulation interface. When the value is set to 0 (HID) the reader will behave as described in the HID manual. When the value is set to 1 (keyboard emulation) the reader will behave as described in the keyboard emulation manual. This property should be the first property changed because it affects which other properties are available. After this property is changed, the reader should be power cycled before changing any other properties.

This property is stored in non-volatile memory, so it will persist when the unit is power cycled. When this property is changed, the unit must be reset (see Command Number 2) or power cycled to have these changes take effect.

Example Set **Interface Type** property to Keyboard Request (Hex):

Cmd Num	Data Len	Prp ID	Prp Value
01	02	10	01

Example Set **Interface Type** property Response (Hex):

Result Code	Data Len	Data
00	00	

Example Get **Interface Type** property Request (Hex):

Cmd Num	Data Len	Prp ID
00	01	10

Example Get **Interface Type** property Response (Hex) (when in HID type):

Result Code	Data Len	Prp Value
00	01	00

## TRACK DATA SEND FLAGS PROPERTY (KB)

Property ID: 0x14  
 Property Type: Byte  
 Length: 1 byte  
 Get Property: Yes  
 Set Property: Yes  
 Default Value: 0x63  
 Description: This property is defined as follows:

ICL	SS	ES	LRC	0	LC	Er	Er
-----	----	----	-----	---	----	----	----

- ICL 0 – Changing the state of the caps lock key will not affect the case of the data  
1 – Changing the state of the caps lock key will affect the case of the data
- SS 0 – Don't send Start Sentinel for each track  
1 – Send Start Sentinel for each track
- ES 0 – Don't send End Sentinel for each track  
1 – Send End Sentinel for each track
- LRC 0 – Don't send LRC for each track  
1 – Send LRC for each track

Note that the LRC is the unmodified LRC from the track data. To verify the LRC the track data needs to be converted back from ASCII to card data format and the start sentinels that were modified to indicate the card encode type need to be converted back to their original values.

Note that this property only applies to track data sent via the keyboard interface and completely in the clear (Security Levels 1 & 2).

- LC 0 – Send card data as upper case  
1 – Send card data as lower case

Note that the state of the Caps Lock key on the host keyboard has no affect on what case the card data is transmitted in unless the ICL bit in this property is set to 1.

- Er 00 – Don't send any card data if error – NOT CURRENTLY IMPLEMENTED  
01 – Don't send track data if error  
11 – Send 'E' for each track error

This property is stored in non-volatile memory, so it will persist when the unit is power cycled. When this property is changed, the unit must be reset (see Command Number 2) or power cycled to have these changes take effect.

### MP FLAGS PROPERTY (KB)

Property ID: 0x15  
 Property Type: Byte  
 Length: 1 byte  
 Get Property: Yes  
 Set Property: Yes  
 Default Value: 0x01  
 Description: This property is defined as follows:

0	0	0	0	0	0	0	0	S
---	---	---	---	---	---	---	---	---

- S 0 – MagnePrint Data will NOT be sent

1 – MagnePrint Data will be sent.

This property is used to designate whether or not the MagnePrint data is sent as part of a keyboard message. Setting S to 1 causes the MagnePrint Status and Encrypted MagnePrint Data to be sent with each swipe. Setting S to 0 causes these fields to be omitted from the data. When the fields are omitted, the Programmable Field Separator that precedes each of these fields will also be omitted.

This property is stored in non-volatile memory, so it will persist when the unit is power cycled. When this property is changed, the unit must be reset (see Command Number 2) or power cycled to have these changes take effect.

### ACTIVE KEYMAP PROPERTY (KB)

Property ID: 0x16  
Property Type: Byte  
Length: 1 byte  
Get Property: Yes  
Set Property: Yes  
Default Value: 0 (United States)

Description: The value is a byte that represents the reader's active key map. The value can be set to 0 for the United States key map or to 1 for the custom key map. The active key map will be used by the reader to convert ASCII data into key strokes. The United States key map should be used will all hosts that are configured to use United States keyboards. The custom key map can be used to set up the reader to work with hosts that are configured to use other countries keyboards. The default custom key map is the same as the United States key map. The key map can be modified to another countries key map by using commands **Get Key Map**, **Set Key Map** and **Save Custom Key Map**.

See the command section of this manual for a complete description of these commands. To set up a reader to use a custom key map, select the appropriate key map to be modified using the active key map property, reset the reader to make this change take affect, use the **Get Key Map** and **Set Key Map** commands to modify the active key map, use the "Save Custom Key Map" command to save the active key map as the custom key map, set the active key map property to custom to use the custom key map, reset the reader to make these changes take affect.

This property is stored in non-volatile memory, so it will persist when the unit is power cycled. When this property is changed, the unit must be reset (see Command Number 2) or power cycled to have these changes take effect.

Example Set **Active Keymap** property Request (Hex):

Cmd Num	Data Len	Prp ID	Prp Value
01	02	16	00

Example Set **Active Keymap** property Response (Hex):

Result Code	Data Len	Data
00	00	

Example Get **Active Keymap** property Request (Hex):

Cmd Num	Data Len	Prp ID
00	01	16

Example Get **Active Keymap** property Response (Hex):

Result Code	Data Len	Prp Value
00	01	00

## ASCII TO KEYPRESS CONVERSION TYPE PROPERTY (KB)

Property ID: 0x17

Property Type: Byte

Length: 1 byte

Get Property: Yes

Set Property: Yes

Default Value: 0 (keymap)

Description: The value is a byte that represents the reader's ASCII-to-keypress conversion type. The value can be set to 0 for keymap (the active keymap is set with the ACTIVE KEYMAP property) or to 1 for ALT ASCII code (international keyboard emulation). When the value is set to 0 (keymap), data will be transmitted to the host according to the active keymap which defaults to the United States keyboard keymap. For example, to transmit the ASCII character '?' (063 decimal), the character is looked up in a keymap. For a United States keyboard keymap, the '/' (forward slash) key combined with the left shift key modifier are stored in the keymap to represent the key press combination that is used to represent the ASCII character '?' (063 decimal).

When the value is set to 1 (ALT ASCII code), instead of using the key map, a international keyboard key press combination consisting of the decimal value of the ASCII character combined with the ALT key modifier is used. For example, to transmit the ASCII character '?' (063 decimal), keypad '0' is sent combined with left ALT key modifier, next keypad '6' is sent combined with the left ALT key modifier, last keypad '3' is sent combined with the left ALT key modifier. In general, if this reader only needs to emulate United States keyboards then this property should be set to 0 (keymap). If this reader needs to be able to emulate all country's keyboards then this property should be set to 1 (ALT ASCII code). The tradeoff is that the ALT ASCII code mode is slightly slower than keymap mode because more key presses need to be transmitted. Some applications are not compatible with ALT ASCII code mode.

This property is stored in non-volatile memory, so it will persist when the unit is power cycled. When this property is changed, the unit must be reset (see Command Number 2) or power cycled to have these changes take effect.

Example Set **ASCII To Keypress Conversion Type** property Request (Hex):

Cmd Num	Data Len	Prp ID	Prp Value
01	02	17	00

Example Set **ASCII To Keypress Conversion Type** property Response (Hex):

Result Code	Data Len	Data
00	00	

Example Get **ASCII To Keypress Conversion Type** property Request (Hex):

Cmd Num	Data Len	Prp ID
00	01	17

Example Get **ASCII To Keypress Conversion Type** property Response (Hex):

Result Code	Data Len	Prp Value
00	01	00

### CRC FLAG PROPERTY (KB)

Property ID: 0x19  
 Property Type: Byte  
 Length: 1 byte  
 Get Property: Yes  
 Set Property: Yes  
 Default Value: 0x01

Description: This property is defined as follows:

0	0	0	0	0	0	E	S
---	---	---	---	---	---	---	---

- E 0 – The Encrypted CRC will NOT be sent  
 1 – The Encrypted CRC will be sent
- S 0 – The Clear Text CRC will NOT be sent  
 1 – The Clear Text CRC will be sent.

This property is used to designate whether or not the calculated CRC is sent as part of a keyboard message. The default state of this property causes only the Clear Text CRC to be sent. When the fields are omitted, the Programmable Field Separator that precedes each of these fields will be sent anyhow.

This property is stored in non-volatile memory, so it will persist when the unit is power cycled. When this property is changed, the unit must be reset (see Command Number 2) or power cycled to have these changes take effect.

**SURESWIPE FLAG PROPERTY (KB)**

Property ID: 0x1A  
 Property Type: Byte  
 Length: 1 byte  
 Get Property: Yes  
 Set Property: Yes  
 Default Value: 0x01  
 Description: This property enables/disables SureSwipe emulation when the Security Level is 2 and the Interface Type is Keyboard. The default is SureSwipe emulation enabled, keyboard data will be emitted in the SureSwipe format (see MagTek document 99875206). This allows clients to receive a reader without security enabled (Security Level 2) and use it exactly like a SureSwipe reader. Later, when a client is ready, they can switch the reader to a higher Security Level and take advantage of the robust security features offered by the reader. A value of 0x01 enables SureSwipe emulation, a value of 0x00 disables it. A user might disable SureSwipe emulation to allow the reader to emit keyboard data in the full V5 format without encryption. This could allow the user to develop software that works with this format without worrying about cryptography at the start.

This property is stored in non-volatile memory, so it will persist when the unit is power cycled. When this property is changed, the unit must be reset (see Command Number 2) or power cycled to have these changes take effect.

**PRE CARD STRING PROPERTY (KB)**

Property ID: 0x1E  
 Property Type: String  
 Length: 0 – 7 bytes  
 Get Property: Yes  
 Set Property: Yes  
 Default Value: The default value is no string with a length of zero.  
 Description: The value is an ASCII string that represents the reader's pre card string. This string can be 0 – 7 bytes long. This string is sent prior to all other card data.

This property is stored in non-volatile memory, so it will persist when the unit is power cycled. When this property is changed, the unit must be reset (see Command Number 2) or power cycled to have these changes take effect.

Example Set **Pre Card String** property Request (Hex):

Cmd Num	Data Len	Prp ID	Prp Value
01	04	1E	31 32 33

Example Set **Pre Card String** property Response (Hex):

Result Code	Data Len	Data
00	00	

Example Get **Pre Card String** property Request (Hex):

Cmd Num	Data Len	Prp ID
00	01	1E

Example Get **Pre Card String** property Response (Hex):

Result Code	Data Len	Prp Value
00	03	31 32 33

## POST CARD STRING PROPERTY (KB)

Property ID: 0x1F  
 Property Type: String  
 Length: 0 – 7 bytes  
 Get Property: Yes  
 Set Property: Yes  
 Default Value: The default value is no string with a length of zero.  
 Description: The value is an ASCII string that represents the reader’s post card string. This string can be 0 – 7 bytes long. This string is sent after all other card data.

This property is stored in non-volatile memory, so it will persist when the unit is power cycled. When this property is changed, the unit must be reset (see Command Number 2) or power cycled to have these changes take effect.

Example Set **Post Card String** property Request (Hex):

Cmd Num	Data Len	Prp ID	Prp Value
01	04	1F	31 32 33

Example Set **Post Card String** property Response (Hex):

Result Code	Data Len	Data
00	00	

Example Get **Post Card String** property Request (Hex):

Cmd Num	Data Len	Prp ID
00	01	1F

Example Get **Post Card String** property Response (Hex):

Result Code	Data Len	Prp Value
00	03	31 32 33



**PRE TRACK STRING PROPERTY (KB)**

Property ID: 0x20  
 Property Type: String  
 Length: 0-7 bytes  
 Get Property: Yes  
 Set Property: Yes  
 Default Value: No string with a length of zero.  
 Description: This string is sent prior to the data for each track. The string can be 0 – 7 bytes long. If the value is 0 no character is sent.

This property is stored in non-volatile memory, so it will persist when the unit is power cycled. When this property is changed, the unit must be reset (see Command Number 2) or power cycled to have these changes take effect.

Example Set **Pre Track String** property Request (Hex):

Cmd Num	Data Len	Prp ID	Prp Value
01	04	20	31 32 33

Example Set **Pre Track String** property Response (Hex):

Result Code	Data Len	Data
00	00	

Example Get **Pre Track String** property Request (Hex):

Cmd Num	Data Len	Prp ID
00	01	20

Example Get **Pre Track String** property Response (Hex):

Result Code	Data Len	Prp Value
00	03	31 32 33

**POST TRACK STRING PROPERTY (KB)**

Property ID: 0x21  
 Property Type: String  
 Length: 0-7 bytes  
 Get Property: Yes  
 Set Property: Yes  
 Default Value: No string with a length of zero  
 Description: This string is sent after the data for each track. The string can be 0 – 7 bytes long. If the value is 0 no character is sent.

This property is stored in non-volatile memory, so it will persist when the unit is power cycled. When this property is changed, the unit must be reset (see Command Number 2) or power cycled to have these changes take effect.

Example Set **Post Track String** property Request (Hex):

Cmd Num	Data Len	Prp ID	Prp Value
01	04	21	31 32 33

Example Set **Post Track String** property Response (Hex):

Result Code	Data Len	Data
00	00	

Example Get **Post Track String** property Request (Hex):

Cmd Num	Data Len	Prp ID
00	01	21

Example Get **Post Track String** property Response (Hex):

Result Code	Data Len	Prp Value
00	03	31 32 33

### TERMINATION STRING PROPERTY (KB)

Property ID: 0x22  
 Property Type: String  
 Length: 0-7 bytes  
 Get Property: Yes  
 Set Property: Yes  
 Default Value: 0x0D (carriage return)  
 Description: This string is sent after the all the data for a transaction. The string can be 0 – 7 bytes long. If the value is 0 no character is sent.

This property is stored in non-volatile memory, so it will persist when the unit is power cycled. When this property is changed, the unit must be reset (see Command Number 2) or power cycled to have these changes take effect.

### FS PROPERTY (KB)

Property ID: 0x23  
 Property Type: Byte  
 Length: 1 byte  
 Get Property: Yes  
 Set Property: Yes  
 Default Value: 0x7C ‘|’  
 Description: This character is sent as the field separator to delimit additional data (MagnePrint info, reader info, DUKPT info, etc.). If the value is 0 no character is sent. If the value is in the range 1 – 127 then the equivalent ASCII character will be sent.

This property is stored in non-volatile memory, so it will persist when the unit is power cycled. When this property is changed, the unit must be reset (see Command Number 2) or power cycled to have these changes take effect.

**SS TK1 ISO ABA PROPERTY (KB)**

Property ID: 0x24  
 Property Type: Byte  
 Length: 1 byte  
 Get Property: Yes  
 Set Property: Yes  
 Default Value: 0x25 '%'

Description: This character is sent as the track 1 start sentinel for cards that have track 1 encoded in ISO/ABA format. If the value is 0 no character is sent. If the value is in the range 1 – 127 then the equivalent ASCII character will be sent.

This property is stored in non-volatile memory, so it will persist when the unit is power cycled. When this property is changed, the unit must be reset (see Command Number 2) or power cycled to have these changes take effect.

**SS TK2 ISO ABA PROPERTY (KB)**

Property ID: 0x25  
 Property Type: Byte  
 Length: 1 byte  
 Get Property: Yes  
 Set Property: Yes  
 Default Value: 0x3B ';'

Description: This character is sent as the track 2 start sentinel for cards that have track 2 encoded in ISO/ABA format. If the value is 0 no character is sent. If the value is in the range 1 – 127 then the equivalent ASCII character will be sent.

This property is stored in non-volatile memory, so it will persist when the unit is power cycled. When this property is changed, the unit must be reset (see Command Number 2) or power cycled to have these changes take effect.

**SS TK3 ISO ABA PROPERTY (KB)**

Property ID: 0x26  
 Property Type: Byte  
 Length: 1 byte  
 Get Property: Yes  
 Set Property: Yes  
 Default Value: 0x2B ('+')

Description: This character is sent as the track 3 start sentinel for cards that have track 3 encoded in ISO/ABA format. If the value is 0 no character is sent. If the value is in the range 1 – 127 then the equivalent ASCII character will be sent.

This property is stored in non-volatile memory, so it will persist when the unit is power cycled. When this property is changed, the unit must be reset (see Command Number 2) or power cycled to have these changes take effect.

### SS TK3 AAMVA PROPERTY (KB)

Property ID: 0x27  
Property Type: Byte  
Length: 1 byte  
Get Property: Yes  
Set Property: Yes  
Default Value: 0x23 ('#')  
Description: This character is sent as the track 3 start sentinel for cards that have track 3 encoded in AAMVA format. If the value is 0 no character is sent. If the value is in the range 1 – 127 then the equivalent ASCII character will be sent.

This property is stored in non-volatile memory, so it will persist when the unit is power cycled. When this property is changed, the unit must be reset (see Command Number 2) or power cycled to have these changes take effect.

### SS TK2 7BITS PROPERTY (KB)

Property ID: 0x28  
Property Type: Byte  
Length: 1 byte  
Get Property: Yes  
Set Property: Yes  
Default Value: 0x40 ('@')  
Description: This character is sent as the track 2 start sentinel for cards that have track 2 encoded in 7 bits per character format. If the value is 0 no character is sent. If the value is in the range 1 – 127 then the equivalent ASCII character will be sent.

This property is stored in non-volatile memory, so it will persist when the unit is power cycled. When this property is changed, the unit must be reset (see Command Number 2) or power cycled to have these changes take effect.

### SS TK3 7BITS PROPERTY (KB)

Property ID: 0x29  
Property Type: Byte  
Length: 1 byte  
Get Property: Yes  
Set Property: Yes  
Default Value: 0x26 ('&')  
Description: This character is sent as the track 3 start sentinel for cards that have track 3 encoded in 7 bits per character format. If the value is 0 no character is sent. If the value is in the range 1 – 127 then the equivalent ASCII character will be sent.

This property is stored in non-volatile memory, so it will persist when the unit is power cycled. When this property is changed, the unit must be reset (see Command Number 2) or power cycled to have these changes take effect.

### ES PROPERTY (KB)

Property ID: 0x2B  
 Property Type: Byte  
 Length: 1 byte  
 Get Property: Yes  
 Set Property: Yes  
 Default Value: 0x3F ('?')  
 Description: This character is sent as the end sentinel for all tracks with any format. If the value is 0 no character is sent. If the value is in the range 1 – 127 then the equivalent ASCII character will be sent.

This property is stored in non-volatile memory, so it will persist when the unit is power cycled. When this property is changed, the unit must be reset (see Command Number 2) or power cycled to have these changes take effect.

### FORMAT CODE PROPERTY (KB)

Property ID: 0x2C  
 Property Type: String  
 Length: 4 bytes  
 Get Property: Yes  
 Set Property: Yes  
 Default Value: "0000"  
 Description: This property is defined as follows:

This property specifies the Format Code that will be returned at the end of a transmitted card swipe. The application sends four characters, but only the last three will be set. The first character is reserved for MagTek use. A value of '0' in the first character means the Format Code is defined by MagTek. A value of '1' in the first character means the Format Code is user defined.

This property is stored in non-volatile memory, so it will persist when the unit is power cycled. When this property is changed, the unit must be reset (see Command Number 2) or power cycled to have these changes take effect.

### ES TRACK 1 PROPERTY

Property ID: 0x2D  
Property Type: Byte  
Length: 1 byte  
Get Property: Yes  
Set Property: Yes  
Default Value: 0xFF (use ES property)  
Description: This character is sent as the end sentinel for track 1 with any format. If the value is 0 no character is sent. If the value is in the range 1 – 127 then the equivalent ASCII character will be sent. If the value is 0xFF then the value of the ES property will be used instead of this property.

This property is stored in non-volatile memory, so it will persist when the unit is power cycled. When this property is changed, the unit must be reset (see Command Number 2) or power cycled to have these changes take effect.

### ES TRACK 2 PROPERTY

Property ID: 0x2E  
Property Type: Byte  
Length: 1 byte  
Get Property: Yes  
Set Property: Yes  
Default Value: 0xFF (use ES property)  
Description: This character is sent as the end sentinel for track 2 with any format. If the value is 0 no character is sent. If the value is in the range 1 – 127 then the equivalent ASCII character will be sent. If the value is 0xFF then the value of the ES property will be used instead of this property.

This property is stored in non-volatile memory, so it will persist when the unit is power cycled. When this property is changed, the unit must be reset (see Command Number 2) or power cycled to have these changes take effect.

### ES TRACK 3 PROPERTY

Property ID: 0x2F  
Property Type: Byte  
Length: 1 byte  
Get Property: Yes  
Set Property: Yes  
Default Value: 0xFF (use ES property)  
Description: This character is sent as the end sentinel for track 3 with any format. If the value is 0 no character is sent. If the value is in the range 1 – 127 then the equivalent ASCII character will be sent. If the value is 0xFF then the value of the ES property will be used instead of this property.

This property is stored in non-volatile memory, so it will persist when the unit is power cycled. When this property is changed, the unit must be reset (see Command Number 2) or power cycled to have these changes take effect.

### SEND ENCRYPTION COUNTER (KB)

Property ID: 0x30  
 Property Type: Byte  
 Length: 1 byte  
 Get Property: Yes  
 Set Property: Yes  
 Default Value: 0x00 (don't send Encryption Counter)  
 Description: This property is used to designate whether or not the Encryption Counter is sent as part of a keyboard message. If the property is set to 0x00, the Encryption Counter is not sent, neither is a field separator sent. If the property is set to 0x01, the Encryption Counter is sent as the next field after the DUKPT Serial Number in a swipe message.

*NOTE: If this property is set to 0x01 and the Format Code is currently "0001", the Format Code will be changed to "0002".*

This property is stored in non-volatile memory, so it will persist when the unit is power cycled. When this property is changed, the unit must be reset (see Command Number 2) or power cycled to have these changes take effect.

### MASK OTHER CARDS

Property ID: 0x31  
 Property Type: Byte  
 Length: 1 byte  
 Get Property: Yes  
 Set Property: Yes  
 Default Value: 0x00 (Don't Mask Other cards)  
 Description: This property is used to designate whether or not the cards which do not decode as ISO/ABA Financial cards or AAMVA Driver License cards should be sent with their data masked or in the clear. The default state is to send the data in the clear (0x00). If this property is set to 0x01, the track(s) will be sent with a "0" for each byte of encoded data read.

This property is stored in non-volatile memory, so it will persist when the unit is power cycled. When this property is changed, the unit must be reset (see Command Number 2) or power cycled to have these changes take effect.

### MSR DIRECTION PROPERTY (INSERT READER ONLY)

Property ID: 0x32  
 Property Type: Byte  
 Length: 1 byte  
 Get Property: Yes  
 Set Property: Yes  
 Default Value: 2 (Withdrawal)  
 Description: This value is a byte that represents the device's magnetic stripe read direction. The device will output card data when a card is swiped in the direction indicated by this property. The value can be set to 1 for insert, 2 for withdrawal or 3 for both directions. (This reader reads better when a card is removed from it than when a card is inserted into it.)

This property is stored in non-volatile memory, so it will persist when the unit is power cycled. When this property is changed, the unit must be reset (see Command Number 2) or power cycled to have these changes take effect.

Examples follow:

Example Set **MSR Direction** property Request (Hex):

Cmd Num	Data Len	Prp ID	Prp Value
01	02	32	02

Example Set **MSR Direction** property Response (Hex):

Result Code	Data Len	Data
00	00	

Example Get **MSR Direction** property Request (Hex):

Cmd Num	Data Len	Prp ID
00	01	32

Example Get **MSR Direction** property Response (Hex):

Result Code	Data Len	Prp Value
00	01	02



**CARD INSERTED PROPERTY (INSERT READER ONLY)**

Property ID:	0x33
Property Type:	Byte
Length:	1 byte
Get Property:	Yes
Set Property:	No
Default Value:	None
Description:	This value is used to determine if a card is fully inserted into the device. If a card is fully inserted into the device, this property will contain 0x01. If not, the property will contain 0x00. This property is intended to be used by hosts that want to check if a card is currently inserted in the device during startup. This card inserted information is also contained in the Card Status field of the HID Input report sent to the host during each card swipe when using the HID interface type. If the reader is configured to only output swipe data in one direction, as it is by default, then the value of this HID report field will always be the same.

Example Get **Card Inserted** property Request (Hex):

Cmd Num	Data Len	Prp ID
00	01	33

Example Get **Card Inserted** property Response (Hex):

Result Code	Data Len	Prp Value
00	01	01

**SEND CLEAR AAMVA CARD DATA PROPERTY**

Property ID:	0x34
Property Type:	Byte
Length:	1 byte
Get Property:	Yes
Set Property:	Yes
Default Value:	0x00
Description:	This character is used to control how to send out AAMVA card data when the security level is above 2.

This property is stored in non-volatile memory, so it will persist when the unit is power cycled. When this property is changed, the unit must be reset (see Command Number 2) or power cycled to have these changes take effect.

- 0 – send out masked AAMVA card data
- 1 – send out clear AAMVA card data

Example Set **Send Clear AAMVA Card Data** property Request (Hex):

Cmd Num	Data Len	Prp ID	Data
01	06	34	01 xx xx xx xx *

\* where “xx xx xx xx” is the MAC.

Example Set **Send Clear AAMVA Card Data** property Response (Hex):

Result Code	Data Len	Data
00	00	

Example Get **Send Clear AAMVA Card Data** property Request (Hex):

Cmd Num	Data Len	Prp ID
00	01	34

Example Get **Send Clear AAMVA Card Data** property Response (Hex):

Result Code	Data Len	Data
00	01	01

## HID SURESWIPE FLAG PROPERTY (HID)

Property ID: 0x38

Property Type: Byte

Length: 1 byte

Get Property: Yes

Set Property: Yes

Default Value: 0x00

Description: This property controls whether, when the reader is configured with Interface Type HID and at Security Level 2, the reader functions as described in this manual or as described in 99875191 (USB HID SURESWIPE & USB HID SWIPE READER TECHNICAL REFERENCE MANUAL).

When this property is set to 0x00, the reader functions as described in this document.

When this property is set to 0x01, the reader functions as described in 99875191 (USB HID SURESWIPE & USB HID SWIPE READER TECHNICAL REFERENCE MANUAL). It returns card swipes as defined in that document and enumerates with the same VID/PID as defined in the SureSwipe document.

This property is stored in non-volatile memory, so it will persist when the unit is power cycled. When this property is changed, the unit must be reset (see Command Number 2) or power cycled to have these changes take effect.

Example Set **HID SureSwipe Flag** property Request (Hex):

Cmd Num	Data Len	Prp ID	Data
01	06	38	01

Example Set **HID SureSwipe Flag** property Response (Hex):

Result Code	Data Len	Data
00	00	

Example Get **HID SureSwipe Flag** property Request (Hex):

Cmd Num	Data Len	Prp ID
00	01	38

Example Get **HID SureSwipe Flag** property Response (Hex):

Result Code	Data Len	Data
00	01	01

## SOFTWARE ID 2 PROPERTY (WIRELESS READER ONLY)

Property ID: 0x3A

Property Type: String

Length: Fixed at 11 bytes

Get Property: Yes

Set Property: No

Description: This is an 11 byte read only property that identifies the software part number and version for the reader. The first 8 bytes represent the part number and the last 3 bytes represent the version. For example, this string might be “21042812D01”.

For the wireless reader, two software IDs are available, one in the dongle and one in the wireless reader. This property returns the software ID from the wireless reader. To get the software ID from the dongle use the “SOFTWARE ID” property.

Example Get **Software ID 2** property Request (Hex):

Cmd Num	Data Len	Prp ID
00	01	3A

Example Get **Software ID 2** property Response (Hex):

Result Code	Data Len	Prp Value
00	01	32 31 30 34 32 38 31 32 44 30 31

## RESET DEVICE COMMAND

Command number: 0x02

Description: This command is used to reset the reader. This command can be used to make previously changed properties take affect without having to unplug and then plug in the reader. When the reader resets, it automatically does a USB detach followed by an attach. After the host sends this command to the reader it should close the USB port, wait a few seconds for the operating system to handle the reader detach followed by the attach and then re-open the USB port before trying to communicate further with the reader.

When an Authentication sequence has failed, only a correctly MACed (See Section 2, Security) Reset command can be used to Reset the reader. This prevents a dictionary attack on the on the keys and minimizes a denial of service attack.

**Note**

*When the reader begins an Authentication Sequence, the Reset command will not be honored until after the Authentication Sequence has successfully completed, the user swipes a card, or the unit is power cycled.*

Data structure: No data is sent with this command  
 Result codes: 0x00 (Success)  
 0x01 (Failure)  
 0x07 (Incorrect MAC – Command not authorized)

Example **Reset Device Request (Hex):**

Cmd Num	Data Len	Data
02	00	

Example **Reset Device Response (Hex):**

Result Code	Data Len	Data
00	00	

**GET KEYMAP ITEM COMMAND (KB)**

Command number: 0x03

Description: This command is used to get a key map item from the active key map. The active key map is determined by the active key map property. Data from a magnetic stripe card is a sequence of ASCII characters. These ASCII characters are mapped to key strokes and these key strokes are sent to the host to represent the ASCII character. The key map maps a single ASCII character to a single USB key usage ID and USB key modifier byte. The key usage ID and the key modifier byte are transmitted to the host via USB to represent the ASCII character. The ASCII value is the value of the ASCII character to be transmitted to the host. See an ASCII table for the values of the ASCII character set. The USB key usage ID is a unique value assigned to every keyboard key. For a list of all key usage IDs see Appendix A. The key modifier byte modifies the meaning of the key usage ID. The modifier byte indicates if any combination of the right or left Ctrl, Shift, Alt or GUI keys are pressed at the same time as the key usage ID. For a list and description of the key modifier byte see Appendix B.

Starting with the firmware release with software ID 21042812F01, when both the key usage ID and the key modifier byte are set to 0xFF for a given ASCII value, the ALT ASCII code is sent instead of the key map

values. The ALT ASCII code is a key press combination consisting of the decimal value of the ASCII character combined with the ALT key modifier. For example, to transmit the ASCII character '?' (063 decimal), keypad '0' is sent combined with left ALT key modifier, next keypad '6' is sent combined with the left ALT key modifier, last keypad '3' is sent combined with the left ALT key modifier.

Data structure:

Request Data:

Offset	Field Name	Description
0	ASCII value	Value of the ASCII character to be retrieved from the key map. This can be any value between 0 and 127 (0x7F). For example, to retrieve the key map item for ASCII character '?' (card data end sentinel) use the ASCII value of '?' which is 63 (0x3F).

Response Data:

Offset	Field Name	Description
0	Key Usage ID	The value of the USB key usage ID that is mapped to the given ASCII value. For example, for the United States keyboard map, usage ID 56 (0x38) (keyboard / and ?) is mapped to ASCII character '?'.
1	Key Modifier Byte	The value of the USB key modifier byte that is mapped to the given ASCII value. For example, for the United States keyboard map, modifier byte 0x02 (left shift key) is mapped to ASCII character '?'.

Result codes:           0x00 (Success)

Example Request (Hex):

Cmd Num	Data Len	Data
03	01	3F

Example Response (Hex):

Result Code	Data Len	Data
00	02	38 02

## SET KEYMAP ITEM COMMAND (KB)

Command number: 0x04

Description: This command is used to set a key map item of the active key map. The active key map is determined by the active key map property. Data from a magnetic stripe card is a sequence of ASCII characters. These ASCII characters are mapped to key strokes and these key strokes are sent to the host to represent the ASCII character. The key map maps a single ASCII character to a single USB key usage ID and USB key modifier byte. The key usage ID and the key modifier byte are transmitted to the host via USB to represent the ASCII character. The ASCII value is the value of the ASCII character to be transmitted to the host. See an ASCII table for the values of the ASCII character set. The USB key usage ID is a unique value assigned to every keyboard key. For a list of all key usage IDs see Appendix A. The key modifier byte modifies the meaning of the key usage ID. The modifier byte indicates if any combination of the right or left Ctrl, Shift, Alt or GUI keys are pressed at the same time as the key usage ID. For a list and description of the key modifier byte see Appendix B. Once a key map item is modified, the changes take affect immediately. However, the changes will be lost if the reader is reset or power cycled. To make the changes permanent, the save custom key map command must be issued. To use the new custom key map after a reset or power cycle, the active key map property must be set to custom.

Starting with the firmware release with software ID 21042812F01, when both the key usage ID and the key modifier byte are set to 0xFF for a given ASCII value, the ALT ASCII code is sent instead of the key map values. The ALT ASCII code is a key press combination consisting of the decimal value of the ASCII character combined with the ALT key modifier. For example, to transmit the ASCII character '?' (063 decimal), keypad '0' is sent combined with left ALT key modifier, next keypad '6' is sent combined with the left ALT key modifier, last keypad '3' is sent combined with the left ALT key modifier.

Data structure:

Request Data:

Offset	Field Name	Description
0	ASCII value	Value of the ASCII character to be set in the key map. This can be any value between 0 and 127 (0x7F). For example, to set the key map item for ASCII character '?' (card data end sentinel) use the ASCII value of '?' which is 63 (0x3F).
1	Key Usage ID	The value of the USB key usage ID that is to be mapped to the given ASCII value. For example, for the United States keyboard map, usage ID 56 (0x38) (keyboard / and ?) is mapped to ASCII character '?'. To change this to the ASCII character '>' use usage ID 55 (0x37) (keyboard . and >).
2	Key Modifier Byte	The value of the USB key modifier byte that is to be mapped to the given ASCII value. For example, for the United States keyboard map, modifier byte 0x02 (left shift key) is mapped to ASCII character '?'. To change this to the ASCII character '>' use modifier byte 0x02 (left shift key).

Response Data: None

Result codes:      0x00 (Success)  
                       0x07 (Incorrect MAC – Command not authorized)

The following example maps the card ASCII data end sentinel character '?' to the '>' keyboard key.

Example **Set Keymap Item** Request (Hex):

Cmd Num	Data Len	Data
04	03	3F 37 02

Example **Set Keymap Item** Response (Hex):

Result Code	Data Len	Data
00	00	

## SAVE CUSTOM KEYMAP COMMAND (KB)

Command number: 0x05

Description: This command is used to save the active key map as the custom key map in non volatile memory. The active key map is determined by the active key map property. Once a key map item is modified, the changes take affect immediately. However, the changes will be lost if the reader is reset or power cycled. To make the changes permanent, the save custom key map command must be issued. To use the new custom key map after a reset or power cycle, the active key map property must be set to custom.

Data structure:

Request Data: None  
 Response Data: None

Result codes: 0x00 (Success)  
 0x07 (Incorrect MAC – Command not authorized)

Example **Save Custom Keymap Request (Hex):**

Cmd Num	Data Len	Data
05	00	

Example **Save Custom Keymap Response (Hex):**

Result Code	Data Len	Data
00	00	

## DUKPT OPERATION

Since key loading is proprietary and performed at MagTek, there are no user commands to support key injection.

### Get DUKPT KSN and Counter

Command number: 0x09

Description: This command is used to report the Key Serial Number and Encryption Counter.

Data structure: No data is sent with this command.

Response Data:

Offset	Field Name	Description
0	Current Key Serial Number	This eighty-bit field includes the Initial Key Serial Number in the leftmost 59 bits and a value for the Encryption Counter in the rightmost 21 bits.

Result codes: 0x00 (Success)  
 0x02 (Bad Parameters – the Request Data is not a correct length)



Example **Get DUKPT KSN and Counter** Request (Hex):

Cmd Num	Data Len	Data
09	0	None

Example **Get DUKPT KSN and Counter** Response (Hex):

Result Code	Data Len	Data
00	0A	FFFF 9876 5432 10E0 0001

## SET SESSION ID COMMAND

Command number: 0x0A

Description: This command is used to set the current Session ID. The new Session ID stays in effect until one of the following occurs:

1. Another Set Session ID command is received.
2. The reader is powered down.
3. The reader is put into Suspend mode.

The Session ID is used by the host to uniquely identify the present transaction. Its primary purpose is to prevent replays. After a card is read, the Session ID will be encrypted, along with the card data, a supplied as part of the transaction message. The clear text version of this will never be transmitted.

Data structure:

Request Data:

Offset	Field Name	Description
0	New Session ID	This eight byte field may contain any value the user wishes.

Response Data: None

Result codes: 0x00 (Success)  
0x02 (Bad Parameters – the Request Data is not a correct length)

Example **Set Session ID** Request (Hex):

Cmd Num	Data Len	Data
0A	08	54 45 53 54 54 45 53 54

Example **Set Session ID** Response (Hex):

Result Code	Data Len	Data
00	00	

## ACTIVATE AUTHENTICATED MODE COMMAND

Command number: 0x10

Description: This command is used to Activate the Authenticated Mode. When set to Security Level 4, this reader will not transmit card data unless it is in the Authenticated Mode. The Authenticated Mode may only be entered by this command.

The application specifies a PreAuthentication Time Limit. This is the maximum number of seconds the reader will wait for the Activation Challenge Reply Command before timing out. If the supplied value is less than 120 seconds, the reader will use 120 seconds. If the reader times out waiting for the Activation Challenge Reply Command, the Authentication attempt fails and anti-hacking behavior may be invoked.

The reader responds with two challenges (Challenge 1 and Challenge 2) encrypted using a variant of the current DUKPT PIN Encryption Key (Key XOR F0F0 F0F0 F0F0 F0F0 F0F0 F0F0 F0F0). When decrypted, Challenge 1 contains 6 bytes of random number (used in the Activation Challenge Reply command) followed by the last two bytes of the KSN. These last two bytes of the KSN may be compared with the last two bytes of the clear text KSN sent in the message to authenticate the reader. The application should complete the Activate Authentication sequence using the Activation Challenge Reply command (see below).

The first two Activate Authenticated Mode commands may proceed without any delay (one error is allowed with no anti-hacking consequences). If a second Activate Authenticated Mode in a row fails, the reader goes into anti-hacking behavior. This consists of an increasing delay being enforced between Activate Authenticated Mode commands. The first delay is 10 seconds, increasing by 10 seconds until a maximum delay of 10 minutes is reached. The user may remove the reader from the anti-hacking mode at any time by swiping any encoded magstripe card. When the reader is in this anti-hacking mode it is **NOT** receptive to the Reset Device command.

Data structure:

Request Data:

Offset	Field Name	Description
0	PreAuthentication Time Limit (msb)	Most significant byte of the PreAuthentication Time Limit
1	PreAuthentication Time Limit (lsb)	Least significant byte of the PreAuthentication Time Limit.

Response Data:

Offset	Field Name	Description
0	Current Key Serial Number	This eighty-bit field includes the Initial Key Serial Number in the leftmost 59 bits and a value for the Encryption Counter in the rightmost 21 bits.
10	Challenge 1	This eight byte challenge may be used later in an Activation Challenge Reply command shown below, and to authenticate the reader as mentioned above.
18	Challenge 2	This eight byte challenge may be used later in a Deactivate Authenticated Mode command shown below.

Result codes:

- 0x00 (Success)
- 0x03 (Redundant – the reader is already in this mode)
- 0x05 (Delayed – the request is refused due to anti-hacking mode)
- 0x07 (Sequence Error – the current Security Level is too low)
- 0x80 (Encryption Counter Expired)

Example **Activate Authenticated Mode Request (Hex):**

Cmd Num	Data Len	Data
10	00	

Example **Activate Authenticated Mode Response (Hex):**

Result Code	Data Len	Data
00	20	FFFF 0123 4567 8000 0003 9845 A48B 7ED3 C294 7987 5FD4 03FA 8543

### ACTIVATION CHALLENGE REPLY COMMAND

Command number: 0x11

Description: This command is used as the second part of an Activate Authentication sequence. In this command, the application sends the first 6 bytes of Challenge 1 (received in response to the Activate Authenticated Mode command), two bytes of time information, and (optionally) an eight byte Session ID encrypted with a variant of the current DUKPT PIN Encryption Key (Key XOR 3C3C 3C3C 3C3C 3C3C 3C3C 3C3C 3C3C 3C3C).

The time information contains a count of the maximum number of seconds the reader should remain in the Authenticated Mode. Regardless of the value of this timer, a user card swipe in the Authenticated Mode ends the Authenticated Mode. The maximum time allowed is 3600 seconds (one hour). To get the full hour, use the value 0x0E10. To get the value of 3 minutes, use the value 0x012C. A value of zero forces the reader to stay in the Authenticated Mode until a card swipe or power down occurs (no timeout).

If the Session ID information is included and the command is successful, it will change the Session ID in the reader.

If the reader decrypts the CR response correctly the Activate Authenticated Mode has succeeded. If the reader can not decrypt the CR command correctly the Activate Authenticated Mode has failed, the DUKPT KSN advances.

Data structure:

Request Data:

Offset	Field Name	Description
0	Response to Challenge 1	Six bytes of Challenge 1 plus two bytes of time as outlined above, encrypted by the specified variant of the current DUKPT Key
8	Session ID	Optional eight byte Session ID encrypted by the specified variant of the current DUKPT Key.

Response Data: None

Result codes:

- 0x00 (Success)
- 0x02 (Bad Parameters – the Request Data is not a correct length)
- 0x04 (Bad Data – the encrypted reply data could not be verified)
- 0x07 (Sequence – not expecting this command)

Example **Activation Challenge Reply Request** (Hex):

Cmd Num	Data Len	Data
11	08	8579827521573495

Example **Activation Challenge Reply Response** (Hex):

Result Code	Data Len	Data
00	00	

## DEACTIVATE AUTHENTICATED MODE COMMAND

Command number: 0x12

Description: This command is used to exit the Authenticated Mode command. It can be used to exit the mode with or without incrementing the DUKPT transaction counter (lower 21 bits of the KSN). The application must send the first 7 bytes of Challenge 2 (from the response to the Activate Authenticated Mode command) and the Increment flag (0x00 indicates no increment, 0x01 indicates increment of the KSN) encrypted with a variant of the current DUKPT PIN Encryption Key (Key XOR 3C3C 3C3C 3C3C 3C3C 3C3C 3C3C 3C3C 3C3C 3C3C 3C3C).

If the reader decrypts Challenge 2 successfully it will exit the Authenticated Mode and, depending on the Increment flag, may increment the KSN.

If the reader cannot decrypt Challenge 2 successfully, it will stay in the Authenticated Mode until either the time specified in the Activate Authenticated Mode command passes or the user swipes a card. This behavior is intended to discourage denial of service attacks. Exiting the Authenticated Mode by timeout or card swipe *always* increments the KSN, exiting Authenticated Mode by the Deactivate Authenticated Mode command *may* increment the KSN.

Data structure:

Request Data:

Offset	Field Name	Description
0	Response to Challenge 2	Seven bytes of Challenge 2 plus one byte of Increment flag as outlined above, encrypted by the specified variant of the current DUKPT Key

Response Data: None

Result codes:      0x00 (Success)  
                       0x02 (Bad Parameters – the Request Data is not a correct length)  
                       0x03 (Bad Data – the encrypted reply data could not be verified)  
                       0x07 (Sequence – not expecting this command)

Example **Deactivate Authenticated Mode Request** (Hex):

Cmd Num	Data Len	Data
12	08	8579827521573495

Example **Deactivate Authenticated Mode Response** (Hex):

Result Code	Data Len	Data
00	00	

## GET READER STATE COMMAND

Command Number: 0x14

Description: This command is used to get the current state of the reader. The state is returned as two bytes that represent the Current State of the reader and how it got to that state (Antecedent). For more information see [Reader States](#).

Data Structure:

Request Data: None

Response Data:

The first byte specifies the current state as follows:

Current Reader State		
Value	Name	Meaning
0x00	WaitActAuth	Waiting for Activate Authenticated Mode. The reader requires Authentication before swipes are accepted.
0x01	WaitActRply	Waiting for Activation Challenge Reply. Activation has been started, the reader is waiting for the Activation Challenge Reply command.
0x02	WaitSwipe	Waiting for Swipe. The reader is waiting for the user to Swipe a card.
0x03	WaitDelay	Waiting for Anti-Hacking Timer. Two or more previous attempts to Authenticate failed, the reader is waiting for the Anti-Hacking timer to expire before it accepts further Activate Authenticated Mode commands.

The second byte specifies how the reader goes to its current state as follows:

Current State Antecedent		
Value	Name	Meaning
0x00	PU	Just Powered Up. The reader has had no swipes and has not been Authenticated since it was powered up.
0x01	GoodAuth	Authentication Activation Successful. The reader processed a valid Activation Challenge Reply command.
0x02	GoodSwipe	Good Swipe. The user swiped a valid card correctly.
0x03	BadSwipe	Bad Swipe. The user swiped a card incorrectly or the card is not valid.
0x04	FailAuth	Authentication Activation Failed. The most recent Activation Challenge Reply command failed.
0x05	FailDeact	Authentication Deactivation Failed. A recent Deactivate Authenticated Mode command failed.
0x06	TOAuth	Authentication Activation Timed Out. The Host failed to send an Activation Challenge Reply command in the time period specified in the Activate Authentication Mode command.
0x07	TOSwipe	Swipe Timed Out. The user failed to swipe a card in the time period specified in the Activation Challenge Reply command.
0x08	KeySyncError	The keys between the MagneSafe processor and the Encrypting IntelliHead are not the same and must be re-loaded before correct operation can resume.

Result codes:           0x00 (Success)

Example **Get Reader State Request** (Hex):

Cmd Num	Data Len	Data
1A	00	

Example **Get Reader State Response** (Hex):

Result Code	Data Len	Data
00	02	00 00

## SET SECURITY LEVEL COMMAND

Command number:   0x15

Description:       This command is used to set the Security Level (see Section 4). The Security Level can be set higher, but never lower. There are two versions of this command, the first one is used to retrieve the current Security Level and does not require MACing. The second one is used to set the Security Level and requires Security/MACing.

Data structure:

Request Data:

Offset	Field Name	Description
0	Security Level	Optional, if present must be either 0x03 or 0x04. If absent this is a query for the current Security Level. If this field is absent, the MAC field should NOT be sent.
1	MAC	Four byte MAC (See Section 4) to secure the command.

## Response Data:

Offset	Field Name	Description
0	Security Level	Only present if there was no request data. This field gives the current Security Level.

Result codes:

- 0x00 (Success)
- 0x02 (Bad Parameters – the Request Data is not a correct length OR the specified Security Level is invalid; OR the current Security Level is either 0, 1, or 4)
- 0x07 (Incorrect MAC – command not authorized)

## Example Set Security Level to 3 Request (Hex):

Cmd Num	Data Len	Data
15	05	03 xx xx xx xx

Where “xx xx xx xx” is the valid MAC (Message Authentication Code).

## Example Set Security Level Response (Hex):

Result Code	Data Len	Data
00	00	

## Example Get Security Level Request (Hex):

Cmd Num	Data Len	Data
15	00	

## Example Get Security Level Response (Hex):

Result Code	Data Len	Data
00	01	03

**GET ENCRYPTION COUNTER COMMAND**

Command number: 0x1C

Description: This command is used to Get the Encryption Counter. The Encryption Counter gives the maximum number of transactions that can be performed by the reader. A transaction is either an encrypted card swipe or a correctly completed Activation Sequence (Activate Authenticated Mode followed by correct Activation Challenge Reply)

The Encryption Counter has three possible states:

1. Disabled – value 0xFFFFFFFF – In this state there is no limit to the number of transactions that can be performed.
2. Expired – value 0x000000 – This state indicates that all transactions are prohibited
3. Active – value 1 to 1,000,000 (0x000001 to 0x0F4240) – In this state, each transaction causes the Encryption Counter to be decremented and allows transactions to be processed. If an Activation Sequence decrements the Encryption Counter to 0, a last encrypted card swipe will be permitted.

Request Data: None

Response Data:

Offset	Field Name	Description
0	Device Serial #	16 bytes, if DSN is shorter than 15 bytes, left justify and fill with binary zeroes. At least one byte (usually the last one) must contain binary zero.
16	Actual Encryption Counter	This three byte field returns the current value of the Encryption Counter.

Result codes: 0x00 (Success)  
 0x02 (Invalid length)

Example **Get Encryption Counter** Request (Hex):

Cmd Num	Data Len	Data
1C	00	

Example **Get Encryption Counter** Response (Hex) - Encryption Counter is 2033:

Result Code	Data Len	Data
00	13	54455354205345545550203030303100 0007F1

## POWER DOWN COMMAND (WIRELESS READER ONLY)

Command number: 0x28

Description: This command is used to power down the magnetic stripe circuit. If the reader is running on battery only (no USB cable attached), the entire reader is powered down. The behavior of the reader is exactly the same as if the user had pressed and held down the User Switch for three seconds to turn it off.

Data structure:

Request Data: None  
 Response Data: None

Result codes: 0x00 (Success)

Example **Power Down** Request (Hex):

Cmd Num	Data Len	Data
28	00	

Example **Power Down** Response (Hex):

Result Code	Data Len	Data
00	00	

## GET BATTERY STATUS COMMAND (WIRELESS READER ONLY)

Command number: 0x29

Description: This command is used to get the status of the battery.



Data structure:

Request Data: None

Response Data:

Offset	Field Name	Description
0	Battery Status	Value of 0x00 indicates battery charge is low; battery should be charged before further use. Value of 0x01 indicates battery charge is sufficient for normal use.

Result codes:           0x00 (success)

Example **Get Battery Status** Request (Hex):

Cmd Num	Data Len	Data
29	00	

Example **Get Battery Status** Response (Hex) (Charge is low):

Result Code	Data Len	Data
00	01	00



## SECTION 3. DEMO PROGRAM

The demo program, which is written in Visual Basic, can be used to do the following:

- Send command requests to the reader and view the command responses.
- Guide application developers in their application development by providing examples, in source code, of how to properly communicate with the reader using the standard Windows APIs.
- Read cards from the reader and view the card data (HID mode only).

The part numbers for the demo program can be found in this document in Section 1 under Accessories.

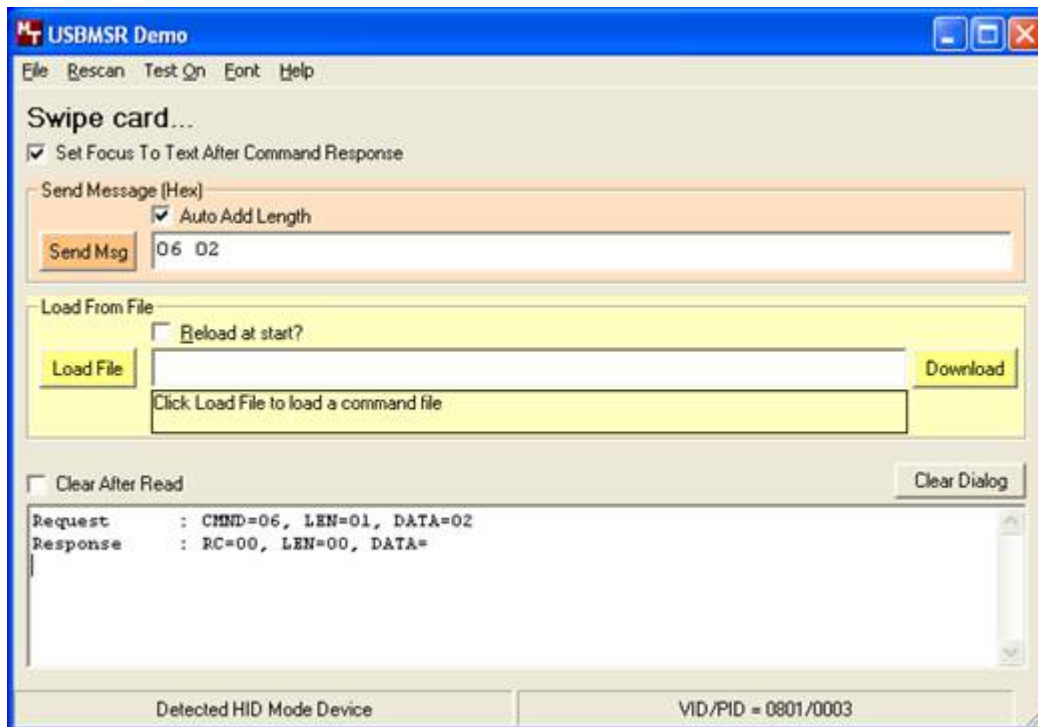
### INSTALLATION

To install the demo program, run the setup.exe file and follow the instructions given on the screen.

### OPERATION

To operate the demo program perform the following steps:

- Attach the reader into a USB port on the host.
- If this is the first time the reader has been plugged into the host, follow the instructions on the screen for installing the Windows HID device driver. This is explained in more detail in the installation section of this document.
- Run the demo program.



- To send commands to the reader, click the *Send Commands* tab (if not already selected).
- Enter a command in the Message edit box. All data entered should be in hexadecimal bytes with a space between each byte. Enter the command number followed by the command data if there is any. **The application will automatically calculate and send the command data length for you** if the *Auto Add Length* box is checked. For example, to send the **Get Property** command for property **Software ID** enter 00 00.
- Press Enter or click *Send Msg* to send the command and receive the result.
- The command request and the command result will be displayed in the Communications Dialog edit box.
- The *Clear Dialog* button clears the Communication Dialog edit box.
- To read cards and view the card data when in the HID mode, click the *Read Cards* tab.
- To read cards and view the card data when in the Keyboard Emulation mode, do not use the demo program. Use a text editor program such as Windows Notepad.

### SOURCE CODE

Source code is included with the demo program. It can be used as a guide for application development. It is described in detail, with comments, to assist developers. The book *USB Complete* by Jan Axelson is also a good guide for application developers, especially the chapter on Human Interface Device Host Applications (see “Reference Documents” in Section 1).

## APPENDIX A. KEYBOARD USAGE ID DEFINITIONS

This appendix is from the following document found on [www.usb.org](http://www.usb.org): Universal Serial Bus HID Usage Tables, Version 1.12 and specifically for this manual, Section 10, Keyboard/Keypad Page (0x07).

### KEYBOARD/KEYPAD PAGE (0X07)

This section is the Usage Page for key codes to be used in implementing a USB keyboard. A Boot Keyboard (84-, 101- or 104-key) should at a minimum support all associated usage codes as indicated in the “Boot” column below.

The usage type of all key codes is Selectors (Sel), except for the modifier keys Keyboard Left Control (0x224) to Keyboard Right GUI (0x231) which are Dynamic Flags (DV).

#### Note

*A general note on Usages and languages: Due to the variation of keyboards from language to language, it is not feasible to specify exact key mappings for every language. Where this list is not specific for a key function in a language, the closest equivalent key position should be used, so that a keyboard may be modified for a different language by simply printing different keycaps. One example is the Y key on a North American keyboard. In Germany this is typically Z. Rather than changing the keyboard firmware to put the Z Usage into that place in the descriptor list, the vendor should use the Y Usage on both the North American and German keyboards. This continues to be the existing practice in the industry, in order to minimize the number of changes to the electronics to accommodate other languages.*

**Table A-1. Keyboard/Keypad**

Usage ID (Dec)	Usage ID (Hex)	Usage Name	Ref: Typical AT-101 Position	PC-AT	Mac	UNIX	Boot
0	00	Reserved (no event indicated) <sup>9</sup>	N/A	√	√	√	4/101/104
1	01	Keyboard ErrorRollOver <sup>9</sup>	N/A	√	√	√	4/101/104
2	02	Keyboard POSTFail <sup>9</sup>	N/A	√	√	√	4/101/104
3	03	Keyboard ErrorUndefined <sup>9</sup>	N/A	√	√	√	4/101/104
4	04	Keyboard a and A <sup>4</sup>	31	√	√	√	4/101/104
5	05	Keyboard b and B	50	√	√	√	4/101/104
6	06	Keyboard c and C <sup>4</sup>	48	√	√	√	4/101/104
7	07	Keyboard d and D	33	√	√	√	4/101/104
8	08	Keyboard e and E	19	√	√	√	4/101/104
9	09	Keyboard f and F	34	√	√	√	4/101/104
10	0A	Keyboard g and G	35	√	√	√	4/101/104
11	0B	Keyboard h and H	36	√	√	√	4/101/104

Usage ID (Dec)	Usage ID (Hex)	Usage Name	Ref: Typical AT-101 Position	PC-AT	Mac	UNIX	Boot
12	0C	Keyboard i and I	24	√	√	√	4/101/104
13	0D	Keyboard j and J	37	√	√	√	4/101/104
14	0E	Keyboard k and K	38	√	√	√	4/101/104
15	0F	Keyboard l and L	39	√	√	√	4/101/104
16	10	Keyboard m and M	52	√	√	√	4/101/104
17	11	Keyboard n and N	51	√	√	√	4/101/104
18	12	Keyboard o and O <sup>4</sup>	25	√	√	√	4/101/104
19	13	Keyboard p and P <sup>4</sup>	26	√	√	√	4/101/104
20	14	Keyboard q and Q <sup>4</sup>	27	√	√	√	4/101/104
21	15	Keyboard r and R	20	√	√	√	4/101/104
22	16	Keyboard s and S <sup>4</sup>	32	√	√	√	4/101/104
23	17	Keyboard t and T	21	√	√	√	4/101/104
24	18	Keyboard u and U	23	√	√	√	4/101/104
25	19	Keyboard v and V	49	√	√	√	4/101/104
26	1A	Keyboard w and W <sup>4</sup>	18	√	√	√	4/101/104
27	1B	Keyboard x and X <sup>4</sup>	47	√	√	√	4/101/104
28	1C	Keyboard y and Y <sup>4</sup>	22	√	√	√	4/101/104
29	1D	Keyboard z and Z <sup>4</sup>	46	√	√	√	4/101/104
30	1E	Keyboard 1 and ! <sup>4</sup>	2	√	√	√	4/101/104
31	1F	Keyboard 2 and ! <sup>4</sup>	3	√	√	√	4/101/104
32	20	Keyboard 3 and # <sup>4</sup>	4	√	√	√	4/101/104
33	21	Keyboard 4 and \$ <sup>4</sup>	5	√	√	√	4/101/104
34	22	Keyboard 5 and % <sup>4</sup>	6	√	√	√	4/101/104
35	23	Keyboard 6 and ^ <sup>4</sup>	7	√	√	√	4/101/104
36	24	Keyboard 7 and & <sup>4</sup>	8	√	√	√	4/101/104
37	25	Keyboard 8 and * <sup>4</sup>	9	√	√	√	4/101/104
38	26	Keyboard 9 and ( <sup>4</sup>	10	√	√	√	4/101/104
39	27	Keyboard 0 and ) <sup>4</sup>	11	√	√	√	4/101/104
40	28	Keyboard Return (ENTER) <sup>5</sup>	43	√	√	√	4/101/104
41	29	Keyboard ESCAPE	110	√	√	√	4/101/104
42	2A	Keyboard DELETE (Backspace)	15	√	√	√	4/101/104
43	2B	Keyboard Tab	16	√	√	√	4/101/104
44	2C	Keyboard Spacebar	61	√	√	√	4/101/104
45	2D	Keyboard - and (underscore) <sup>4</sup>	12	√	√	√	4/101/104
46	2E	Keyboard = and + <sup>4</sup>	13	√	√	√	4/101/104
47	2F	Keyboard [ and { <sup>4</sup>	27	√	√	√	4/101/104
48	30	Keyboard ] and } <sup>4</sup>	28	√	√	√	4/101/104
49	31	Keyboard \ and	29	√	√	√	4/101/104
50	32	Keyboard Non-US # and ~ <sup>2</sup>	42	√	√	√	4/101/104
51	33	Keyboard ; and : <sup>4</sup>	40	√	√	√	4/101/104

Usage ID (Dec)	Usage ID (Hex)	Usage Name	Ref: Typical AT-101 Position	PC-AT	Mac	UNIX	Boot
52	34	Keyboard ' and <sup>4</sup>	41	√	√	√	4/101/104
53	35	Keyboard Grave Accent and Tilde <sup>4</sup>	1	√	√	√	4/101/104
54	36	Keyboard, and < <sup>4</sup>	53	√	√	√	4/101/104
55	37	Keyboard. and > <sup>4</sup>	54	√	√	√	4/101/104
56	38	Keyboard / and ?	55	√	√	√	4/101/104
57	39	Keyboard Caps Lock <sup>11</sup>	30	√	√	√	4/101/104
58	3A	Keyboard F1	112	√	√	√	4/101/104
59	3B	Keyboard F2	113	√	√	√	4/101/104
60	3C	Keyboard F3	114	√	√	√	4/101/104
61	3D	Keyboard F4	115	√	√	√	4/101/104
62	3E	Keyboard F5	116	√	√	√	4/101/104
63	3F	Keyboard F6	117	√	√	√	4/101/104
64	40	Keyboard F7	118	√	√	√	4/101/104
65	41	Keyboard F8	119	√	√	√	4/101/104
66	42	Keyboard F9	120	√	√	√	4/101/104
67	43	Keyboard F10	121	√	√	√	4/101/104
68	44	Keyboard F11	122	√	√	√	101/104
69	45	Keyboard F12	123	√	√	√	101/104
70	46	Keyboard PrintScreen <sup>1</sup>	124	√	√	√	101/104
71	47	Keyboard Scroll Lock <sup>11</sup>	125	√	√	√	4/101/104
72	48	Keyboard Pause <sup>1</sup>	126	√	√	√	101/104
73	49	Keyboard Insert <sup>1</sup>	75	√	√	√	101/104
74	4A	Keyboard Home <sup>1</sup>	80	√	√	√	101/104
75	4B	Keyboard PageUp <sup>1</sup>	85	√	√	√	101/104
76	4C	Keyboard Delete Forward <sup>1;14</sup>	76	√	√	√	101/104
77	4D	Keyboard End <sup>1</sup>	81	√	√	√	101/104
78	4E	Keyboard PageDown <sup>1</sup>	86	√	√	√	101/104
79	4F	Keyboard RightArrow <sup>1</sup>	89	√	√	√	101/104
80	50	Keyboard LeftArrow <sup>1</sup>	79	√	√	√	101/104
81	51	Keyboard DownArrow <sup>1</sup>	84	√	√	√	101/104
82	52	Keyboard UpArrow <sup>1</sup>	83	√	√	√	101/104
83	53	Keypad Num Lock and Clear <sup>1</sup>	90	√	√	√	101/104
84	54	Keypad / <sup>1</sup>	95	√	√	√	101/104
85	55	Keypad *	100	√	√	√	4/101/104
86	56	Keypad -	105	√	√	√	4/101/104
87	57	Keypad +	106	√	√	√	4/101/104
88	58	Keypad ENTER5	108	√	√	√	101/104
89	59	Keypad 1 and End	93	√	√	√	4/101/104
90	5A	Keypad 2 and Down Arrow	98	√	√	√	4/101/104
91	5B	Keypad 3 and PageDn	103	√	√	√	4/101/104

Usage ID (Dec)	Usage ID (Hex)	Usage Name	Ref: Typical AT-101 Position	PC-AT	Mac	UNIX	Boot
92	5C	Keypad 4 and Left Arrow	92	√	√	√	4/101/104
93	5D	Keypad 4 and Left Arrow	97	√	√	√	4/101/104
94	5E	Keypad 4 and Left Arrow	102	√	√	√	4/101/104
95	5F	Keypad 7 and Home	91	√	√	√	4/101/104
96	60	Keypad 8 and Up Arrow	96	√	√	√	4/101/104
97	61	Keypad 9 and PageUp	101	√	√	√	4/101/104
98	62	Keypad 0 and Insert	99	√	√	√	4/101/104
99	63	Keypad . and Delete	104	√	√	√	4/101/104
100	64	Keyboard Non-US \ and   <sup>3,6</sup>	45	√	√	√	4/101/104
101	65	Keyboard Application <sup>10</sup>	129	√		√	104
102	66	Keyboard Power <sup>9</sup> =			√	√	
103	67	Keypad =			√		
104	68	Keyboard F13	62		√		
105	69	Keyboard F14	63		√		
106	6A	Keyboard F15	64		√		
107	6B	Keyboard F16	65				
107	6C	Keyboard F17					
109	6D	Keyboard F18					
110	6E	Keyboard F19					
111	6F	Keyboard F20					
112	70	Keyboard F21					
113	71	Keyboard F22					
114	72	Keyboard F23					
115	73	Keyboard F24					
116	74	Keyboard Execute				√	
117	75	Keyboard Help				√	
118	76	Keyboard Menu				√	
119	77	Keyboard Select				√	
120	78	Keyboard Stop				√	
121	79	Keyboard Again				√	
122	7A	Keyboard Undo				√	
123	7B	Keyboard Cut				√	
124	7C	Keyboard Copy				√	
125	7D	Keyboard Paste				√	
126	7E	Keyboard Find				√	
127	7F	Keyboard Mute				√	
128	80	Keyboard Volume Up				√	
129	81	Keyboard Volume Down				√	
130	82	Keyboard Locking Caps Lock <sup>12</sup>				√	
131	83	Keyboard Locking Num Lock <sup>12</sup>				√	



Usage ID (Dec)	Usage ID (Hex)	Usage Name	Ref: Typical AT-101 Position	PC-AT	Mac	UNIX	Boot
132	84	Keyboard Locking Scroll Lock <sup>12</sup>				√	
133	85	Keypad Comma <sup>27</sup>	107				
134	86	Keypad Equal Sign <sup>29</sup>					
135	87	Keyboard International1 <sup>15-28</sup>	56				
136	88	Keyboard International2 <sup>16</sup>					
137	89	Keyboard International3 <sup>17</sup>					
138	8A	Keyboard International4 <sup>18</sup>					
139	8B	Keyboard International5 <sup>19</sup>					
140	8C	Keyboard International6 <sup>20</sup>					
141	8D	Keyboard International7 <sup>21</sup>					
142	8E	Keyboard International8 <sup>22</sup>					
143	8F	Keyboard International9 <sup>22</sup>					
144	90	Keyboard Lang1 <sup>25</sup>					
145	91	Keyboard Lang2 <sup>26</sup>					
146	92	Keyboard Lang3 <sup>30</sup>					
147	93	Keyboard Lang4 <sup>31</sup>					
148	94	Keyboard Lang5 <sup>32</sup>					
149	95	Keyboard Lang6 <sup>8</sup>					
150	96	Keyboard Lang7 <sup>8</sup>					
151	97	Keyboard Lang8 <sup>8</sup>					
152	98	Keyboard Lang9 <sup>8</sup>					
153	99	Keyboard Alternate Erase <sup>7</sup>					
154	9A	Keyboard Sys/Req Attention <sup>1</sup>					
155	9B	Keyboard Cancel					
156	9C	Keyboard Clear					
157	9D	Keyboard Prior					
158	9E	Keyboard Return					
159	9F	Keyboard Separator					
160	A0	Keyboard Out					
161	A1	Keyboard Oper					
162	A2	Keyboard Clear/Again					
163	A3	Keyboard Cr/Sel/Props					
164	A4	Keyboard Ex Sel					
165-175	A5-CF	Reserved					
176	B0	Keypad 00					
177	B1	Keypad 000					
178	B2	Thousands Separator <sup>33</sup>					
179	B3	Decimal Separator <sup>33</sup>					
180	B4	Currency Unit <sup>34</sup>					
181	B5	Currency Sub-unit <sup>34</sup>					

Usage ID (Dec)	Usage ID (Hex)	Usage Name	Ref: Typical AT-101 Position	PC-AT	Mac	UNIX	Boot
182	B6	Keypad (					
183	B7	Keypad )					
184	B8	Keypad {					
185	B9	Keypad}					
186	BA	Keypad Tab					
187	BB	Keypad Backspace					
188	BC	Keypad A					
189	BD	Keypad B					
190	BE	Keypad C					
191	BF	Keypad D					
192	C0	Keypad E					
193	C1	Keypad F					
194	C2	Keypad XOR					
195	C3	Keypad ^					
196	C4	Keypad %					
197	C5	Keypad <					
198	C6	Keypad >					
199	C7	Keypad &					
200	C8	Keypad &&					
201	C9	Keypad					
202	CA	Keypad					
203	CB	Keypad :					
204	CC	Keypad #					
205	CD	Keypad Space					
206	CE	Keypad @					
207	CF	Keypad !					
208	D0	Keypad Memory Store					
209	D1	Keypad Memory Recall					
210	D2	Keypad Memory Clear					
211	D3	Keypad Memory Add					
212	D4	Keypad Memory Subtract					
213	D5	Keypad Memory Multiple					
214	D6	Keypad Memory Divide					
215	D7	Keypad +/-					
216	D8	Keypad Clear					
217	D9	Keypad Clear Entry					
218	DA	Keypad Binary					
219	DB	Keypad Octal					
220	DC	Keypad Decimal					
221	DD	Keypad Hexadecimal					

Usage ID (Dec)	Usage ID (Hex)	Usage Name	Ref: Typical AT-101 Position	PC-AT	Mac	UNIX	Boot
222-223	DE-DF	Reserved					
224	E0	Keyboard LeftControl	58	√	√	√	
225	E1	Keyboard LeftShift	44	√	√	√	
226	E2	Keyboard LeftA;t	60	√	√	√	
227	E3	Keyboard Left GUI <sup>10;23</sup>	127	√	√	√	
228	E4	Keyboard RightControl	64	√	√	√	
229	E5	Keyboard RightShift	57	√	√	√	
230	E6	Keyboard RightAlt	62	√	√	√	
231	E7	Keyboard Right GUI <sup>10;24</sup>	128	√	√	√	
232 – 65535	E8-FFFF	Reserved					

**Footnotes**

1. Usage of keys is not modified by the state of the Control, Alt, Shift or Num Lock keys. That is, a key does not send extra codes to compensate for the state of any Control, Alt, Shift or Num Lock keys.
2. Typical language mappings: US: \| Belg: µ`£ FrCa: <|> Dan: \* Dutch: <|> Fren: \*µ Ger: #’ Ital: ù\$ LatAm: }`] Nor:,\* Span: }Ç Swed: , \* Swiss: \$£ UK: #~.
3. Typical language mappings: Belg:<|> FrCa:«°» Dan:<|> Dutch:|] Fren:<|> Ger:<|> Ital:<|> LatAm:<|> Nor:<|> Span:<|> Swed:<|> Swiss:<|> UK:<|> Brazil: \|.
4. Typically remapped for other languages in the host system.
5. Keyboard Enter and Keypad Enter generate different Usage codes.
6. Typically near the Left-Shift key in AT-102 implementations.
7. Example, Erase-Eaze™ key.
8. Reserved for language-specific functions, such as Front End Processors and Input Method Editors.
9. Reserved for typical keyboard status or keyboard errors. Sent as a member of the keyboard array. Not a physical key.
10. Windows key for Windows 95, and “Compose.”
11. Implemented as a non-locking key; sent as member of an array.
12. Implemented as a locking key; sent as a toggle button. Available for legacy support; however, most systems should use the non-locking version of this key.
13. Backs up the cursor one position, deleting a character as it goes.
14. Deletes one character without changing position.
- 15-20. See additional foot notes in Universal Serial Bus HID Usage Tables, Copyright © 1996-2005, USB Implementers Forum.
21. Toggle Double-Byte/Single-Byte mode.
22. Undefined, available for other Front End Language Processors.
23. Windowing environment key, examples are Microsoft Left Win key, Mac Left Apple key, Sun Left Meta key
24. Windowing environment key, examples are Microsoft® RIGHT WIN key, Macintosh® RIGHT APPLE key, Sun® RIGHT META key.
25. Hangul/English toggle key. This usage is used as an input method editor control key on a Korean language keyboard.
26. Hanja conversion key. This usage is used as an input method editor control key on a Korean language keyboard.
27. Keypad Comma is the appropriate usage for the Brazilian keypad period (.) key. This represents the closest possible match, and system software should do the correct mapping based on the current locale setting.
28. Keyboard International1 should be identified via footnote as the appropriate usage for the Brazilian forward-slash (/) and question-mark (?) key. This usage should also be renamed to either “Keyboard Non-US / and ?” or to “Keyboard International1” now that it's become clear that it does not only apply to Kanji keyboards anymore.

29. Used on AS/400 keyboards.
30. Defines the Katakana key for Japanese USB word-processing keyboards.
31. Defines the Hiragana key for Japanese USB word-processing keyboards.
32. Usage 0x94 (Keyboard LANG5) “Defines the Zenkaku/Hankaku key for Japanese USB word-processing keyboards.
33. The symbol displayed will depend on the current locale settings of the operating system. For example, the US thousands separator would be a comma, and the decimal separator would be a period.
34. The symbol displayed will depend on the current locale settings of the operating system. For example the US currency unit would be \$ and the sub-unit would be ¢.

## APPENDIX B. MODIFIER BYTE DEFINITIONS

This appendix is from the following document found on [www.usb.org](http://www.usb.org): Device Class Definition for Human Interface Devices (HID) Version 1.11, and specifically for this manual, Section 8.3 Report Format for Array Items.

The modifier byte is defined as follows:

**Table B-1. Modifier Byte**

Bit	Key
0	LEFT CTRL
1	LEFT SHIFT
2	LEFT ALT
3	LEFT GUI
4	RIGHT CTRL
5	RIGHT SHIFT
6	RIGHT ALT
7	RIGHT GUI



## APPENDIX C. GUIDE ON DECRYPTING DATA

The key that was used to encrypt each data block can be determined by using the Key Serial Number field along with the Base Derivation Key associated with this reader. The resulting DUKPT key, as described in ANS X9.24 Part 1, is the key which was used to encrypt the data. (The key is described as the PIN key in the standard but since there are no PINs being used in this application, the derived key is used.)

These sequences are based on the following data:

- Derivation Key: 0123 4567 89AB CDEF FEDC BA98 7654 3210
- Initially Loaded Key Serial Number (KSN): FFFF 9876 5432 10E0 0000
- Initially Loaded PIN Entry Device Key: 6AC2 92FA A131 5B4D 858A B3A3 D7D5 933A

When a data field consists of more than one block, Cipher Block Chaining (CBC) method is used by the encrypting algorithm.

To decrypt this group of data, follow these steps:

- Start decryption on the last block.
- The result of the decryption is then XORed with the previous block.
- Continue until reaching the first block.
- The first block can skip the XOR operation.





## APPENDIX D. COMMAND EXAMPLES

This Appendix gives examples of command sequences and cryptographic operations. The intent is to clarify any ambiguities the user might find in the body of the document. Each example shows a sequence as it actually runs, thus the user can check algorithms against the examples to assure they are computing correctly.

### Example 1: Configuring a reader before encryption is enabled (Security Level 2). In this example the reader is set up for HID operation:

```
; This script demonstrates configuration commands for HID mode.
; It assumes the reader is at Security Level 2 and that the Device
; Serial Number has never been set.
00 01 10      ; Get current interface
Request       : CMND=00, LEN=01, DATA=10
Response      : RC= 00, LEN=01, DATA=01

01 02 10 00   ; Set Interface to HID
Request       : CMND=01, LEN=02, DATA=10 00
Response      : RC= 00, LEN=00, DATA=

02 00         ; Reset so changes take effect
Request       : CMND=02, LEN=00, DATA=
Response      : RC= 00, LEN=00, DATA=

Delay         : (waited 5 seconds)

00 01 10      ; Get current interface (should return 0)
Request       : CMND=00, LEN=01, DATA=10
Response      : RC= 00, LEN=01, DATA=00

00 01 01      ; Get current USB SN
Request       : CMND=00, LEN=01, DATA=01
Response      : RC= 00, LEN=00, DATA=

01 05 01 31323334 ; Set USB SN to "1234"
Request       : CMND=01, LEN=05, DATA=01 31 32 33 34
Response      : RC= 00, LEN=00, DATA=

00 01 02      ; Get current Polling Interval
Request       : CMND=00, LEN=01, DATA=02
Response      : RC= 00, LEN=01, DATA=01

01 02 02 02   ; Set Polling Interval to 2 ms
Request       : CMND=01, LEN=02, DATA=02 02
Response      : RC= 00, LEN=00, DATA=

00 01 03      ; Get current Device Serial Number
Request       : CMND=00, LEN=01, DATA=03
Response      : RC= 00, LEN=00, DATA=

01 08 03 42303030373935 ; Set DSN to "B000795"
Request       : CMND=01, LEN=08, DATA=03 42 30 30 30 37 39 35
Response      : RC= 00, LEN=00, DATA=
```

## USB MagneSafe V5

---

```
00 01 05      ; Get current Track ID Enable
Request       : CMND=00, LEN=01, DATA=05
Response      : RC= 00, LEN=01, DATA=95

01 02 05 85   ; Set to read only Tracks 1 & 2
Request       : CMND=01, LEN=02, DATA=05 85
Response      : RC= 00, LEN=00, DATA=

00 01 07      ; Get current ISO Track Mask
Request       : CMND=00, LEN=01, DATA=07
Response      : RC= 00, LEN=06, DATA=30 34 30 34 30 59

01 07 07 303430342A4E ; Set ISO Track Mask "0404*N" (uses * as mask char)
Request       : CMND=01, LEN=07, DATA=07 30 34 30 34 2A 4E
Response      : RC= 00, LEN=00, DATA=

00 01 0A      ; Get Max Packet Size
Request       : CMND=00, LEN=01, DATA=0A
Response      : RC= 00, LEN=01, DATA=08

01 02 0A 40   ; Set Max Packet Size to 64
Request       : CMND=01, LEN=02, DATA=0A 40
Response      : RC= 00, LEN=00, DATA=

02 00         ; Reset so changes take effect
Request       : CMND=02, LEN=00, DATA=
Response      : RC= 00, LEN=00, DATA=

Delay         : (waited 20 seconds)
00 01 10      ; Get current interface (should be 00)
Request       : CMND=00, LEN=01, DATA=10
Response      : RC= 00, LEN=01, DATA=00

00 01 01      ; Get current USB SN (should be "1234")
Request       : CMND=00, LEN=01, DATA=01
Response      : RC= 00, LEN=04, DATA=31 32 33 34

00 01 02      ; Get current Polling Interval (should be 02)
Request       : CMND=00, LEN=01, DATA=02
Response      : RC= 00, LEN=01, DATA=02

00 01 03      ; Get current Device Serial Number (should be "B000795")
Request       : CMND=00, LEN=01, DATA=03
Response      : RC= 00, LEN=07, DATA=42 30 30 30 37 39 35

00 01 05      ; Get current Track ID Enable (should be 85)
Request       : CMND=00, LEN=01, DATA=05
Response      : RC= 00, LEN=01, DATA=85

00 01 07      ; Get current ISO Track Mask (should be "0404*N")
Request       : CMND=00, LEN=01, DATA=07
Response      : RC= 00, LEN=06, DATA=30 34 30 34 2A 4E

00 01 0A      ; Get Max Packet Size (should be 40)
Request       : CMND=00, LEN=01, DATA=0A
Response      : RC= 00, LEN=01, DATA=40
```

```

;                               ; END OF SCRIPT

Finished downloading

Card Encode Type = ISO

DUKPT Key Serial Number = 00000000000000000000

Track 1 Encrypted = 25 42 35 34 35 32 33 30 30 35 35 31 32 32 37 31 38 39 5E
48 4F 47 41 4E 2F 50 41 55 4C 20 20 20 20 20 20 5E 30 38 30 34 33 32 31 30
30 30 30 30 30 30 37 32 35 30 30 30 30 30 30 3F 00 00 00 00

Track 2 Encrypted = 3B 35 34 35 32 33 30 30 35 35 31 32 32 37 31 38 39 3D 30
38 30 34 33 32 31 30 30 30 30 30 30 30 37 32 35 30 3F 00 00 00

Track 3 Encrypted =

MagnePrint Status (hex) = 000005A1

MagnePrint Data (hex) = 01 00 02 8C 97 A8 CA 5B 69 CF D8 66 AA 23 88 E3 E1
2B E3 79 04 D3 31 6E F5 3F 9C 30 0B E2 43 A0 4E 6C 68 09 87 B2 52 DC 89 04
A6 F0 2B A7 73 E7 03 AF EA AD C0 1F 00 00

Device Serial Number = B000795

Track 1 Masked = %B5452300551227189^HOGAN/PAUL
^08043210000000725000000?

Track 2 Masked = ;5452300551227189=080432100000007250?

Encrypted Session ID (Hex) = 00 00 00 00 00 00 00 00

```

**Example 2: Configuring a reader before encryption is enabled (Security Level 2). In this example the reader is set up for Keyboard Emulation:**

```

; This script demonstrates configuration commands for KB mode.
; It assumes the reader is at Security Level 2 and that the Device
; Serial Number has never been set.
00 01 10      ; Get current interface
Request       : CMND=00, LEN=01, DATA=10
Response      : RC= 00, LEN=01, DATA=01

01 02 10 01  ; Set Interface to KB
Request       : CMND=01, LEN=02, DATA=10 01
Response      : RC= 00, LEN=00, DATA=

02 00        ; Reset so changes take effect
Request       : CMND=02, LEN=00, DATA=
Response      : RC= 00, LEN=00, DATA=

Delay         : (waited 5 seconds)

00 01 10      ; Get current interface (should return 0)
Request       : CMND=00, LEN=01, DATA=10
Response      : RC= 00, LEN=01, DATA=01

```

## USB MagneSafe V5

---

```
00 01 01      ; Get current USB SN
Request       : CMND=00, LEN=01, DATA=01
Response      : RC= 00, LEN=00, DATA=

01 05 01 31323334 ; Set USB SN to "1234"
Request       : CMND=01, LEN=05, DATA=01 31 32 33 34
Response      : RC= 00, LEN=00, DATA=

00 01 02      ; Get current Polling Interval
Request       : CMND=00, LEN=01, DATA=02
Response      : RC= 00, LEN=01, DATA=01

01 02 02 02    ; Set Polling Interval to 2 ms
Request       : CMND=01, LEN=02, DATA=02 02
Response      : RC= 00, LEN=00, DATA=

00 01 03      ; Get current Device Serial Number
Request       : CMND=00, LEN=01, DATA=03
Response      : RC= 00, LEN=00, DATA=

01 08 03 42303030373935 ; Set DSN to "B000795"
Request       : CMND=01, LEN=08, DATA=03 42 30 30 30 37 39 35
Response      : RC= 00, LEN=00, DATA=

00 01 05      ; Get current Track ID Enable
Request       : CMND=00, LEN=01, DATA=05
Response      : RC= 00, LEN=01, DATA=95

01 02 05 85    ; Set to read only Tracks 1 & 2
Request       : CMND=01, LEN=02, DATA=05 85
Response      : RC= 00, LEN=00, DATA=

00 01 07      ; Get current ISO Track Mask
Request       : CMND=00, LEN=01, DATA=07
Response      : RC= 00, LEN=06, DATA=30 34 30 34 30 59

01 07 07 303430342A4E ; Set ISO Track Mask "0404*N" (uses * as mask char)
Request       : CMND=01, LEN=07, DATA=07 30 34 30 34 2A 4E
Response      : RC= 00, LEN=00, DATA=

00 01 19      ; Get current CRC Flags
Request       : CMND=00, LEN=01, DATA=19
Response      : RC= 00, LEN=01, DATA=01

01 02 19 03    ; Set to return Both
Request       : CMND=01, LEN=02, DATA=19 03
Response      : RC= 00, LEN=00, DATA=

00 01 1A      ; Get Current SureSwipe Flag
Request       : CMND=00, LEN=01, DATA=1A
Response      : RC= 00, LEN=01, DATA=01

;01 02 1A 00   ; Set to NO SureSwipe (enable if at Security Level 2)

00 01 1E      ; Get current Pre Card String
Request       : CMND=00, LEN=01, DATA=1E
```

```

Response      : RC= 00, LEN=00, DATA=

01 07 1E 435244545354 ; Set to "CRDTST"
Request       : CMND=01, LEN=07, DATA=1E 43 52 44 54 53 54
Response      : RC= 00, LEN=00, DATA=

00 01 1F          ; Get current Post Card String
Request       : CMND=00, LEN=01, DATA=1F
Response      : RC= 00, LEN=00, DATA=

01 07 1F 435244454E44 ; Set to "CRDEND"
Request       : CMND=01, LEN=07, DATA=1F 43 52 44 45 4E 44
Response      : RC= 00, LEN=00, DATA=

00 01 20          ; Get current Pre Track String
Request       : CMND=00, LEN=01, DATA=20
Response      : RC= 00, LEN=00, DATA=

01 07 20 54524B545354 ; Set to "TRKTST"
Request       : CMND=01, LEN=07, DATA=20 54 52 4B 54 53 54
Response      : RC= 00, LEN=00, DATA=

00 01 21          ; Get current Post Track String
Request       : CMND=00, LEN=01, DATA=21
Response      : RC= 00, LEN=00, DATA=

01 07 21 54524B454E44 ; Set to "TRKEND"
Request       : CMND=01, LEN=07, DATA=21 54 52 4B 45 4E 44
Response      : RC= 00, LEN=00, DATA=

00 01 22          ; Get current Termination String
Request       : CMND=00, LEN=01, DATA=22
Response      : RC= 00, LEN=01, DATA=0D

01 07 22 5458454E440D ; Set to "TXEND<ENTER>"
Request       : CMND=01, LEN=07, DATA=22 54 58 45 4E 44 0D
Response      : RC= 00, LEN=00, DATA=

00 01 2C          ; Get current Format Code
Request       : CMND=00, LEN=01, DATA=2C
Response      : RC= 00, LEN=05, DATA=31 FF FF FF FF

01 05 2C 31323334      ; Set to "1234"
Request       : CMND=01, LEN=05, DATA=2C 31 32 33 34
Response      : RC= 00, LEN=00, DATA=

02 00            ; Reset so changes take effect
Request       : CMND=02, LEN=00, DATA=
Response      : RC= 00, LEN=00, DATA=

Delay         : (waited 20 seconds)
00 01 10        ; Get current interface (should return 01)
Request       : CMND=00, LEN=01, DATA=10
Response      : RC= 00, LEN=01, DATA=01

00 01 01        ; Get current USB SN (should return "1234")
Request       : CMND=00, LEN=01, DATA=01

```

```
Response      : RC= 00, LEN=04, DATA=31 32 33 34

00 01 02      ; Get current Polling Interval (should return 02)
Request       : CMND=00, LEN=01, DATA=02
Response      : RC= 00, LEN=01, DATA=02

00 01 03      ; Get current Device Serial Number (should return "B000795")
Request       : CMND=00, LEN=01, DATA=03
Response      : RC= 00, LEN=07, DATA=42 30 30 30 37 39 35

00 01 05      ; Get current Track ID Enable (should return 85)
Request       : CMND=00, LEN=01, DATA=05
Response      : RC= 00, LEN=01, DATA=85

00 01 07      ; Get current ISO Track Mask (should return "0404*N")
Request       : CMND=00, LEN=01, DATA=07
Response      : RC= 00, LEN=06, DATA=30 34 30 34 2A 4E

00 01 19      ; Get current CRC Flags (should return 03)
Request       : CMND=00, LEN=01, DATA=19
Response      : RC= 00, LEN=01, DATA=03

00 01 1A      ; Get Current SureSwipe Flag (should return 00, if Set was
done)
Request       : CMND=00, LEN=01, DATA=1A
Response      : RC= 00, LEN=01, DATA=01

00 01 1E      ; Get current Pre Card String (should return "CRDTST")
Request       : CMND=00, LEN=01, DATA=1E
Response      : RC= 00, LEN=06, DATA=43 52 44 54 53 54

00 01 1F      ; Get current Post Card String (should return "CRDEND")
Request       : CMND=00, LEN=01, DATA=1F
Response      : RC= 00, LEN=06, DATA=43 52 44 45 4E 44

00 01 20      ; Get current Pre Track String (should return "TRKTST")
Request       : CMND=00, LEN=01, DATA=20
Response      : RC= 00, LEN=06, DATA=54 52 4B 54 53 54

00 01 21      ; Get current Post Track String (should return "TRKEND")
Request       : CMND=00, LEN=01, DATA=21
Response      : RC= 00, LEN=06, DATA=54 52 4B 45 4E 44

00 01 22      ; Get current Termination String (should return "TXEND<ENTER>")
Request       : CMND=00, LEN=01, DATA=22
Response      : RC= 00, LEN=06, DATA=54 58 45 4E 44 0D

00 01 2C      ; Get current Format Code (should return "1234")
Request       : CMND=00, LEN=01, DATA=2C
Response      : RC= 00, LEN=04, DATA=31 32 33 34

;             ; END OF SCRIPT
```

Finished downloading

Raw Swipe:

```
CRDTSTTRKTST%B5452300551227189^HOGAN/PAUL
^0804321000000072500000?TRKENDTRKTST;5452300551227189=080432100000007250?TR
KENDCRDEND|0000|%B5452300551227189^HOGAN/PAUL
^0804321000000072500000?|;5452300551227189=080432100000007250?||A1050000|01
0002AC501724CC063E08E2C52B53793DD53167753CDC3CE8EBC5C3555E30B68B73E4DB8912E6
372CA772E723EFEAADC02F02048C76|B000795|0000000000000000||DB3E||1234TXEND
```

Parsed Card Swipe:

```
CRDTST
  TRKTST
    %B5452300551227189^HOGAN/PAUL      ^0804321000000072500000?
  TRKEND
  TRKTST
    ;5452300551227189=080432100000007250?
  TRKEND
CRDEND
|0000
| %B5452300551227189^HOGAN/PAUL      ^0804321000000072500000?
| ;5452300551227189=080432100000007250?
|
|A1050000
|010002AC501724CC063E08E2C52B53793DD53167753CDC3CE8EBC5C3555E30B68B73E4DB891
2E6372CA772E723EFEAADC02F02048C76
|B000795
|0000000000000000
|
|DB3E
|
|1234
TXEND
```

### Example 3: HID reader card swipe in Security Level 2:

This example shows the data received in a HID report for a reader at Security Level 2.

Raw HID Report:

Byte	Content
0	00 00 00 3C 25 1F 00 25 42 35 34 35 32 33 30 30 35 35 31 32
20	32 37 31 38 39 5E 48 4F 47 41 4E 2F 50 41 55 4C 20 20 20 20
40	20 20 5E 30 38 30 34 33 32 31 30 30 30 30 30 30 30 37 32 35
60	30 30 30 30 30 30 3F 00 00 00 00 00 00 00 00 00 00 00 00 00
80	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
100	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 3B
120	35 34 35 32 33 30 30 35 35 31 32 32 37 31 38 39 3D 30 38 30
140	34 33 32 31 30 30 30 30 30 30 30 37 32 35 30 3F 00 00 00 00
160	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
180	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
200	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
220	00 00 00 00 00 00 00 00 00 00 00 00 00 3B 35 31 36 33 34 39 39 30
240	38 30 30 32 30 34 34 35 3D 30 30 30 30 30 30 30 30 30 30 30
260	30 3F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
280	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
300	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
320	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

```

340  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
360  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
380  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
400  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
420  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
440  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
460  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
480  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 02 00 00 00
500  00 00 00 00 00 3C 25 1F 25 42 35 34 35 32 33 30 30 35 35 31
520  32 32 37 31 38 39 5E 48 4F 47 41 4E 2F 50 41 55 4C 20 20 20
540  20 20 20 5E 30 38 30 34 33 32 31 30 30 30 30 30 30 30 37 32
560  35 30 30 30 30 30 30 3F 00 00 00 00 00 00 00 00 00 00 00
580  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
600  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
620  3B 35 34 35 32 33 30 30 35 35 31 32 32 37 31 38 39 3D 30 38
640  30 34 33 32 31 30 30 30 30 30 30 30 37 32 35 30 3F 00 00 00
660  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
680  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
700  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
720  00 00 00 00 00 00 00 00 00 00 00 00 00 3B 35 31 36 33 34 39 39
740  30 38 30 30 32 30 34 34 35 3D 30 30 30 30 30 30 30 30 30 30
760  30 30 3F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
780  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
800  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
820  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
840  00 00 00 00 00 00 00 00 00 00 00 00 3C 25 1F 36

```

According to the USB MagneSafe Swipe Reader Technical Reference Manual the HID report is broken down like this:

Offset	Usage Name
0	Track 1 decode status
1	Track 2 decode status
2	Track 3 decode status
3	Track 1 encrypted data length
4	Track 2 encrypted data length
5	Track 3 encrypted data length
6	Card encode type
7 - 118	Track 1 encrypted data
119 - 230	Track 2 encrypted data
231 - 342	Track 3 encrypted data
343	Card status
344 - 347	MagnePrint status
348	MagnePrint data length
349 - 476	MagnePrint data
477 - 492	Device serial number
493-494	Reader Encryption Status
495 - 504	DUKPT serial number/counter
505	Track 1 Masked data length
506	Track 2 Masked data length
507	Track 3 Masked data length
508 - 619	Track 1 Masked data
620 - 731	Track 2 Masked data
732 - 843	Track 3 Masked data
844 - 851	Encrypted Session ID
852	Track 1 Absolute data length
853	Track 2 Absolute data length



854           Track 3 Absolute data length  
 855           MagnePrint Absolute data length

Using this information, we can put the respective data from the Raw Data into the structure:

```

Offset       Usage Name
0            Track 1 decode status
              00
1            Track 2 decode status
              00
2            Track 3 decode status
              00
3            Track 1 encrypted data length
              3C (60 bytes, see Track 1 encrypted data below)
4            Track 2 encrypted data length
              25 (37 bytes, see Track 2 encrypted data below)
5            Track 3 encrypted data length
              1F (31 bytes, see Track 3 encrypted data below)
6            Card encode type
              00
7 - 118      Track 1 encrypted data (60 bytes, not encrypted, no keys yet)
              25 42 35 34 35 32 33 30 30 35 35 31 32 32 37 31 38 39 5E
48
              4F 47 41 4E 2F 50 41 55 4C 20 20 20 20 20 20 5E 30 38 30
34
              33 32 31 30 30 30 30 30 30 30 37 32 35 30 30 30 30 30
3F
              00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00
              00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00
              00 00 00 00 00 00 00 00 00 00 00 00 00
119 - 230    Track 2 encrypted data (37 bytes, not encrypted, no keys
yet)
              3B 35 34 35 32 33 30 30 35 35 31 32 32 37 31 38 39 3D 30
38
              30 34 33 32 31 30 30 30 30 30 30 30 37 32 35 30 3F 00 00
00
              00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00
              00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00
              00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00
              00 00 00 00 00 00 00 00 00 00 00 00 00
231 - 342    Track 3 encrypted data (31 bytes, not encrypted, no keys
yet)
              3B 35 31 36 33 34 39 39 30 38 30 30 32 30 34 34 35 3D 30
30
              30 30 30 30 30 30 30 30 30 30 30 3F 00 00 00 00 00 00 00
00
              00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00
              00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00
    
```

```
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
343 Card status
00 (not used, always zero)
344 - 347 MagnePrint status
00 00 00 00 (not available in Security Level 2)
348 MagnePrint data length
00 (Security Level 2, no MP data)
349 - 476 MagnePrint data (not available in Security Level 2)
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
477 - 492 Device serial number (not set, not filled)
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
493-494 Reader Encryption Status (Security Level 2, keys loaded)
00 02
495 - 504 DUKPT serial number/counter (Security Level 2, not
available)
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
505 Track 1 Masked data length
3C
506 Track 2 Masked data length
25
507 Track 3 Masked data length
1F
508 - 619 Track 1 Masked data (same as encrypted data)
25 42 35 34 35 32 33 30 30 35 35 31 32 32 37 31 38 39 5E
48 4F 47 41 4E 2F 50 41 55 4C 20 20 20 20 20 20 5E 30 38 30
34 33 32 31 30 30 30 30 30 30 30 37 32 35 30 30 30 30 30 30
3F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
620 - 731 Track 2 Masked data (same as encrypted data)
3B 35 34 35 32 33 30 30 35 35 31 32 32 37 31 38 39 3D 30
38 30 34 33 32 31 30 30 30 30 30 30 30 37 32 35 30 3F 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

```

00          00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
732 - 843   Track 3 Masked data (same as encrypted data)
            3B 35 31 36 33 34 39 39 30 38 30 30 32 30 34 34 35 3D 30
30          30 30 30 30 30 30 30 30 30 30 30 3F 00 00 00 00 00 00 00
00          00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00          00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00          00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00          00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
844 - 851   Encrypted Session ID (user didn't load, all zeroes)
            00 00 00 00 00 00 00 00
852         Track 1 Absolute data length (same as above)
            3C
853         Track 2 Absolute data length (same as above)
            25
854         Track 3 Absolute data length (same as above)
            1F
855         MagnePrint Absolute data length (same as above)
            36

```

**Example 4: Keyboard reader card swipe in Security Level 2, SureSwipe Mode:**

This example shows the data received from a KB swipe for a reader at Security Level 2. All properties are at their defaults making this a SureSwipe format.

```

Raw KB Data:
Byte   Content
  0    %B5452300551227189^HOGAN/PAUL      ^08043210000000
  50    725000000?;5452300551227189=0804321000000007250?+51
 100    63499080020445=0000000000000?

```

According to the USB MagneSafe Swipe Reader Technical Reference Manual and USB KB SureSwipe & USB KB Swipe Reader Technical Reference Manual (99875206) the KB Data (SureSwipe format) is broken down like this:

```

[Tk1 SS] [Tk1 Data] [ES] [Tk2 SS] [Tk2 Data] [ES] [Tk3 SS] [Tk3 Data] [ES]
[CR]

```

Each of the Pxx elements has the default value in this configuration, thus we can reinterpret the format as:

```

%[Tk1 Data]?;[Tk2 Data]?+[Tk3 Data]?<ENTER>

```

We can show this as:

```

%[Tk1 Data]?
;[Tk2 Data]?

```

```
+ [Tk3 Data]?  
<ENTER>
```

And we can fill this template in with the data we received to get:

```
%B5452300551227189^HOGAN/PAUL      ^08043210000000725000000?  
;5452300551227189=080432100000007250?  
+5163499080020445=0000000000000?
```

This shows the three tracks of data as received.

### Example 5: Keyboard reader card swipe in Security Level 2, Not SureSwipe Mode:

This example shows the data received from a KB swipe for a reader at Security Level 2 with the SureSwipe property set to 00 (False).

Raw KB Data:

Byte	Content
0	%B5452000000007189^HOGAN/PAUL      ^08040000000000
50	000000000?;5452000000007189=08040000000000000?+51
100	63000050000445=000000000000? 0200  %B54523005512271
150	89^HOGAN/PAUL      ^08043210000000725000000? ;5452
200	300551227189=080432100000007250? +5163499080020445
250	=000000000000?   0000000000000000 6F36  1000

According to the USB MagneSafe Swipe Reader Technical Reference Manual the KB Data is broken down like this:

```
[P30]  
[P32] [Tk1 SS] [Tk1 Masked Data] [ES] [P33]  
[P32] [Tk2 SS] [Tk2 Masked Data] [ES] [P33]  
[P32] [Tk3 SS] [Tk3 Masked Data] [ES] [P33]  
[P31]  
[P35] [Reader Encryption Status]  
[P35] [Tk1 Encrypted Data (including TK1 SS and ES)]  
[P35] [Tk2 Encrypted Data (including TK2 SS and ES)]  
[P35] [Tk3 Encrypted Data (including TK3 SS and ES)]  
[P35] [MagnePrint Status]  
[P35] [Encrypted MagnePrint data]  
[P35] [Device serial number]  
[P35] [Encrypted Session ID]  
[P35] [DUKPT serial number/counter]  
[P35] [Clear Text CRC]  
[P35] [Encrypted CRC]  
[P35] [Format Code]  
[P34]
```

Each of the Pxx elements has the default value in this configuration, thus we can reinterpret the format as:

```
%[Tk1 Masked Data]?  
;[Tk2 Masked Data]?  
+[Tk3 Masked Data]?  
|[Reader Encryption Status]
```

```

|[Tk1 Encrypted Data (including TK1 SS and ES)]
|[Tk2 Encrypted Data (including TK2 SS and ES)]
|[Tk3 Encrypted Data (including TK3 SS and ES)]
|[MagnePrint Status]
|[Encrypted MagnePrint data]
|[Device serial number]
|[Encrypted Session ID]
|[DUKPT serial number/counter]
|[Clear Text CRC]
|[Encrypted CRC]
|[Format Code]
<ENTER>

```

Using this information, we can put the respective data from the Raw Data into the structure:

```

%B545200000007189^HOGAN/PAUL      ^080400000000000000000000?
;5452000000007189=080400000000000000?
+5163000050000445=000000000000?
|0200
%B5452300551227189^HOGAN/PAUL      ^080432100000000725000000?
;5452300551227189=0804321000000007250?
+5163499080020445=000000000000?
|
|0000000000000000
|6F36
|1000

```

Note: The Device Serial Number field is empty because the DSN has not been set.

Note: The MagnePrint Status, the MagnePrint Data, the DUKPT serial number/counter and Encrypted CRC fields are empty because this reader is at Security Level 2 (encryption not enabled).

Note that at Security Level 2 the following fields are represented as ASCII characters:

- Masked Track data
- Encrypted Track data
- Format Code

Note that all other fields are represented as Hexadecimal data, that is two ASCII characters together give the value of a single byte.

### Example 6: Changing from Security Level 2 to Security Level 3:

```

; This script demonstrates changing from Security Level 2 to Security Level
3.
; It assumes the reader is at Security Level 2 with the ANSI X9.24 Example
; key loaded and the KSN counter set to 1.
09 00      ; Get current KSN (should be FFFF9876543210E00001)

```

```
Request      : CMND=09, LEN=00, DATA=
Response     : RC= 00, LEN=0A, DATA=FF FF 98 76 54 32 10 E0 00 01

; For KSN 1, MAC Key: 042666B4918430A3 68DE9628D03984C9
;
; The command to change Security Level looks like: 15 05 03 nnnnnnnn
; where nnnnnnnn is the MAC.
;
; The data to be MACd is: 15 05 03
; Data to be MACd must be in blocks of eight bytes, so we left justify and
; zero fill the block to get: 15 05 03 00 00 00 00 00 (This is the block to
MAC)
; For convenience show it as the compacted form: 1505030000000000
;
; The MAC algorithm run with this data uses the following cryptographic
; operations:
;
; Single DES Encrypt the data to be MACd with the left half of the MAC Key:
;   1505030000000000 1DES Enc with 042666B4918430A3 = BFBA7AE4C1597E3D
;
; Single DES Decrypt the result with the right half of the MAC Key:
;   BFBA7AE4C1597E3D 1DES Dec with 68DE9628D03984C9 = DA91AB9A8AD9AB4C
;
; Single DES Encrypt the result with the left half of the MAC Key:
;   DA91AB9A8AD9AB4C 1DES Enc with 042666B4918430A3 = E7E2FA3882BB386C
;
; The leftmost four bytes of the final result are the MAC = E7E2FA38
;
; Send the MACd Set Security Level command
15 05 03 E7E2FA38
Request      : CMND=15, LEN=05, DATA=03 E7 E2 FA 38
Response     : RC= 00, LEN=00, DATA=

02 00      ; Reset so changes take effect
Request     : CMND=02, LEN=00, DATA=
Response    : RC= 00, LEN=00, DATA=

Delay      : (waited 5 seconds)
09 00      ; Get current KSN (should be FFFF9876543210E00002)
Request    : CMND=09, LEN=00, DATA=
Response   : RC= 00, LEN=0A, DATA=FF FF 98 76 54 32 10 E0 00 02

15 00      ; Get current Security Level (Should be 03)
Request    : CMND=15, LEN=00, DATA=
Response   : RC= 00, LEN=01, DATA=03
```

**Example 7: Changing from Security Level 2 to Security Level 4:**

```
; This script demonstrates changing from Security Level 2 to Security Level
4.
; It assumes the reader is at Security Level 2 with the ANSI X9.24 Example
; key loaded and the KSN counter set to 1.
09 00      ; Get current KSN (should be FFFF9876543210E00001)
Request    : CMND=09, LEN=00, DATA=
Response   : RC= 00, LEN=0A, DATA=FF FF 98 76 54 32 10 E0 00 01
```

```

; For KSN 1, MAC Key: 042666B4918430A3 68DE9628D03984C9
;
; The command to change Security Level looks like: 15 05 04 nnnnnnnn
; where nnnnnnnn is the MAC.
;
; The data to be MACd is: 15 05 04
; Data to be MACd must be in blocks of eight bytes, so we left justify and
; zero fill the block to get: 15 05 04 00 00 00 00 00 (This is the block to
MAC)
; For convenience show it as the compacted form: 1505040000000000
;
; The MAC algorithm run with this data uses the following cryptographic
; operations:
;
; Single DES Encrypt the data to be MACd with the left half of the MAC Key:
; 1505040000000000 1DES Enc with 042666B4918430A3 = 644E76C88FFA0044
;
; Single DES Decrypt the result with the right half of the MAC Key:
; 644E76C88FFA0044 1DES Dec with 68DE9628D03984C9 = DEAC363779906C06
;
; Single DES Encrypt the result with the left half of the MAC Key:
; DEAC363779906C06 1DES Enc with 042666B4918430A3 = 2F38A60E3F6AD6AD
;
; The leftmost four bytes of the final result are the MAC = 2F38A60E
;
; Send the MACd Set Security Level command
15 05 04 2F38A60E
Request      : CMND=15, LEN=05, DATA=04 2F 38 A6 0E
Response    : RC= 00, LEN=00, DATA=

02 00      ; Reset so changes take effect
Request    : CMND=02, LEN=00, DATA=
Response   : RC= 00, LEN=00, DATA=

Delay      : (waited 5 seconds)
09 00      ; Get current KSN (should be FFFF9876543210E00002)
Request    : CMND=09, LEN=00, DATA=
Response   : RC= 00, LEN=0A, DATA=FF FF 98 76 54 32 10 E0 00 02

15 00      ; Get current Security Level (Should be 04)
Request    : CMND=15, LEN=00, DATA=
Response   : RC= 00, LEN=01, DATA=04

```

#### Example 8: Changing from Security Level 3 to Security Level 4:

```

; This script demonstrates changing from Security Level 3 to Security Level
4.
; It assumes the reader is at Security Level 3 with the ANSI X9.24 Example
; key loaded and the KSN counter set to 2.
09 00      ; Get current KSN (should be FFFF9876543210E00002)
Request    : CMND=09, LEN=00, DATA=
Response   : RC= 00, LEN=0A, DATA=FF FF 98 76 54 32 10 E0 00 02

; For KSN 2, MAC Key: C46551CEF9FDDBB0 AA9AD834130DC4C7
;
; The command to change Security Level looks like: 15 05 04 nnnnnnnn

```

```

; where nnnnnnnn is the MAC.
;
; The data to be MACd is: 15 05 04
; Data to be MACd must be in blocks of eight bytes, so we left justify and
; zero fill the block to get: 15 05 04 00 00 00 00 00 (This is the block to
MAC)
; For convenience show it as the compacted form: 1505040000000000
;
; The MAC algorithm run with this data uses the following cryptographic
; operations:
;
; Single DES Encrypt the data to be MACd with the left half of the MAC Key:
; 1505040000000000 1DES Enc with C46551CEF9FDDBB0 = 735323A914B9482E
;
; Single DES Decrypt the result with the right half of the MAC Key:
; 735323A914B9482E 1DES Dec with AA9AD834130DC4C7 = 390E2E2AC8CB4EE6
;
; Single DES Encrypt the result with the left half of the MAC Key:
; 390E2E2AC8CB4EE6 1DES Enc with C46551CEF9FDDBB0 = D9B7F3D8064C4B26
;
; The leftmost four bytes of the final result are the MAC = D9B7F3D8
;
; Send the MACd Set Security Level command
15 05 04 D9B7F3D8
Request      : CMND=15, LEN=05, DATA=04 D9 B7 F3 D8
Response    : RC= 00, LEN=00, DATA=

02 00      ; Reset so changes take effect
Request     : CMND=02, LEN=00, DATA=
Response    : RC= 00, LEN=00, DATA=

Delay      : (waited 5 seconds)
09 00      ; Get current KSN (should be FFFF9876543210E00003)
Request    : CMND=09, LEN=00, DATA=
Response   : RC= 00, LEN=0A, DATA=FF FF 98 76 54 32 10 E0 00 03

15 00      ; Get current Security Level (Should be 04)
Request    : CMND=15, LEN=00, DATA=
Response   : RC= 00, LEN=01, DATA=04

```

**Example 9: Configuring a reader after encryption is enabled (Security Level 3 or 4). In this example the reader is in Keyboard Mode:**

```

; This script demonstrates configuration commands for KB mode.
; It assumes the reader is at Security Level 3 or 4 and that the KSN counter
; is at 0x10.
09 00      ; Get current KSN (should be FFFF9876543210E00010)
Request    : CMND=09, LEN=00, DATA=
Response   : RC= 00, LEN=0A, DATA=FF FF 98 76 54 32 10 E0 00 10

; For this KSN counter the MAC Key is: 59598DCBD9BD6BC0 94165CE45358A057
00 01 02   ; Get current Polling Interval
Request    : CMND=00, LEN=01, DATA=02
Response   : RC= 00, LEN=01, DATA=01

; Form MAC for Set Property command

```



```

; Message to be sent is: 01 06 02 01 nnnnnnnn (nnnnnnnn is the MAC)
; Message to be MACd is: 0106020100000000
; This is the simplest MAC, simply TDES encrypt the message to be MACd with
; the MAC Key:
;         0106020100000000 MACd with 59598DCBD9BD6BC0 94165CE45358A057
; gets 8720CE23310961B5
; MAC is first four bytes: 8720CE23
01 06 02 01 8720CE23 ; Set Polling Interval to 1 ms
Request      : CMND=01, LEN=06, DATA=02 01 87 20 CE 23
Response     : RC= 00, LEN=00, DATA=

00 01 1E      ; Get current Pre Card String
Request      : CMND=00, LEN=01, DATA=1E
Response     : RC= 00, LEN=00, DATA=

; Form MAC for Set Property command
; Message to be sent is: 01 05 1E nnnnnnnn (nnnnnnnn is the MAC)
; Message to be MACd is: 01051E0000000000
; This is the simplest MAC, simply TDES encrypt the message to be MACd with
; the MAC Key:
;         01051E0000000000 MACd with 59598DCBD9BD6BC0 94165CE45358A057
; gets 5157FCBC179B0B95
; MAC is first four bytes: 5157FCBC
01 05 1E 5157FCBC ; Set to ""
Request      : CMND=01, LEN=05, DATA=1E 51 57 FC BC
Response     : RC= 00, LEN=00, DATA=

00 01 1F      ; Get current Post Card String
Request      : CMND=00, LEN=01, DATA=1F
Response     : RC= 00, LEN=00, DATA=

; Form MAC for Set Property command
; Message to be sent is: 01 05 1F nnnnnnnn (nnnnnnnn is the MAC)
; Message to be MACd is: 01051F0000000000
; This is the simplest MAC, simply TDES encrypt the message to be MACd with
; the MAC Key:
;         01051F0000000000 MACd with 2B5F01F4F0CCFAEA 639D523231BFE4A2
; gets 4885838CCC672376
; MAC is first four bytes: 4885838C
01 05 1F 4885838C ; Set to ""
Request      : CMND=01, LEN=05, DATA=1F 48 85 83 8C Response      : RC= 00,
LEN=00, DATA=
Response     : RC= 00, LEN=00, DATA=

00 01 20      ; Get current Pre Track String
Request      : CMND=00, LEN=01, DATA=20
Response     : RC= 00, LEN=00, DATA=

; Form MAC for Set Property command
; Message to be sent is: 01 05 20 nnnnnnnn (nnnnnnnn is the MAC)
; Message to be MACd is: 0105200000000000
; This is the simplest MAC, simply TDES encrypt the message to be MACd with
; the MAC Key:
;         0105200000000000 MACd with 9CF640F279C251E6 15F725EEEAC234AF
; gets 442A09E6588BBF04
; MAC is first four bytes: 442A09E6
01 05 20 442A09E6 ; Set to ""

```

## USB MagneSafe V5

---

```
Request      : CMND=01, LEN=05, DATA=20 44 2A 09 E6
Response     : RC= 00, LEN=00, DATA=

00 01 21    ; Get current Post Track String
Request      : CMND=00, LEN=01, DATA=21
Response     : RC= 00, LEN=00, DATA=

; Form MAC for Set Property command
; Message to be sent is: 01 05 21 nnnnnnnn (nnnnnnnn is the MAC)
; Message to be MACd is: 0105210000000000
; This is the simplest MAC, simply TDES encrypt the message to be MACd with
; the MAC Key:
;       0105210000000000 MACd with C3DF489FDF11ACB4 F03DE97C27DCB32F
; gets  1FA9A44C703099E1
; MAC is first four bytes: 1FA9A44C
01 05 21 1FA9A44C ; Set to ""
Request      : CMND=01, LEN=05, DATA=21 1F A9 A4 4C
Response     : RC= 00, LEN=00, DATA=

00 01 22    ; Get current Termination String
Request      : CMND=00, LEN=01, DATA=22
Response     : RC= 00, LEN=01, DATA=0D

; Form MAC for Set Property command
; Message to be sent is: 01 06 22 0D nnnnnnnn (nnnnnnnn is the MAC)
; Message to be MACd is: 0106220D00000000
; This is the simplest MAC, simply TDES encrypt the message to be MACd with
; the MAC Key:
;       0106220D00000000 MACd with 6584885077214CF1 4737FA93F92334D2
; gets  381AD461F2BDC522
; MAC is first four bytes: 381AD461
01 06 22 0D 381AD461 ; Set to "<ENTER>"
Request      : CMND=01, LEN=06, DATA=22 0D 38 1A D4 61
Response     : RC= 00, LEN=00, DATA=

00 01 2C    ; Get current Format Code
Request      : CMND=00, LEN=01, DATA=2C
Response     : RC= 00, LEN=05, DATA=31 FF FF FF FF

; Form MAC for Set Property command
; Message to be sent is: 01 09 2C 31303030 nnnnnnnn (nnnnnnnn is the MAC)
; Message to be MACd is: 01092C3130303000
; This is the simplest MAC, simply TDES encrypt the message to be MACd with
; the MAC Key:
;       01092C3130303000 MACd with E161D1956A6109D2 F37AFD7F9CC3969A
; gets  D153861529E88020
; MAC is first four bytes: D1538615
01 09 2C 31303030 D1538615 ; Set to "1000"
Request      : CMND=01, LEN=09, DATA=2C 31 30 30 30 D1538615
Response     : RC= 00, LEN=00, DATA=

02 00      ; Reset so changes take effect
Request      : CMND=02, LEN=00, DATA=
Response     : RC= 00, LEN=00, DATA=

Delay       : (waited 5 seconds)
00 01 02    ; Get current Polling Interval (should return 01)
```

```

Request      : CMND=00, LEN=01, DATA=02
Response     : RC= 00, LEN=01, DATA=01

00 01 1E     ; Get current Pre Card String (should return "")
Request      : CMND=00, LEN=01, DATA=1E
Response     : RC= 00, LEN=00, DATA=

00 01 1F     ; Get current Post Card String (should return "")
Request      : CMND=00, LEN=01, DATA=1F
Response     : RC= 00, LEN=00, DATA=

00 01 20     ; Get current Pre Track String (should return "")
Request      : CMND=00, LEN=01, DATA=20
Response     : RC= 00, LEN=00, DATA=

00 01 21     ; Get current Post Track String (should return "")
Request      : CMND=00, LEN=01, DATA=21
Response     : RC= 00, LEN=00, DATA=

00 01 22     ; Get current Termination String (should return "<ENTER>")
Request      : CMND=00, LEN=01, DATA=22
Response     : RC= 00, LEN=01, DATA=0D

00 01 2C     ; Get current Format Code
Request      : CMND=00, LEN=01, DATA=2C
Response     : RC= 00, LEN=04, DATA=31 30 30 30

```

**Example 10: Authentication. A reader already in Security Level 3 or 4 is put into the Authenticated Mode, allowed to stay in that mode for a time, then Deactivated.**

```

; This example demonstrates the Authentication Sequence.
; It is not scripted, some of the data is deliberately randomized. This
; makes it impossible for a simple script to produce the correct results.
; As an example it shows all the steps in authentication and deactivation.

```

```

; It assumes the reader is at Security Level 4, with the DUKPT KSN
; counter set to 2.

```

```

09 00        ; Get current KSN (should be FFFF9876543210E00002)

```

```

; Send the Activate Authenticated Mode command (4 minutes)

```

```

10 02 00F0

```

```

Request      : CMND=10, LEN=02, DATA=00 F0

```

```

Response     : RC= 00, LEN=1A, DATA=FF FF 98 76 54 32 10 E0 00 03 AA AA AA
AA AA AA AA AA DD DD DD DD DD DD DD DD

```

```

          |----- Current KSN -----| |----
Challenge 1 ----| |---- Challenge 2 ----|

```

```

Response     : RC= 00, LEN=1A, DATA=FF FF 98 76 54 32 10 E0 00 03 BE 5C 98
35 17 7E 45 2A A7 2D 2D B2 36 BF 29 D2

```

```

; Challenge 1 Encrypted: BE5C9835177E452A

```

```

; Challenge 2 Encrypted: A72D2DB236BF29D2

```

```

; Note that the KSN now ends with a counter of 3!

```

```

; Decrypt Challenge 1 using variant of Current Encryption Key

```

```

; (Current Encryption Key XOR with F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0)

```

```

;

```

```
; Current Key    0DF3D9422ACA561A 47676D07AD6BAD05
; XOR           F0F0F0F0F0F0F0F0 F0F0F0F0F0F0F0F0
; =            FD0329B2DA3AA6EA B7979DF75D9B5DF5
;
; BE5C9835177E452A TDES Dec with FD0329B2DA3AA6EA B7979DF75D9B5DF5 =
7549AB6EB4840003
;
; Note that the final two bytes of the result = 0003, matching the KSN as
; transmitted in the clear. This provides Authentication to the host that
; the reader is what it claims to be (proves key knowledge).
;
; Decrypt Challenge 2 using Current Encryption Key variant as above
; A72D2DB236BF29D2 TDES Dec with FD0329B2DA3AA6EA B7979DF75D9B5DF5 =
34DB9230698281B4
;
;
; Build an Activation Challenge Reply command (cmd, len, cryptogram)
; 11 08 XXXXXXXXXXXXXXXXXXXX
;
; The clear text input for the cryptogram is composed of the first six
bytes
; of the decrypted Challenge 1 followed by two bytes specifying how long to
; stay in the Authenticated Mode.
;
; CCCCCCCCCCCC TTTT
;
; Time examples:
; For 30 seconds use 002E
; For 99 seconds use 0063
; For 480 seconds use 01E0
; For 1200 seconds use 04B0
;
; These fields are concatenated to form an eight byte block, we will use
480
; seconds:
;
; CCCCCCCCCCCC01E0
;
; The block is encrypted using a variant of the Current Encryption Key
; (Current Encryption Key XOR with 3C3C3C3C3C3C3C3C3C3C3C3C3C3C3C3C)
;
; Current Key    0DF3D9422ACA561A 47676D07AD6BAD05
; XOR           3C3C3C3C3C3C3C3C 3C3C3C3C3C3C3C3C
; =            31CFE57E16F66A26 7B5B513B91579139
;
; 7549AB6EB48401E0 TDES Enc with 31CFE57E16F66A26 7B5B513B91579139 =
RRRRRRRRRRRRRRRR
;
; Send the Activation Challenge Reply Command
11 08 A30DDE3BFD629ACD

; Build a Deactivate Authenticated Mode command (cmd, len, cryptogram)
; 12 08 XXXXXXXXXXXXXXXXXXXX
;
; The clear text input for the cryptogram is composed of the first seven
bytes
; of the decrypted Challenge 2 followed by one byte specifying whether to
```

```

; increment the DUKPT KSN or not (00 = no increment, 01 = increment).
;
; DDDDDDDDDDDDDDD II
;
; These fields are concatenated to form an eight byte block, we will
specify
; No Increment:
;
; DDDDDDDDDDDDDDD00
;
; The block is encrypted using a variant of the Current Encryption Key
; (Current Encryption Key XOR with 3C3C3C3C3C3C3C3C3C3C3C3C3C3C3C)
;
; Current Key 0DF3D9422ACA561A 47676D07AD6BAD05
; XOR 3C3C3C3C3C3C3C3C 3C3C3C3C3C3C3C3C
; = 31CFE57E16F66A26 7B5B513B91579139
;
; 34DB923069828100 TDES Enc with 31CFE57E16F66A26 7B5B513B91579139 = CA
CB BD 5F 58 D5 C9 50
;
; Send the Deactivate Authenticated Mode command
12 08 CACBBD5F58D5C950

```

**Example 11: Swipe decryption, HID Reader in Security Level 3 or 4:**

This example shows the data received in a HID report for a reader at Security Level 3, KSN Count = 8. It will go on to show the steps to decrypt ALL the data received.

Raw HID Report:

Byte	Content
0	00 00 00 40 28 20 00 C2 5C 1D 11 97 D3 1C AA 87 28 5D 59 A8
20	92 04 74 26 D9 18 2E C1 13 53 C0 51 AD D6 D0 F0 72 A6 CB 34
40	36 56 0B 30 71 FC 1F D1 1D 9F 7E 74 88 67 42 D9 BE E0 CF D1
60	EA 10 64 C2 13 BB 55 27 8B 2F 12 00 00 00 00 00 00 00 00 00
80	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
100	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
120	4C 5D B7 D6 F9 01 C7 F0 FE AE 79 08 80 10 93 B3 DB FE 51 CC
140	F6 D4 83 E7 89 D7 D2 C0 07 D5 39 49 9B AA DC C8 D1 6C A2 00
160	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
180	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
200	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
220	00 00 00 00 00 00 00 00 00 00 00 00 76 BB 01 3C 0D FD 81 95 F1
240	6F 2F BC 50 A3 51 71 AA 37 01 31 F8 74 42 31 3E E3 64 57 B8
260	7C 87 F9 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
280	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
300	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
320	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
340	00 00 00 00 A1 05 00 00 38 47 03 57 6B C5 C2 CB 20 BC 04 C6
360	8B 5C E1 97 2A E8 9E 08 7B 1C 4D 47 D5 D0 E3 17 06 10 69 03
380	E6 0B 82 03 07 92 69 0A 57 1D B0 2D 0A 88 85 5A 35 AB B5 54
400	97 98 00 6B 42 00 00 00 00 00 00 00 00 00 00 00 00 00 00
420	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
440	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
460	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

```

480  00 00 00 00 00 00 00 00 00 00 00 00 00 06 FF FF 98 76 54
500  32 10 E0 00 08 3C 25 1F 25 42 35 34 35 32 30 30 30 30 30 30
520  30 30 37 31 38 39 5E 48 4F 47 41 4E 2F 50 41 55 4C 20 20 20
540  20 20 20 5E 30 38 30 34 30 30 30 30 30 30 30 30 30 30 30 30
560  30 30 30 30 30 30 30 30 3F 00 00 00 00 00 00 00 00 00 00 00
580  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
600  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
620  3B 35 34 35 32 30 30 30 30 30 30 30 30 37 31 38 39 3D 30 38
640  30 34 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 3F 00 00 00
660  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
680  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
700  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
720  00 00 00 00 00 00 00 00 00 00 00 00 00 3B 35 31 36 33 30 30 30
740  30 35 30 30 30 30 34 34 35 3D 30 30 30 30 30 30 30 30 30 30 30
760  30 30 3F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
780  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
800  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
820  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
840  00 00 00 00 21 68 5F 15 8B 5C 6B E0 3C 25 1F 36

```

According to the USB MagneSafe Swipe Reader Technical Reference Manual the HID report is broken down like this:

Offset	Usage Name
0	Track 1 decode status
1	Track 2 decode status
2	Track 3 decode status
3	Track 1 encrypted data length
4	Track 2 encrypted data length
5	Track 3 encrypted data length
6	Card encode type
7 - 118	Track 1 encrypted data
119 - 230	Track 2 encrypted data
231 - 342	Track 3 encrypted data
343	Card status
344 - 347	MagnePrint status
348	MagnePrint data length
349 - 476	MagnePrint data
477 - 492	Device serial number
493-494	Reader Encryption Status
495 - 504	DUKPT serial number/counter
505	Track 1 Masked data length
506	Track 2 Masked data length
507	Track 3 Masked data length
508 - 619	Track 1 Masked data
620 - 731	Track 2 Masked data
732 - 843	Track 3 Masked data
844 - 851	Encrypted Session ID
852	Track 1 Absolute data length
853	Track 2 Absolute data length
854	Track 3 Absolute data length
855	MagnePrint Absolute data length

Using this information, we can put the respective data from the Raw Data into the structure:

Offset	Usage Name
0	Track 1 decode status 00
1	Track 2 decode status 00
2	Track 3 decode status 00
3	Track 1 encrypted data length 40 (64 bytes, always in multiples of 8)
4	Track 2 encrypted data length 28 (40 bytes, always in multiples of 8)
5	Track 3 encrypted data length 20 (32 bytes, always in multiples of 8)
6	Card encode type (ISO/ABA) 00
7 - 118	Track 1 encrypted data C2 5C 1D 11 97 D3 1C AA 87 28 5D 59 A8 92 04 74 26 D9 18 2E C1 13 53 C0 51 AD D6 D0 F0 72 A6 CB 34 36 56 0B 30 71 FC 1F D1 1D 9F 7E 74 88 67 42 D9 BE E0 CF D1 EA 10 64 C2 13 BB 55 27 8B 2F 12 00
119 - 230	Track 2 encrypted data 72 4C 5D B7 D6 F9 01 C7 F0 FE AE 79 08 80 10 93 B3 DB FE 51 CC F6 D4 83 E7 89 D7 D2 C0 07 D5 39 49 9B AA DC C8 D1 6C A2 00
231 - 342	Track 3 encrypted data 76 BB 01 3C 0D FD 81 95 F1 6F 2F BC 50 A3 51 71 AA 37 01 31 F8 74 42 31 3E E3 64 57 B8 7C 87 F9 00
343	Card status 00 (not used, always zero)
344 - 347	MagnePrint status A1 05 00 00
348	MagnePrint data length

```

38
349 - 476   MagnePrint data
              47 03 57 6B C5 C2 CB 20 BC 04 C6 8B 5C E1 97 2A E8 9E 08
7B
              1C 4D 47 D5 D0 E3 17 06 10 69 03 E6 0B 82 03 07 92 69 0A
57
              1D B0 2D 0A 88 85 5A 35 AB B5 54 97 98 00 6B 42 00 00 00
00
              00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00
              00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00
              00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00
              00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00
              00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
477 - 492   Device serial number (not set, not filled)
              00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
493-494     Reader Encryption Status (keys loaded, encrypting)
              00 06
495 - 504   DUKPT serial number/counter
              FF FF 98 76 54 32 10 E0 00 08
505         Track 1 Masked data length
              3C
506         Track 2 Masked data length
              25
507         Track 3 Masked data length
              1F
508 - 619   Track 1 Masked data
              25 42 35 34 35 32 30 30 30 30 30 30 30 30 37 31 38 39 5E
48
              4F 47 41 4E 2F 50 41 55 4C 20 20 20 20 20 20 5E 30 38 30
34
              30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30
3F
              00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00
              00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00
              00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
620 - 731   Track 2 Masked data
              3B 35 34 35 32 30 30 30 30 30 30 30 30 37 31 38 39 3D 30
38
              30 34 30 30 30 30 30 30 30 30 30 30 30 30 30 30 3F 00 00
00
              00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00
              00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00
              00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00
              00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
732 - 843   Track 3 Masked data
              3B 35 31 36 33 30 30 30 30 35 30 30 30 30 34 34 35 3D 30
30
              30 30 30 30 30 30 30 30 30 30 30 3F 00 00 00 00 00 00 00
00

```



```

00          00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00          00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00          00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00          00 00 00 00 00 00 00 00 00 00 00 00 00
844 - 851   Encrypted Session ID (user didn't load, all zeroes)
            21 68 5F 15 8B 5C 6B E0
852         Track 1 Absolute data length
            3C
853         Track 2 Absolute data length
            25
854         Track 3 Absolute data length
            1F
855         MagnePrint Absolute data length
            36

```

The data is coherent structurally, let's work on decryption.

First, we note the KSN = FFFF9876543210E00008, counter is 8.  
 For the standard ANSI key example, counter 8 gets us the following  
 Encryption Key: 27F66D5244FF621E AA6F6120EDEB427F

There are five encrypted fields: Tracks 1, 2, and 3 encrypted data,  
 Encrypted MagnePrint data, Encrypted Session ID. We will show the  
 decryption  
 of each of these fields in detail. For convenience each will be grouped as  
 blocks of eight bytes.

```

Track 1 encrypted data
C2 5C 1D 11 97 D3 1C AA
87 28 5D 59 A8 92 04 74
26 D9 18 2E C1 13 53 C0
51 AD D6 D0 F0 72 A6 CB
34 36 56 0B 30 71 FC 1F
D1 1D 9F 7E 74 88 67 42
D9 BE E0 CF D1 EA 10 64
C2 13 BB 55 27 8B 2F 12
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00

```

As Track 1 Encrypted Data Length cites 64 bytes only, we can eliminate  
 the trailing blocks:

```

Block # 1   C25C1D1197D31CAA
          2   87285D59A8920474
          3   26D9182EC11353C0
          4   51ADD6D0F072A6CB
          5   3436560B3071FC1F
          6   D11D9F7E74886742
          7   D9BEE0CFD1EA1064
          8   C213BB55278B2F12

```

Appendix C tells us to decrypt the last block:

```
C213BB55278B2F12 TDES Dec with 27F66D5244FF621E AA6F6120EDEB427F
gets E98ED0F0D1EA1064
XOR D9BEE0CFD1EA1064
gets 3030303F00000000 (decrypted last block)
```

Continue on in reverse block order:

```
D9BEE0CFD1EA1064 TDES Dec with 27F66D5244FF621E AA6F6120EDEB427F
gets E12DA84C41B85772
XOR D11D9F7E74886742
gets 3030373235303030 (decrypted block 7)
```

Continue on in reverse block order:

```
D11D9F7E74886742 TDES Dec with 27F66D5244FF621E AA6F6120EDEB427F
gets 0704673B0041CC2F
XOR 3436560B3071FC1F
gets 3332313030303030 (decrypted block 6)
```

Continue on in reverse block order:

```
3436560B3071FC1F TDES Dec with 27F66D5244FF621E AA6F6120EDEB427F
gets 718DF68EC04A96FF
XOR 51ADD6D0F072A6CB
gets 2020205E30383034 (decrypted block 5)
```

Continue on in reverse block order:

```
51ADD6D0F072A6CB TDES Dec with 27F66D5244FF621E AA6F6120EDEB427F
gets 0989597B8D3373E0
XOR 26D9182EC11353C0
gets 2F5041554C202020 (decrypted block 4)
```

Continue on in reverse block order:

```
26D9182EC11353C0 TDES Dec with 27F66D5244FF621E AA6F6120EDEB427F
gets BF110311E7D5453A
XOR 87285D59A8920474
gets 38395E484F47414E (decrypted block 3)
```

Continue on in reverse block order:

```
87285D59A8920474 TDES Dec with 27F66D5244FF621E AA6F6120EDEB427F
gets F2692820A5E12B9B
XOR C25C1D1197D31CAA
gets 3035353132323731 (decrypted block 2)
```

Continue on in reverse block order:

```
C25C1D1197D31CAA TDES Dec with 27F66D5244FF621E AA6F6120EDEB427F
gets 2542353435323330 (decrypted block 1)
```

Ordering the decrypted blocks 1st to last we get:

HEX	ASCII
2542353435323330	%B545230
3035353132323731	05512271
38395E484F47414E	89^HOGAN
2F5041554C202020	/PAUL
2020205E30383034	^0804
3332313030303030	32100000
3030373235303030	00725000
3030303F00000000	000?

We can ignore the last four bytes because the Track 1 Absolute Length field cites only 60 characters.

ASCII string "%B5452300551227189^HOGAN/PAUL  
^08043210000000725000000?"

This is an accurate decryption of the track.

Track 2 encrypted data

```
72 4C 5D B7 D6 F9 01 C7
F0 FE AE 79 08 80 10 93
B3 DB FE 51 CC F6 D4 83
E7 89 D7 D2 C0 07 D5 39
49 9B AA DC C8 D1 6C A2
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
```

As Track 2 Encrypted Data Length cites 40 bytes only, we can eliminate the trailing blocks:

Block # 1	724C5DB7D6F901C7
2	F0FEAE7908801093
3	B3DBFE51CCF6D483
4	E789D7D2C007D539
5	499BAADCC8D16CA2

Appendix C tells us to decrypt the last block:

```
499BAADCC8D16CA2 TDES Dec with 27F66D5244FF621E AA6F6120EDEB427F
gets D0BBE2E2FF07D539
XOR E789D7D2C007D539
gets 373235303F000000 (decrypted last block)
```

Continue on in reverse block order:

```
E789D7D2C007D539 TDES Dec with 27F66D5244FF621E AA6F6120EDEB427F
gets 82EBCE61FCC6E4B3
XOR B3DBFE51CCF6D483
gets 3130303030303030 (decrypted block 4)
```

Continue on in reverse block order:

```
B3DBFE51CCF6D483 TDES Dec with 27F66D5244FF621E AA6F6120EDEB427F
```

```
gets C9C39E4138B423A1
  XOR F0FEAE7908801093
gets 393D303830343332  (decrypted block 3)
```

Continue on in reverse block order:

```
F0FEAE7908801093 TDES Dec with 27F66D5244FF621E AA6F6120EDEB427F
gets 47796C85E4CE30FF
  XOR 724C5DB7D6F901C7
gets 3535313232373138  (decrypted block 2)
```

Continue on in reverse block order:

```
724C5DB7D6F901C7 TDES Dec with 27F66D5244FF621E AA6F6120EDEB427F
gets 3B35343532333030  (decrypted block 1)
```

Ordering the decrypted blocks 1st to last we get:

HEX	ASCII
3B35343532333030	;5452300
3535313232373138	55122718
393D303830343332	9=080432
3130303030303030	10000000
373235303F000000	7250?

We can ignore the last three bytes because the Track 2 Absolute Length field cites only 37 characters.

ASCII string ";5452300551227189=080432100000007250?"

This is an accurate decryption of the track.

Track 3 encrypted data

```
76 BB 01 3C 0D FD 81 95
F1 6F 2F BC 50 A3 51 71
AA 37 01 31 F8 74 42 31
3E E3 64 57 B8 7C 87 F9
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
```

We hope the preceding examples have given you enough information to decrypt this

track, the final results should be the ASCII string  
";5163499080020445=000000000000?"

MagnePrint data

```
47 03 57 6B C5 C2 CB 20
BC 04 C6 8B 5C E1 97 2A
E8 9E 08 7B 1C 4D 47 D5
D0 E3 17 06 10 69 03 E6
0B 82 03 07 92 69 0A 57
1D B0 2D 0A 88 85 5A 35
```

```

AB B5 54 97 98 00 6B 42
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00

```

As MagnePrint Data Length cites 56 bytes only, we can eliminate the trailing blocks:

```

Block # 1 4703576BC5C2CB20
        2  BC04C68B5CE1972A
        3  E89E087B1C4D47D5
        4  D0E31706106903E6
        5  0B82030792690A57
        6  1DB02D0A88855A35
        7  ABB5549798006B42

```

Appendix C tells us to decrypt the last block:

```

ABB5549798006B42 TDES Dec with 27F66D5244FF621E AA6F6120EDEB427F
  gets D3B7EDDFD3045A35
  XOR 1DB02D0A88855A35
  gets CE07C0D55B810000 (decrypted last block)

```

Continue on in reverse block order:

```

1DB02D0A88855A35 TDES Dec with 27F66D5244FF621E AA6F6120EDEB427F
  gets B52307C37D314482
  XOR 0B82030792690A57
  gets BEA104C4EF584ED5 (decrypted block 6)

```

Continue on in reverse block order:

```

0B82030792690A57 TDES Dec with 27F66D5244FF621E AA6F6120EDEB427F
  gets AF4EABEE4973E402
  XOR D0E31706106903E6
  gets 7FADBCE8591AE7E4 (decrypted block 5)

```

Continue on in reverse block order:

```

D0E31706106903E6 TDES Dec with 27F66D5244FF621E AA6F6120EDEB427F
  gets 269870C3659D905E
  XOR E89E087B1C4D47D5
  gets CE0678B879D0D78B (decrypted block 4)

```

Continue on in reverse block order:

```

E89E087B1C4D47D5 TDES Dec with 27F66D5244FF621E AA6F6120EDEB427F
  gets 7B8F912DAF1B3149
  XOR BC04C68B5CE1972A
  gets C78B57A6F3FAA663 (decrypted block 3)

```

Continue on in reverse block order:

```

BC04C68B5CE1972A TDES Dec with 27F66D5244FF621E AA6F6120EDEB427F
  gets 078FD0419993F7B0
  XOR 4703576BC5C2CB20
  gets 408C872A5C513C90 (decrypted block 2)

```

Continue on in reverse block order:

```
4703576BC5C2CB20 TDES Dec with 27F66D5244FF621E AA6F6120EDEB427F
gets 01000184EA10B939 (decrypted block 1)
```

Ordering the decrypted blocks 1st to last we get:

```
HEX
01000184EA10B939
408C872A5C513C90
C78B57A6F3FAA663
CE0678B879D0D78B
7FADBCE8591AE7E4
BEA104C4EF584ED5
CE07C0D55B810000
```

We can ignore the last four bytes because the MagnePrint Data Absolute Length field cites only 54 characters.

```
01000184EA10B939408C872A5C513C90C78B57A6F3FAA663CE0678B879D0D78B7FADBCE8591A
E7E4BEA104C4EF584ED5CE07C0D55B81
```

This is an accurate decryption of the MagnePrint data.

Encrypted Session ID (user didn't load, all zeroes)

```
21 68 5F 15 8B 5C 6B E0
```

As this is a simple eight byte block, we only need decrypt it with the appropriate key:

```
21685F158B5C6BE0 TDES Dec with 27F66D5244FF621E AA6F6120EDEB427F
gets 0000000000000000
```

This is an accurate decryption of the Encrypted Session ID, which was not loaded by the user and thus was all zeroes.

### Example 12: Swipe decryption, Keyboard Reader in Security Level 3 or 4:

This example shows the data received in a KB swipe for a reader at Security Level 3, KSN Count = 8. It will go on to show the steps to decrypt ALL the data received.

Raw KB Data:

Byte	Content
0	%B5452000000007189^HOGAN/PAUL ^08040000000000
50	000000000?;5452000000007189=0804000000000000000?+51
100	63000050000445=000000000000? 0600 C25C1D1197D31CAA
150	87285D59A892047426D9182EC11353C051ADD6D0F072A6CB34
200	36560B3071FC1FD11D9F7E74886742D9BEE0CFD1EA1064C213
250	BB55278B2F12 724C5DB7D6F901C7F0FEAE7908801093B3DBF
300	E51CCF6D483E789D7D2C007D539499BAADCC8D16CA2 E31234
350	A91059A0FBFE627954EE21868AEE3979540B67FCC40F61CECA
400	54152D1E A1050000 8628E664C59BBAA232BA90BFB3E6B41D
450	6F4B691E633C311CBE6EE7466B81196EC07B12648DCAC4FD7F
500	D0E212B479C60BAD8C74F82F327667  21685F158B5C6BE0 F
550	FFF9876543210E00008 B78F  0000

According to the USB MagneSafe Swipe Reader Technical Reference Manual the

KB Data is broken down like this:

```
[P30]
[P32] [Tk1 SS] [Tk1 Masked Data] [ES] [P33]
[P32] [Tk2 SS] [Tk2 Masked Data] [ES] [P33]
[P32] [Tk3 SS] [Tk3 Masked Data] [ES] [P33]
[P31]
[P35] [Reader Encryption Status]
[P35] [Tk1 Encrypted Data (including TK1 SS and ES)]
[P35] [Tk2 Encrypted Data (including TK2 SS and ES)]
[P35] [Tk3 Encrypted Data (including TK3 SS and ES)]
[P35] [MagnePrint Status]
[P35] [Encrypted MagnePrint data]
[P35] [Device serial number]
[P35] [Encrypted Session ID]
[P35] [DUKPT serial number/counter]
[P35] [Clear Text CRC]
[P35] [Encrypted CRC]
[P35] [Format Code]
[P34]
```

Each of the Pxx elements has the default value in this configuration, thus we can reinterpret the format as:

```
%[Tk1 Masked Data]?
;[Tk2 Masked Data]?
+[Tk3 Masked Data]?
|[Reader Encryption Status]
|[Tk1 Encrypted Data (including TK1 SS and ES)]
|[Tk2 Encrypted Data (including TK2 SS and ES)]
|[Tk3 Encrypted Data (including TK3 SS and ES)]
|[MagnePrint Status]
|[Encrypted MagnePrint data]
|[Device serial number]
|[Encrypted Session ID]
|[DUKPT serial number/counter]
|[Clear Text CRC]
|[Encrypted CRC]
|[Format Code]
<ENTER>
```

Using this information, we can put the respective data from the Raw Data into the structure:

```
%B5452000000007189^HOGAN/PAUL      ^08040000000000000000000000000000?
;54520000000007189=08040000000000000000000000000000?
+5163000050000445=00000000000000000000000000000000?
|0600
|C25C1D1197D31CAA87285D59A892047426D9182EC11353C051ADD6D0F072A6CB3436560B307
1FC1FD11D9F7E74886742D9BEE0CFD1EA1064C213BB55278B2F12
|724C5DB7D6F901C7F0FEAE7908801093B3DBFE51CCF6D483E789D7D2C007D539499BAADCC8D
16CA2
|E31234A91059A0FBFE627954EE21868AEE3979540B67FCC40F61CECA54152D1E
|A1050000
```

```
|8628E664C59BBAA232BA90BFB3E6B41D6F4B691E633C311CBE6EE7466B81196EC07B12648DC
AC4FD7FD0E212B479C60BAD8C74F82F327667
|
|21685F158B5C6BE0
|FFFF9876543210E00008
|B78F
|
|0000
```

Note: The Device Serial Number field is empty because the DSN has not been set.

Note: The Encrypted CRC field is empty because the default configuration is to send it empty.

Note that at Security Level 3 the following fields are represented as ASCII characters:

- Masked Track data
- Format Code

Note that all other fields are represented as Hexadecimal data, that is two ASCII characters together give the value of a single byte.

The data is coherent structurally, let's work on decryption.

First, we note the KSN = FFFF9876543210E00008, counter is 8.  
For the standard ANSI key example, counter 8 gets us the following  
Encryption Key: 27F66D5244FF621E AA6F6120EDEB427F

There are five encrypted fields: Tracks 1, 2, and 3 encrypted data, Encrypted MagnePrint data, Encrypted Session ID. We will show the decryption of each of these fields in detail. For convenience each will be grouped as blocks of eight bytes.

```
Track 1 encrypted data
Block # 1  C25C1D1197D31CAA
          2  87285D59A8920474
          3  26D9182EC11353C0
          4  51ADD6D0F072A6CB
          5  3436560B3071FC1F
          6  D11D9F7E74886742
          7  D9BEE0CFD1EA1064
          8  C213BB55278B2F12
```

```
Appendix C tells us to decrypt the last block:
C213BB55278B2F12 TDES Dec with 27F66D5244FF621E AA6F6120EDEB427F
gets E98ED0F0D1EA1064
XOR D9BEE0CFD1EA1064
gets 3030303F00000000 (decrypted last block)
```

```
Continue on in reverse block order:
D9BEE0CFD1EA1064 TDES Dec with 27F66D5244FF621E AA6F6120EDEB427F
gets E12DA84C41B85772
XOR D11D9F7E74886742
gets 3030373235303030 (decrypted block 7)
```



```
Continue on in reverse block order:
D11D9F7E74886742 TDES Dec with 27F66D5244FF621E AA6F6120EDEB427F
gets 0704673B0041CC2F
XOR 3436560B3071FC1F
gets 3332313030303030 (decrypted block 6)
```

```
Continue on in reverse block order:
3436560B3071FC1F TDES Dec with 27F66D5244FF621E AA6F6120EDEB427F
gets 718DF68EC04A96FF
XOR 51ADD6D0F072A6CB
gets 2020205E30383034 (decrypted block 5)
```

```
Continue on in reverse block order:
51ADD6D0F072A6CB TDES Dec with 27F66D5244FF621E AA6F6120EDEB427F
gets 0989597B8D3373E0
XOR 26D9182EC11353C0
gets 2F5041554C202020 (decrypted block 4)
```

```
Continue on in reverse block order:
26D9182EC11353C0 TDES Dec with 27F66D5244FF621E AA6F6120EDEB427F
gets BF110311E7D5453A
XOR 87285D59A8920474
gets 38395E484F47414E (decrypted block 3)
```

```
Continue on in reverse block order:
87285D59A8920474 TDES Dec with 27F66D5244FF621E AA6F6120EDEB427F
gets F2692820A5E12B9B
XOR C25C1D1197D31CAA
gets 3035353132323731 (decrypted block 2)
```

```
Continue on in reverse block order:
C25C1D1197D31CAA TDES Dec with 27F66D5244FF621E AA6F6120EDEB427F
gets 2542353435323330 (decrypted block 1)
```

```
Ordering the decrypted blocks 1st to last we get:
HEX          ASCII
2542353435323330 %B545230
3035353132323731 05512271
38395E484F47414E 89^HOGAN
2F5041554C202020 /PAUL
2020205E30383034 ^0804
3332313030303030 32100000
3030373235303030 00725000
3030303F00000000 000?
```

We can ignore the last four bytes because they are all hex 00 and fall after the End Sentinel.

```
ASCII string "%B5452300551227189^HOGAN/PAUL
^08043210000000725000000?"
```

This is an accurate decryption of the track.

```
Track 2 encrypted data
Block # 1 724C5DB7D6F901C7
        2 F0FEAE7908801093
```

```
3 B3DBFE51CCF6D483
4 E789D7D2C007D539
5 499BAADCC8D16CA2
```

Appendix C tells us to decrypt the last block:

```
499BAADCC8D16CA2 TDES Dec with 27F66D5244FF621E AA6F6120EDEB427F
gets D0BBE2E2FF07D539
XOR E789D7D2C007D539
gets 373235303F000000 (decrypted last block)
```

Continue on in reverse block order:

```
E789D7D2C007D539 TDES Dec with 27F66D5244FF621E AA6F6120EDEB427F
gets 82EBCE61FCC6E4B3
XOR B3DBFE51CCF6D483
gets 3130303030303030 (decrypted block 4)
```

Continue on in reverse block order:

```
B3DBFE51CCF6D483 TDES Dec with 27F66D5244FF621E AA6F6120EDEB427F
gets C9C39E4138B423A1
XOR F0FEAE7908801093
gets 393D303830343332 (decrypted block 3)
```

Continue on in reverse block order:

```
F0FEAE7908801093 TDES Dec with 27F66D5244FF621E AA6F6120EDEB427F
gets 47796C85E4CE30FF
XOR 724C5DB7D6F901C7
gets 3535313232373138 (decrypted block 2)
```

Continue on in reverse block order:

```
724C5DB7D6F901C7 TDES Dec with 27F66D5244FF621E AA6F6120EDEB427F
gets 3B35343532333030 (decrypted block 1)
```

Ordering the decrypted blocks 1st to last we get:

HEX	ASCII
3B35343532333030	;5452300
3535313232373138	55122718
393D303830343332	9=080432
3130303030303030	10000000
373235303F000000	7250?

We can ignore the last three bytes because they are all hex 00 and fall after the End Sentinel.

ASCII string ";5452300551227189=080432100000007250?"

This is an accurate decryption of the track.

Track 3 encrypted data

```
Block # 1 E31234A91059A0FB
2 FE627954EE21868A
3 EE3979540B67FCC4
4 0F61CECA54152D1E
```

Appendix C tells us to decrypt the last block:

```
0F61CECA54152D1E TDES Dec with 27F66D5244FF621E AA6F6120EDEB427F
```

```
gets DE0949643B57C3C4
  XOR EE3979540B67FCC4
gets 3030303030303030F00  (decrypted last block)
```

Continue on in reverse block order:

```
EE3979540B67FCC4 TDES Dec with 27F66D5244FF621E AA6F6120EDEB427F
gets CB5F4964DE11B6BA
  XOR FE627954EE21868A
gets 353D303030303030  (decrypted block 3)
```

Continue on in reverse block order:

```
FE627954EE21868A TDES Dec with 27F66D5244FF621E AA6F6120EDEB427F
gets D32A0499226994CF
  XOR E31234A91059A0FB
gets 3038303032303434  (decrypted block 2)
```

Continue on in reverse block order:

```
E31234A91059A0FB TDES Dec with 27F66D5244FF621E AA6F6120EDEB427F
gets 2B35313633343939  (decrypted block 1)
```

Ordering the decrypted blocks 1st to last we get:

HEX	ASCII
2B35313633343939	+5163499
3038303032303434	08002044
353D303030303030	3=000000
3030303030303030F00	000000?

We can ignore the last byte because it is hex 00 and falls after the End Sentinel.

ASCII string "+5163499080020443=000000000000? "

This is an accurate decryption of the track.

MagnePrint data

```
Block # 1 8628E664C59BBAA2
        2 32BA90BFB3E6B41D
        3 6F4B691E633C311C
        4 BE6EE7466B81196E
        5 C07B12648DCAC4FD
        6 7FD0E212B479C60B
        7 AD8C74F82F327667
```

Appendix C tells us to decrypt the last block:

```
AD8C74F82F327667 TDES Dec with 27F66D5244FF621E AA6F6120EDEB427F
gets 09162DCA11E5C60B
  XOR 7FD0E212B479C60B
gets 76C6CFD8A59C0000  (decrypted last block)
```

Continue on in reverse block order:

```
7FD0E212B479C60B TDES Dec with 27F66D5244FF621E AA6F6120EDEB427F
gets AE81BFA4A2C80006
  XOR C07B12648DCAC4FD
gets 6EFAADC02F02C4FB  (decrypted block 6)
```

Continue on in reverse block order:

```
C07B12648DCAC4FD TDES Dec with 27F66D5244FF621E AA6F6120EDEB427F
gets AAC8D06ACCF27E6D
XOR BE6EE7466B81196E
gets 14A6372CA7736703 (decrypted block 5)
```

```
Continue on in reverse block order:
BE6EE7466B81196E TDES Dec with 27F66D5244FF621E AA6F6120EDEB427F
gets 01D78CB7D1DAEA95
XOR 6F4B691E633C311C
gets 6E9CE5A9B2E6DB89 (decrypted block 4)
```

```
Continue on in reverse block order:
6F4B691E633C311C TDES Dec with 27F66D5244FF621E AA6F6120EDEB427F
gets 0D2620B051231748
XOR 32BA90BFB3E6B41D
gets 3F9CB00FE2C5A355 (decrypted block 3)
```

```
Continue on in reverse block order:
32BA90BFB3E6B41D TDES Dec with 27F66D5244FF621E AA6F6120EDEB427F
gets 41499B60A6AAD427
XOR 8628E664C59BBAA2
gets C7617D0463316E85 (decrypted block 2)
```

```
Continue on in reverse block order:
8628E664C59BBAA2 TDES Dec with 27F66D5244FF621E AA6F6120EDEB427F
gets 010002D4B69CD2C0 (decrypted block 1)
```

\*\*\*\*\*

Ordering the decrypted blocks 1st to last we get:

```
HEX
010002D4B69CD2C0
C7617D0463316E85
3F9CB00FE2C5A355
6E9CE5A9B2E6DB89
14A6372CA7736703
6EFAADC02F02C4FB
76C6CFD8A59C0000
```

We can ignore the last two bytes because we know the MagnePrint data is actually 54 bytes long.

```
010002D4B69CD2C0C7617D0463316E853F9CB00FE2C5A3556E9CE5A9B2E6DB8914A6372C
A77367036EFAADC02F02C4FB76C6CFD8A59C0000
```

This is an accurate decryption of the MagnePrint data.

```
Encrypted Session ID (user didn't load, all zeroes)
21685F158B5C6BE0
```

```
As this is a simple eight byte block, we only need decrypt it with
the appropriate key:
21685F158B5C6BE0 TDES Dec with 27F66D5244FF621E AA6F6120EDEB427F
gets 0000000000000000
```

This is an accurate decryption of the Encrypted Session ID, which was not loaded by the user and thus was all zeroes.

## APPENDIX E. IDENTIFYING ISO/ABA AND AAMVA CARDS

### ISO/ABA FINANCIAL CARDS

1. If low level decoding algorithm finds data for available tracks to be in the ISO format particular to each track, the card is classified as ISO. In order to be considered for ISO Financial masking, the card must first be classed as ISO.
2. In order for any track on a card to be considered for ISO/ABA masking, the card must be classified as ISO by the low level decoding algorithm.
3. ISO/ABA masking is considered for each track independently. One track may qualify for masking and another not.
4. Track 1
  - a. The goal is to send the Format Code in the clear, the PAN partially masked, the Name and Expiration Date in the clear, and the rest of the track masked.
  - b. If Format Code, PAN, Name, or Expiration Date are not correctly structured, the rest of the track (from the point of discrepancy) will be sent in the clear.
  - c. If the Format Code, PAN, Name, or Expiration Date contain the '?' character (End Sentinel), the field is not correctly structured.
  - d. A correctly structured Format Code is the first character on the card and contains the character 'B'.
  - e. A correctly structured PAN has a maximum of 19 digits and is ended by the character '^' (Field Separator).
  - f. A correctly structured Name has a maximum of 26 characters and is ended by the character '^' (Field Separator).
  - g. A correctly structured Expiration Date has 4 characters.
5. Tracks 2 & 3
  - a. The goal is to send the PAN partially masked, the Expiration Date in the clear, and the rest of the track masked.
  - b. If the PAN or Expiration Date are not correctly structured, the rest of the track (from the point of discrepancy) will be sent in the clear.
  - c. If the PAN or Expiration Date contain the '?' character (End Sentinel), the field is not correctly structured.
  - d. A correctly structured PAN has a maximum of 19 digits and is ended by the character '=' (Field Separator).
  - e. A correctly structured Expiration Date has 4 characters.

### AAMVA DRIVER LICENSES

1. If the card reader reads three tracks of data and Track 1 is formatted per ISO Track 1 rules, Track 2 is formatted per ISO Track 2 rules, and Track 3 is formatted per ISO Track 1 rules, the card is considered to be an AAMVA card. Some MagTek readers do not support reading of Track 3, so this rule will not apply on such readers.
2. If low level decoding algorithm finds data for available tracks to be in the ISO format particular to each track, and Track 2 contains a correctly structured PAN field whose first 6 digits are "604425" or contain values in the range "636000" to "636062" inclusive, the card is considered to be an AAMVA card.
3. AAMVA card masking, when enabled, works as follows:

- a. Tracks 1 & 3 are sent entirely masked i.e., zeros are supplied in all character positions.
- b. Track 2:
  - The goal is to send the Driver License ID (DLID) partially masked, the Expiration Date in the clear, the Birth Date in the clear, and the rest of the track masked.
  - If the DLID, Expiration Date, or Birth Date are not correctly structured, the rest of the track (from the point of discrepancy) will be sent in the clear.
  - If the DLID, Expiration Date, or Birth Date contain the '?' character (End Sentinel), the field is not correctly structured.
  - A correctly structured DLID has a maximum of 19 digits and is ended by the character '=' (Field Separator).
  - A correctly structured Expiration Date has 4 characters.
  - A correctly structured Birth Date has 8 characters.