

iDynamo MagneSafe V5

COMMUNICATION REFERENCE MANUAL

PART NUMBER 99875483-3

APRIL 2011

MAGTEK[®]
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

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.

iPhone, iPod and iPad are trademarks of Apple Inc., registered in the U.S. and other countries.

REVISIONS

Rev Number	Date	Notes
1.01	February 2, 2010	Initial Release
2.01	May 24, 2010	Changed card transmit buffer size to 500, added new property to enable/disable ASIC, modified several descriptions to make them consistent with other manuals
3.01	April 13, 2011	Updated MP Flags Property

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 numérique 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

SECTION 1. SECURITY	1
SECURITY LEVEL 3	1
COMMANDS AND SECURITY LEVELS.....	2
SECTION 2. COMMUNICATIONS.....	3
CARD DATA	3
Masked Track Data	4
Reader Encryption Status.....	6
Encrypted Track Data.....	7
MagnePrint Status	8
Encrypted MagnePrint Data	8
Device Serial Number.....	8
Encrypted Session ID	9
DUKPT Key Serial Number	9
Encryption Counter.....	9
Clear Text CRC	9
Encrypted CRC.....	9
Format Code.....	9
PROGRAMMABLE CONFIGURATION OPTIONS	10
COMMANDS	10
PRIVILEGED COMMANDS.....	11
COMMAND NUMBER	11
DATA LENGTH.....	11
DATA	11
RESULT CODE	11
GET AND SET PROPERTY COMMANDS	12
Result Codes	12
Property ID.....	13
Property Default Values.....	14
SOFTWARE ID PROPERTY	14
DEVICE SERIAL NUMBER PROPERTY	14
MAGNESAFE VERSION NUMBER PROPERTY	15
TRACK ID ENABLE PROPERTY	15
ISO Track Mask Property	16
AAMVA Track Mask Property.....	16
INTERFACE TYPE PROPERTY	17
TRACK DATA SEND FLAGS PROPERTY	17
MP FLAGS PROPERTY.....	18
CRC FLAG PROPERTY.....	19
DECODE ENABLE PROPERTY	19
SS JIS TYPE 2 PROPERTY	20
ES JIS TYPE 2 PROPERTY	20
PRE CARD STRING PROPERTY.....	21
POST CARD STRING PROPERTY	21
PRE-TRACK STRING PROPERTY	22
POST TRACK STRING PROPERTY	23
TERMINATION STRING PROPERTY	23
FS PROPERTY	24
SS TRACK 1 ISO ABA PROPERTY	24
SS TRACK 2 ISO ABA PROPERTY	24
SS TRACK 3 ISO ABA PROPERTY	25
SS TRACK 3 AAMVA PROPERTY	25
SS TRACK 2 7BITS PROPERTY.....	25
SS TRACK 3 7BITS PROPERTY.....	26
ES PROPERTY	26
FORMAT CODE PROPERTY	26
ES TRACK 1 PROPERTY.....	27
ES TRACK 2 PROPERTY	27

ES TRACK 3 PROPERTY	27
SEND ENCRYPTION COUNTER PROPERTY	28
MASK OTHER CARDS PROPERTY	28
SEND CLEAR AAMVA CARD DATA PROPERTY	28
RESET DEVICE COMMAND	29
DUKPT OPERATION	29
Get DUKPT KSN and Counter Command.....	30
SET SESSION ID COMMAND	30
ACTIVATE AUTHENTICATED MODE COMMAND.....	31
ACTIVATION CHALLENGE REPLY COMMAND	32
DEACTIVATE AUTHENTICATED MODE COMMAND.....	33
GET READER STATE COMMAND.....	34
GET ENCRYPTION COUNTER COMMAND.....	35
ENCRYPT BULK DATA COMMAND	36
READ ASIC CONTROL COMMAND.....	37
APPENDIX A. GUIDE ON DECRYPTING DATA	39
APPENDIX B. COMMAND EXAMPLE	41
APPENDIX C. IDENTIFYING ISO/ABA AND AAMVA CARDS	49
ISO/ABA FINANCIAL CARDS	49
AAMVA DRIVER LICENSES.....	50

SECTION 1. SECURITY

This iDynamo is a secure card reader authenticator (SCRA) designed to work with the iPhone 4, iPhone 3GS, iPhone 3G, iPod Touch and iPad. 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

This reader only supports Security Level 3.

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.

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, which can 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 3
Get Property	Y
Set Property	S
Reset	X*
Get DUKPT KSN and Counter	Y
Set Session ID	Y
Activate Authenticated Mode	Y
Activation Challenge Reply	Y
Deactivate Authenticated Mode	Y
Get Reader State	Y
Get Encryption Counter	Y
Bulk Encrypt	Y

- * The Reset command has special behavior. When an Authentication sequence has failed, only a correctly MACed 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. COMMUNICATIONS

CARD DATA

The details about how the card data and commands are structured follow later in this section.

The reader will send only one swipe message per card swipe. When a card is swiped, the swipe message will be sent even if the data is not decodable. 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.

The reader will always send data in blocks of 500 bytes. If card data is more than 500 bytes, the reader will send this using 2 blocks of 500 bytes. If card data is less than or equal to 500 bytes, the reader will only send 1 block with 500 bytes. If data is less than 500 bytes in a block, the reader will use a lower case ‘x’ (0x78) as padding characters. Note: The longest message always fits within 2 blocks.

A Swipe Message is composed of readable ASCII characters. It includes:

- Structural ASCII characters intended to give clues to the structure of the rest of the data.
- Simple ASCII fields that convey the ASCII representation of:
 - Masked Track Data
 - Device Serial Number
 - Format Code
- Binary fields that use sets of two ASCII characters representing hexadecimal digits to convey the binary value of each byte in the field. The ASCII characters 0123456789ABCDEF convey the hexadecimal values of 0123456789ABCDEF respectively. The Binary fields are:
 - Reader Encryption Status
 - Encrypted Track Data
 - MagnePrint Status
 - Encrypted MagnePrint Data
 - Encrypted Session ID
 - DUKPT Key Serial Number
 - Clear Text CRC
 - Encrypted CRC

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 have values outside of the printable ASCII character range).

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 TK1 SS and ES)]
 [P35] [Tk3 Encrypted Data (including TK1 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., P31) is the decimal value of the property ID in the command list (see **Pre Card String**).

Label	Property ID	P-Value	Description	Default
	0x1E	P30	Pre card string	0 (0x00)
	0x1F	P31	Post card string	0 (0x00)
	0x20	P32	Pre track string	0 (0x00)
	0x21	P33	Post track string	0 (0x00)
	0x22	P34	Terminating string	C/R (0x0D)
	0x23	P35	Programmable field separator	" " (0x7C)
Tk1 SS	0x24	P36	ISO/ABA start sentinel	"0%" (0x25)
Tk2-SS	0x25	P37	ISO/ABA 5-bit start sentinel	"." (0x3B)
Tk3-SS	0x26	P38	ISO/ABA start sentinel	"+" (0x2B)
Tk3-SS AAMVA	0x27	P39	AAMVA start sentinel	"#" (0x23)
Tk2-SS 7 bit	0x28	P40	7 bit start sentinel (ISO/ABA Track 1 start sentinel)	"@" (0x40)
Tk3-SS 7 bit	0x29	P41	7 bit start sentinel (ISO/ABA Track 1 start sentinel)	"&" (0x26)
ES	0x2B	P43	End Sentinel	"?" (0x3F)
	0x2D	P45	Track 1 Specific End Sentinel	"?" (0x3F)
	0x2E	P46	Track 2 Specific End Sentinel	"?" (0x3F)
	0x2F	P47	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.

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 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. The characters that are masked depend 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 entirely masked or sent totally in the 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 C** for a description of how ISO/ABA and AAMVA cards are identified.)

Each of these properties allows the application to specify masking details for the Primary Account Number and Driver’s License / ID Number (DL/ID#), the masking character to be used, and whether or not to apply an adjustment to force the Mod 10 9 (Luhn algorithm) digit at the end of the number to be correct.

Track 1 Masked Data

This Simple ASCII field contains the Masked Track Data for track 1. The device transmits all characters.

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 Simple ASCII 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 Simple ASCII 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.

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, always set to One
Bit 2	=	Encryption Enabled, always set to One
Bit 3	=	Always set to Zero
Bit 4	=	Timed Out waiting for user to swipe card
Bit 5	=	Always set to Zero
Bit 6	=	Always set to Zero
Bit 7	=	Always set to Zero
Bit 8	=	Encryption Counter Expired
Bits 9–15	=	Unassigned (always set to Zero)

Notes:

- (1) Encryption will only be performed when Encryption Enabled (bit 2) and Initial DUKPT key Injected (bit 1) 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 (bit 0) 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]

Encrypted Track Data

If decodable track data exists for a given track, both the Masked Track Data field and the Encrypted Track Data field for that track will contain data.

The encrypted data from each track is decoded and converted to ASCII, then 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 A for more information.

Track 1 Encrypted Data

This Binary field contains the encrypted track data for track 1.

Track 2 Encrypted Data

This Binary field contains the encrypted track data for track 2.

Track 3 Encrypted Data

This Binary field contains the encrypted track data for track 3.

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	1	0	1	0	0	0	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.

Encrypted MagnePrint Data

This 56 byte Binary field contains the MagnePrint data. After decryption, the final two bytes should be discarded leaving the 54 byte MagnePrint data. 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 Simple 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.

Encrypted Session ID

This eight 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 value 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 0xFF.

Encryption Counter

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

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. If the property is True, the CRC is sent, if it is False, the CRC is not sent. The default state for this property is True.

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.

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 "0001". 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. 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.

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.

Command requests and responses are sent to and received from the reader using command strings. Command requests are sent to the reader via a serial port. The response to a command is retrieved from the corresponding serial port.

Each command and response is composed of a series of readable ASCII characters followed by the ASCII character CR. The ASCII characters preceding the CR are the message; there should always be an even number of them and they should contain only the characters 0123456789ABCDEF. The receiver will combine two successive ASCII characters from the message to form one “byte” (see the descriptions of the commands) which may have any value from 0x00 to 0xFF.

The following table shows the structure of a command message:

Byte	Usage
0	Command Number
1	Data Length
2 – 23	Data

The following table shows the structure of a response to a command.

Byte	Usage
0	Result Code
1	Data Length
2 – 23	Data

PRIVILEGED COMMANDS

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

- (1) Set Property
- (2) Reset Device*

* The Reset Device command is usually not Privileged. The exception occurs when sending a sequence to Activate the Authenticated Mode, during which time the Reset Device command is Privileged to prevent a hacker from using this sequence to exhaust DUKPT keys, thereby rendering the reader unusable.

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.

Value (Hex)	Command Number	Description
00	Get Property	Gets a property from the reader
01	Set Property	Sets a property in the reader
02	Reset Device	Resets the reader
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	Completes the Activation of Authenticated Mode of secure operation
12	Deactivate Authenticated Mode	Deactivates the Authenticated Mode of secure operation
14	Get Reader State	Gets the current state of the reader
1C	Get Encryption Counter	Gets the encryption counter
30	Bulk Encrypt	Encrypts Bulk Data
A0	Read ASIC Control	Enables / disables the Read ASIC

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 maximum length of this field is fixed at 120 bytes. Valid data should be placed in the field starting at offset 2.

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. 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

Result Codes

The result codes for the **Get** and **Set Property** commands can be any of the codes listed in the generic result code table.

Property ID

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

Value (Hex)	P-Value	Property	Description
00		Software ID	The reader's software identifier
03		Device Serial Num	The reader's serial number
04		MagneSafe Version Number	Version number of MagneSafe feature set
05		Track ID Enable	Track enable / ID enable
07		ISO Track Mask	Specifies Masking factors for ISO cards
08		AAMVA Track Mask	Specifies Masking factors for AAMVA cards
10		Interface Type	Type of interface
14		Track, Data Send Flags	Track data send flags
15		MP Flags	Enables sending of MagnePrint data
19		CRC Flag	Enables/disables sending CRC
1B		Decode Enable	Enables decoding for certain formats
1C		SS JIS TYPE 2	Start sentinel character for JIS type 2
1D		ES JIS TYPE 2	End Sentinel Character for JIS type 2
1E	P30	Pre-Card String	Pre card string
1F	P31	Post-Card String	Post card string
20	P32	Pre-Track String	Pre track string
21	P33	Post-Track String	Post track string
22	P34	Termination String	Terminating string
23	P35	FS	Field Separator for additional data
24	P36	SS Track 1 ISO ABA	Start sentinel char for track 1 – ISO/ABA
25	P37	SS Track 2 ISO ABA	Start sentinel char for track 2 – ISO/ABA
26	P38	SS Track 3 ISO ABA	Start sentinel char for track 3 – ISO/ABA
27	P39	SS Track 3 AAMVA	Start sentinel char for track 3 – AAMVA
28	P40	SS Track 2 7BITS	Start sentinel char for track 2 – 7 bit data
29	P41	SS Track 3 7BITS	Start sentinel char for track 3 – 7 bit data
2B	P43	ES	End sentinel char for all tracks/formats except JIS type 2
2C		Format Code	Defines the Format Code to be sent with the message
2D	P45	ES Track 1	End sentinel char for track 1
2E	P46	ES Track 2	End sentinel char for track 2 except JIS type 2
2F	P47	ES Track 3	End sentinel char for track 3
30		Send Encryption Counter	Enables/disables sending Encryption Counter
31		Mask Other Cards	Enables/disables masking of cards that don't meet the ISO Financial or the AAMVA formats
34		Send clear AAMVA card data flag	Enables/disables sending of clear AAMVA card data

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.

Property Default Values

Each property specifies a default value. This is the firmware default value and may be changed during the manufacturing or order fulfillment process to support the needs of specific clients.

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 “21042875A01”.

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	0B	32 31 30 34 32 38 37 35 41 30 31

DEVICE SERIAL NUMBER 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. 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 for these changes to take effect.

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

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

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

Result Code	Data Len	Data
00	00	

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

Cmd Num	Data Len	Prp ID
00	01	03

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

Result Code	Data Len	Prp Value
00	03	31 32 33

MAGNESAFE VERSION NUMBER PROPERTY

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 maximum 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 ₃	T ₃	T ₂	T ₂	T ₁	T ₁
----	---	----------------	----------------	----------------	----------------	----------------	----------------

Id 0 – Decodes standard ISO/ABA cards only

1 – Decodes AAMV 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_# 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 for these changes to take effect.

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 for these changes to 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 DL/ID# 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 DL/ID# will be '0'
 - All data after the DL/ID# 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 for these changes to take effect.

INTERFACE TYPE PROPERTY

- Property ID: 0x10
 Property Type: Byte
 Length: 1 byte
 Get Property: Yes
 Set Property: No
 Default Value: 2 (Indicates UART interface)
 Description: The value is a 1-byte **read-only** property that represents the reader's interface type. It is always set to 2 indicating this is an UART/RS232 reader.

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

Cmd Num	Data Len	Prp ID
00	01	10

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

Result Code	Data Len	Prp Value
00	01	02

TRACK DATA SEND FLAGS PROPERTY

- 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	0	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

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

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 for these changes to take effect.

MP FLAGS PROPERTY

Property ID: 0x15

Property Type: Byte

Length: 1 byte

Get Property: Yes

Set Property: Yes

Default Value: 0x00

Description: This property is defined as follows:

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 card swipe message. Setting S to 1 causes the MagnePrint Status and Unencrypted MagnePrint Data to be sent with each swipe. Setting S to 0 causes these fields to be omitted from the data. When these 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 for these changes to take effect.

CRC FLAG PROPERTY

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 Encrypted and/or the Clear Text CRC will be sent as part of a card swipe message. In the default state of this property, the device will send only the Clear Text CRC. When these 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 for these changes to take effect.

DECODE ENABLE PROPERTY

Property ID: 0x1B
 Property Type: Byte
 Length: 1 byte
 Get Property: Yes
 Set Property: Yes
 Default Value: 0x00
 Description: This property is defined as follows:

Bit Position	7	6	5	4	3	2	1	0
Decode Type	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	JIS Type 2

When a decode type bit is set to 1 (true), the decode type represented by that bit is enabled. When a decode type bit is set to 0 (false), the decode type represented by that bit is disabled. The reserved decode type bits should always be set to zero.

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 for these changes to take effect.

Example Set property Request (Hex):

Cmd Num	Data Len	Prp ID	Prp Value
01	02	1B	01 (enable JIS Type 2 decode type)

Example Set property Response (Hex):

Result Code	Data Len	Data
00	00	

Example Get property Request (Hex):

Cmd Num	Data Len	Prp ID
00	01	1B

Example Get property Response (Hex):

Result Code	Data Len	Prp Value
00	01	01

SS JIS TYPE 2 PROPERTY

Property ID: 0x1C
Property Type: Byte
Length: 1 byte
Get Property: Yes
Set Property: Yes
Default Value: 0x7F 'DEL'
Description: This character is sent as the start sentinel for cards that are encoded in the JIS type 2 format. If the value is in the range 0 – 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 for these changes to take effect.

ES JIS TYPE 2 PROPERTY

Property ID: 0x1D
Property Type: Byte
Length: 1 byte
Get Property: Yes
Set Property: Yes
Default Value: 0x7F 'DEL'
Description: This character is sent as the end sentinel for cards that are encoded in the JIS type 2 format. If the value is in the range 0 – 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 for these changes to take effect.

PRE CARD STRING PROPERTY

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 for these changes to 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

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 for these changes to 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

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 for these changes to 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

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 for these changes to 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

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 for these changes to take effect.

FS PROPERTY

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, Device 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 for these changes to take effect.

SS TRACK 1 ISO ABA PROPERTY

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 for these changes to take effect.

SS TRACK 2 ISO ABA PROPERTY

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 for these changes to take effect.

SS TRACK 3 ISO ABA PROPERTY

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 for these changes to take effect.

SS TRACK 3 AAMVA PROPERTY

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 for these changes to take effect.

SS TRACK 2 7BITS PROPERTY

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 for these changes to take effect.

SS TRACK 3 7BITS PROPERTY

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 for these changes to take effect.

ES PROPERTY

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 except JIS type 2. 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 for these changes to take effect.

FORMAT CODE PROPERTY

Property ID: 0x2C
Property Type: String
Length: 4 bytes
Get Property: Yes
Set Property: Yes
Default Value: "0000"
Description: 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' means the Format Code is application 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 for these changes to 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 for these changes to 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 except JIS type 2. 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 for these changes to 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 for these changes to take effect.

SEND ENCRYPTION COUNTER PROPERTY

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, neither the Encryption Counter nor the field separator will be sent. If the property is set to 0x01, the Encryption Counter is sent as the field immediately following the DUKPT Serial Number in a swipe message.

NOTE: If this property is set to 0x01 and the Format Code is currently "0000", the Format Code will be changed to "0001".

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 for these changes to take effect.

MASK OTHER CARDS PROPERTY

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) 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 for these changes to take effect.

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 property is used to control how to send out AAMVA 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 for these changes to 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

RESET DEVICE COMMAND

Command number: 0x02

Description: This command resets the reader. It can be used to make previously changed properties take effect without power cycling the reader.

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

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

Cmd Num	Data Len	Data
02	00	

Example **Reset Device** 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

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 Parameter – The Request Data is not the correct length. The request command contains no data, so the Data Length must be 0.

Example Request (Hex):

Cmd Num	Data Len	Data
09	00	none

Example 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, and 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 application wishes.

Response Data: None

Result codes: 0x00 Success

0x02 Bad Parameter – The Request Data is not the correct length. The Session ID is an 8-byte field, so the Data Length must be 8.

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. Note that this command provides the only means by which to enter this Mode.

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 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 application 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: None

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	8579 8275 2157 3495

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 got 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.

Current State Antecedent		
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 Request (Hex):

Cmd Num	Data Len	Data
14	00	

Example Response (Hex):

Result Code	Data Len	Data
00	02	00 00

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

ENCRYPT BULK DATA COMMAND

Command number: 0x30

Description: This command will encrypt up to a maximum of 120 bytes. The Data-Response variant of the DUKPT key will be used to encrypt data. It will also compute a MAC for the S/N, Num Bytes Encrypted, KSN and Cryptogram. Data to be encrypted that are not a multiple of 8 bytes will be padded with NULLs to be a multiple of 8.

The DUKPT key counter/pointer will be incremented before processing this command.

Example Request (**Encrypt Bulk Data**) (Hex):

Cmd Num	Data Len	Data
30	05	01 02 03 04 05

Example **Encrypt Bulk Data Response** (Hex):

Result Code	Data Len	DSN (16 bytes)	Num Bytes Encrypted (1 byte)	KSN (10 bytes)	Cryptogram (8 bytes)	MAC (4 bytes)
00	0x27	32 31 30 34 32 38 31 32 44 30 31 31 31 31 31 00	05	31 32 33 34 35 31 32 33 34 35	01 02 03 04 05 06 07 08	01 02 03 04

DSN – Device Serial Number, this data field will always be fixed at 16 bytes. If the serial number is less than 15 bytes, it will be left justified. The 16th byte will always be set to NULL.

Cryptogram – Encrypted data, the length of which is always a multiple of 8, this field can be maximum of 120 characters.

Result codes: 0x00 Success
 0x02 Bad Parameters, the Data Len is not supported
 0x07 Security Level < 2, MSC1 CMUT was incorrect

READ ASIC CONTROL COMMAND

Command number: 0xA0

Description: This command is used to enable or disable the Read ASIC. Setting S to 0 causes the read ASIC to be in the disabled and low power state. In this state, the ASIC will not read cards. The S bit must be set to 1 to enable cards to be read.

Data structure:

Request Data:

Offset	Field Name	Description
0	ASIC Control	This 1-byte field controls the state of the Read ASIC:

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

S 0 – Read ASIC disabled.
 1 – Read ASIC enabled.

Response Data: None

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

Example **Read ASIC Control** Request (Hex), Enable ASIC:

Cmd Num	Data Len	Data
A0	01	01

Example **Read ASIC Control** Response (Hex):

Result Code	Data Len	Data
00	00	

APPENDIX A. 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 B. COMMAND EXAMPLE

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

Example 1: Swipe decryption, iDynamo MagneSafe V5 Reader:

This example shows the data received in a Card 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 Card Swipe Data:

Byte	Content
0	%B5452000000007189^HOGAN/PAUL ^08040000000000
50	000000000?;5452000000007189=080400000000000000?+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 iDynamo MagneSafe V5 Communications Reference Manual, the Card Swipe 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 TK1 SS and ES)]
[P35] [Tk3 Encrypted Data (including TK1 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 TK1 SS and ES)]
|[Tk3 Encrypted Data (including TK1 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      ^080400000000000000000000?
;5452000000007189=080400000000000000?
+5163000050000445=00000000000000?
|0600
|C25C1D1197D31CAA87285D59A892047426D9182EC11353C051ADD6D0F072A6CB3436560B3071FC1FD11D9F7
E74886742D9BEE0CFD1EA1064C213BB55278B2F12
|724C5DB7D6F901C7F0FEAE7908801093B3DBFE51CCF6D483E789D7D2C007D539499BAADCC8D16CA2
|E31234A91059A0FBFE627954EE21868AEE3979540B67FCC40F61CECA54152D1E
|A1050000
|8628E664C59BBAA232BA90BFB3E6B41D6F4B691E633C311CBE6EE7466B81196EC07B12648DCAC4FD7FD0E21
2B479C60BAD8C74F82F327667
|
|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: 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 that the KSN = FFFF9876543210E00008 and the counter = 8.
For the standard ANSI key example, counter 8 gets us the following Encryption Key:
27F66D5244FF621E AA6F6120EDEB427F

There are five encrypted fields:

1. Track 1 encrypted data
2. Track 2 encrypted data
3. Track 3 encrypted data
4. Encrypted MagnePrint data
5. 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 A 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 A 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 A tells us to decrypt the last block:

```
0F61CECA54152D1E TDES Dec with 27F66D5244FF621E AA6F6120EDEB427F
gets DE0949643B57C3C4
XOR EE3979540B67FCC4
gets 3030303030303F00 (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
3030303030303F00	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 A 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 (application 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 application and thus was all zeroes.

APPENDIX C. IDENTIFYING ISO/ABA AND AAMVA CARDS

ISO/ABA FINANCIAL CARDS

1. If a 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 track may 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 terminated by the character '^' (Field Separator).
 - f. A correctly structured Name has a maximum of 26 characters terminated 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 the Expiration Date is incorrectly structured, the rest of the track (from the point of discrepancy) will be sent in the clear.
 - c. If the PAN or the Expiration Date contains the '?' character (End Sentinel), the field is not correctly structured.
 - d. A correctly structured PAN has a maximum of 19 digits and is terminated 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 the reading of Track 3, so this rule will not apply to such readers.
2. If a low-level decoding algorithm finds data for the 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 is “604425” or contains values in the range “636000” to “636062” inclusive, then 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 contains the ‘?’ character (End Sentinel), the field is not correctly structured.
 - A correctly structured DLID has a maximum of 19 digits and is terminated by the character ‘=’ (Field Separator).
 - A correctly structured Expiration Date has 4 characters.
 - A correctly structured Birth Date has 8 characters.