iDynamo 5 (Gen II)

Secure Card Reader Authenticator
Programmer's Manual (COMMANDS)

June 2021

Document Number:
D998200309-20

REGISTERED TO ISO 9001:2015

Table 0-1 - Revisions

Rev Number	Date	Notes
17	May 10, 2019	Initial Release from master programmer's manual Rev 17
20	Jun 8, 2021	Rebuild from master programmer's manual Rev 20 Add Contactless Quick Chip feature and supporting detail, split EMV Quick Chip feature into Contact vs. Contactless; Add OEM Features feature and supporting information; Update **Table 1-2** and add iDynamo 5 (Gen II), iDynamo 6 and Power Management scheme PM7; Add QuickPass Support feature and supporting information; Remove Dynasty; Move terminal country codes and terminal language codes into tag tables instead of a dedicated appendix; Throughout, clarify Currency Codes are driven by ISO standard; Fix sentinels in **Table 3-1**; **1.5** add features Pairing Mode Control, Apple VAS, Conserve DUKPT Keys, No EMV MSR Flow; **7.2** add Result Code 0x55 (Embedded V5 Head Only); Misc. clarifications and corrections.

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 MagTek Support Services 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 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 INFORMATION

This device complies with Part 15 of the FCC Rules. Operation 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.

Note: This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient or relocate the receiving antenna.
- Increase the separation between the equipment and receiver.
- Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.
- Consult the dealer or an experienced radio/TV technician for help.

Caution: Any changes or modifications not expressly approved by the party responsible for compliance could void the user's authority to operate this equipment.

CUR/UR

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

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 conformé 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 ***UL 60950-1, 2nd Edition, 2011-12-19*** (Information Technology Equipment - Safety - Part 1: General Requirements), ***CSA C22.2 No. 60950-1-07, 2nd Edition, 2011-12*** (Information Technology Equipment - Safety - Part 1: General Requirements).

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 (Pb).

SOFTWARE LICENSE AGREEMENT

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

TERMS, CONDITIONS, AND RESTRICTIONS

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

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

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

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

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

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

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

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

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

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

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

QUESTIONS REGARDING THIS AGREEMENT SHOULD BE ADDRESSED IN WRITING TO MAGTEK, INCORPORATED, ATTENTION: CUSTOMER SUPPORT, AT THE ADDRESS LISTED IN THIS DOCUMENT, OR E-MAILED TO SUPPORT@MAGTEK.COM.

Table of Contents

1 Introduction

1.1 About This Document

This document describes how to communicate with Secure Card Reader Authenticator (SCRA) devices which implement MagneSafe V5.

1.2 About SDKs

MagTek provides convenient SDKs and corresponding documentation for many programming languages and operating systems. The API libraries included in the SDKs wrap the details of the connection in an interface that conceptually parallels the device's internal operation, freeing software developers to focus on the business logic, without having to deal with the complexities of platform APIs for connecting to the various available connection types, communicating using the various available protocols, and parsing the various available data formats. Information about using MagTek wrapper APIs is available in separate documentation, including ***D99875535 Secure Card Reader Authenticator API PROGRAMMING REFERENCE MANUAL***.

The SDKs and corresponding documentation include:

- Functions for sending the direct commands described in this manual
- Wrappers for commonly used commands that further simplify development
- Sample source code to demonstrate how to communicate with the device using the direct commands described in this manual

To download the SDKs and documentation, search www.magtek.com for "SDK" and select the SDK and documentation for the programming languages and platforms you need, or contact MagTek Support Services for assistance.

Software developers also have the option to revert to direct communication with the device using libraries available in the chosen development framework. For example, custom software written in Visual Basic or visual C++ may make API calls to the standard Windows USB HID driver. This document provides information and support for developing host software using that method.

MagTek has also developed software that demonstrates direct communication with the device, which software developers can use to test the device and to which provides a starting point for developing other software. For more information, see the MagTek web site, or contact your reseller or MagTek Support Services.

1.3 About Terminology

The general terms “device” and “host” are used in different, often incompatible ways in a multitude of specifications and contexts. For example, “host” may have different a meaning in the context of USB communication than in the context of networked financial transaction processing. In this document, “device” and “host” are used strictly as follows:

- **Device** refers to the Secure Card Reader Authenticator (SCRA) that receives and responds to the command set specified in this document. Devices include Dynamag, eDynamo, and so on.
- **Host** refers to the piece of general-purpose electronic equipment the device is connected or paired to, which can send data to and receive data from the device. Host types include PC and Mac computers/laptops, tablets, smartphones, teletype terminals, and even test harnesses. In many cases the host may have custom software installed on it that communicates with the device. When “host” must be used differently, it is qualified as something specific, such as “acquirer host” or “USB host.”

Similarly, the word “user” is used in different ways in different contexts. This document separates users into more descriptive categories:

- The **cardholder**
- The **operator** (such as a cashier, bank teller, customer service representative, or server), and
- The **developer** or the **administrator** (such as an integrator configuring the device for the first time).

Because some connection types, payment brands, and other vocabulary name spaces (notably Bluetooth LE, EMV, smart phones, and more recent versions of Windows) use very specific meanings for the term “Application,” this document favors the term **software** to refer to software on the host that provides a user interface for the operator.

The combination of device(s), host(s), software, firmware, configuration settings, physical mounting and environment, user experience, and documentation is referred to as the **solution**.

1.4 About Connections and Data Formats

MagneSafe V5 products transmit data using a set of common data formats across a variety of physical connection layers, which can include universal serial bus (USB) acting as a keyboard (“USB KB”), USB acting as a vendor-defined HID device (“USB HID”), RS-232, Apple iAP (Lightning or USB), bidirectional audio connectors, Bluetooth, Bluetooth LE, and so on. The set of available physical connection types and the data formats available on each connection type is device-dependent. **Table 1-1** shows the physical connection types available on each product, and the data formats supported on each connection type for that device. Details about connection types and formats can be found in section **2 Connection Types** and section **3 Data Formats**. Section headings in this document include tags that indicate which connection types and/or data formats they apply to.

Table 1-1 - Device Connection Types / Data Formats

Product / Connection	Audio	Bluetooth LE GATT	Bluetooth LE GATT KB	Bluetooth	iAP1 Lightning	iAP2 Lightning	iAP2 USB	RS-232 / UART	SPI	USB HID	USB KB
BulleT KB				Streaming (MSR data)						HID	
BulleT SPP				Streaming							
cDynamo					Streaming						
Dynamag, Dynamag Duo, USB Enc IntelliHead V5										HID	Streaming
DynaMAX		GATT	Streaming							HID	
DynaPAD										HID	Streaming
DynaWave								SLIP		HID	
eDynamo		GATT								HID	
iDynamo 5					Streaming						
iDynamo 5 (Gen II)						Streaming					
iDynamo 6						SLIP	SLIP			HID	
kDynamo						SLIP					

Product / Connection	Audio	Bluetooth LE GATT	Bluetooth LE GATT KB	Bluetooth	iAP1 Lightning	iAP2 Lightning	iAP2 USB	RS-232 / UART	SPI	USB HID	USB KB
mDynamo										HID	
P-series and I-65 w/V5										HID	Streaming
pDynamo		GATT								HID	
sDynamo						Streaming					
SPI Enc IntelliHead V5									Streaming		
tDynamo		GATT								HID	
UART Enc IntelliHead V5								Streaming			
uDynamo	TLV									HID	

1.5 About Device Features

The information in this document applies to multiple devices. When developing solutions that use a specific device or set of devices, integrators must be aware of each device's connection types, data formats, features, and configuration options, which affect the availability and behavior of some commands. **Table 1-2** provides a list of device features that may impact command availability and behavior. All section headings in this document include tags that indicate which features they apply to.

Table 1-2 - Device Features

Feature / Product	BulleT KB BulleT SPP	cDynamo	Dynamag, USB Enc IntelliHead V5	Dynamag Duo	DynaMAX	DynaPAD	DynaWave	eDynamo	iDynamo 5	iDynamo 5 (Gen II)	iDynamo 6	kDynamo	mDynamo	P-series, I-65 w/V5	pDynamo	sDynamo	SPI Encrypting IntelliHead V5	tDynamo	UART Enc IntelliHead V5	uDynamo
MSR Swipe	Y	Y	Y	Y	Y	Y	N	Y	Y	Y	Y	Y	N	N	Y	Y	Y	Y	Y	Y
MSR Insert	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N	N	N	N	N	N
MSR 3 Tracks	Y	Y	Y	Y	Y	N	N	Y	Y	Y	Y	Y	N	Y	Y	Y	Y	Y	Y	
MSR Disable	N	Y	N	N	N	N	N	N	Y	N	N	N	N	N	N	N	N	N	N	N
MSR Swap Tracks 1/3	N	N	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
MSR Embedded V5 Head	N	N	N	N	N	N	N	N	N	Y	Y	Y	N	N	N	Y	N	Y	N	N
MSR Configurabe MSR Variants		Y	Y	Y	Y		N	Y	Y	Y	Y	Y	Y		Y	Y		Y		
MSR Configurable MP Variants		N	N	N	Y		N	Y	N	N	Y	Y	Y		Y	N		Y		
MSR SureSwipe		N	Y	Y	Y	Y	N	Y	N	N	N	N	N	Y	N	N	N	N	N	N
MSR JIS Capable		Y	Y[3]	N	N	N	N	N	Y	N	N	N	N	N	N	Y	Y	N	Y	
MSR SHA-1		N	Y	Y	Y	Y	N	Y	N	N	N	N	N		Y	N		N		
MSR SHA-256		N	N	N	N	N	N	N	N	N	N	N	N	N	N	N		N		
MSR Configurable SHA		N	N	N	Y		N	Y	N	N	N	N	N			N		N		
MSR MagneSafe 2.0							N	Y		N	N	N			N			N		
Configurable Encryption Algorithm	N	N	N	N	N	N	N	N	N	N	Y	N	N	N	N	N		N	N	N
Set Mask Service Code	N	N	Y[2]	N	N	N	Y	N	N	N	N	N	N	Y[2]	N	N	Y[2]	N	N	N
Never Mask Service Code			N[2]				N	Y	Y	Y	Y	Y	Y	N[2]		Y	N[2]	Y		

Feature / Product	BulleT KB BulleT SPP	cDynamo	Dynamag, USB Enc IntelliHead V5	Dynamag Duo	DynaMAX	DynaPAD	DynaWave	eDynamo	iDynamo 5	iDynamo 5 (Gen II)	iDynamo 6	kDynamo	mDynamo	P-series, I-65 w/V5	pDynamo	sDynamo	SPI Encrypting IntelliHead V5	tDynamo	UART Enc IntelliHead V5	uDynamo
EMV Contact	N	N	N	N	N	N	N	Y	N	N	Y	Y	Y	N	N	N	N	Y	N	N
EMV Contactless	N	N	N	N	N	N	Y	N	N	N	Y	Y	N	N	N	N	N	Y	N	N
EMV ODA	N	N	N	N	N	N	Y	Y	N	N	N	Y	Y	N	N	N	N	Y	N	N
EMV MSR Flow	N	N	N	N	N	N	N	N	N	N	Y	Y	N	N	N	N	N	Y	N	N
No EMV MSR Flow	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N	Y	Y	Y	Y	Y	N	Y	Y
EMV Contact Quick Chip	N	N	N	N	N	N	N	Y[4]	N	N	Y	Y	Y[4]	N	N		N	Y	N	N
EMV Contactless Quick Chip	N	N	N	N	N	N	Y	N	N	N	Y	Y	N	N	N	N	N	Y	N	N
Comprehensive Checksums	N	N	N	N	N	N	Y	N	N	N	N	N	N	N	N	N	N	N	N	N
Conserve DUKPT Keys	N	N	N	N	N	N	N	Y[7]	N	N	N	N	Y[7]	N	N	N	N	N	N	N
QuickPass Support	N	N	N	N	N	N	Y	N	N	N	N	N	N	N	N	N	N	N	N	N
Apple VAS	N	N	N	N	N	N	Y	N	N	N	N	N	N	N	N	N	N	N	N	N
Application Selection Options	N	N	N	N	N	N	Y	N	N	N	Y	Y	N	N	N	N	N	Y	N	N
External PIN Accessory Support	N	N	N	N	N	N	Y	N	N	N	Y	N	N	N	N	N	N	N	N	N
Keypad Entry	N	N	N	N	N	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N
Fixed Key	N	N	N	N	N	N	N	N	N	N	N	N	Y	N	N	N	N	N	N	N
MSR Secondary DUKPT Key	N	N	N	N	Y	N	Y	Y	N	N	N	N	Y	N	Y	N	N	N	N	Y
Power Mgt Scheme (PM#)	1	N	N	N	2	N	N	3	N	N	7	5	N	N	6	N	N	5	N	4
Battery-Backed RTC							N	Y		N	N	N	N					N		
OEM Features	N	N	N	N	N	N	Y	N	N	N	N	N	N	N	N	N	N	N	N	N
Transaction Validation	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N	N	N	N	N
Display	N	N	N	N	N	Y*	N	N	N	N	N	N	N	N	Y	N	N	N	N	N
Multi-Language	N	N	N	N	N	N	Y	N	N	N	Y	Y	N	N	N	N	N	Y	N	N
Tamper	N	N	N	N	N	N	N	Y	N	N	N	N	N	N	N	N	N	N	N	N

Feature / Product	BulleT KB BulleT SPP	cDynamo	Dynamag, USB Enc IntelliHead V5	Dynamag Duo	DynaMAX	DynaPAD	DynaWave	eDynamo	iDynamo 5	iDynamo 5 (Gen II)	iDynamo 6	kDynamo	mDynamo	P-series, I-65 w/V5	pDynamo	sDynamo	SPI Encrypting IntelliHead V5	tDynamo	UART Enc IntelliHead V5	uDynamo
Extended Commands	N	N	N	N	N	N	Y	Y	N	N	Y	Y	Y	N	N	N	N	Y	N	N
Extended Notifications	N	N	N	N	N	N	Y	Y	N	N	Y	Y	Y	N	N	N	N	Y	N	N
Dual USB Ports	N	N	N	N	N	N	N	N	N	N	Y	N	N	N	N	N	N	Y	N	N
Pairing Modes	N	N	N	N	N	N	N	Y[5]	N	N	N	N	N	N	Y	N	N	Y	N	N
Pairing Mode Control	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N	N
Custom Advertising	N	N	N	N	N	N	N	Y[6]	N	N	N	N	N	N	Y	N	N	Y	N	N
Configurable iAP FID	N	Y	N	N	N	N	N	N	N	Y	Y	Y	N	N	N	N	N	N	N	N
Auxiliary Ports	N	N	N	N	N	N	N	N	N	N	N	N	Y	N	N	N	N	N	N	N
Configurable Baud Rate	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N
Configurable Pushbutton	N	N	N	N	N	N	N	N	N	N	N	Y	N	N	N	N	N	Y	N	N
External LED Control	N	N	N	N	N	N	N	N	N	N	N	N	Y[1]	N	N	N	N	N	N	N
Encrypt Bulk Data (b)	120	120	24	24	24	N	N	24	120	N	N	N	24	N	N	N	120	N	12	24

Feature / Product	BulleT KB BulleT SPP	cDynamo	Dynamag, USB Enc IntelliHead V5	Dynamag Duo	DynaMAX	DynaPAD	DynaWave	eDynamo	iDynamo 5	iDynamo 5 (Gen II)	iDynamo 6	kDynamo	mDynamo	P-series, I-65 w/V5	pDynamo	sDynamo	SPI Encrypting IntelliHead V5	tDynamo	UART Enc IntelliHead V5	uDynamo

1) This feature is available in mDynamo firmware revision 1000003358D00 (released August 2017) and newer.
2) This feature was introduced in SPI Encrypting IntelliHead V5 in firmware version 21042876C01 released July 2017, P-series and I-65 w/V5 in firmware version 21165822E01 released March 2018, Dynamag and USB Encrypting IntelliHead V5 in firmware version 21042840K00 released January 2019.
3) This feature is available in Dynamag and USB Enc IntelliHead V5 firmware version 21042840K00 (released January 2019) and newer.
4) EMV Contact Quick Chip is available in mDynamo firmware revision 1000003358F01 (released December 2017), eDynamo firmware revision 1000003354F00 (released October 2018), and newer.
5) Pairing Modes feature is available in eDynamo firmware revision 1000002650B01 and newer, except Bluetooth LE Property 0x13 which was added in 1000002650C01.
6) Custom Advertising feature is available in eDynamo firmware revision 1000002650C02 and newer, except Bluetooth LE Property 0x08 Configuration Bits “Never Advertise” and “USB Power Not Exit Airplane Mode” which were added in 1000002650C01.
7) This feature is only supported by the eDynamo starting with firmware revision ***1000003354J00*** and mDynamo starting with firmware revision ***1000003358G01***.

2 Connection Types

Table 1-1 on page **14** includes a list of connection types available for each device. The following subsections provide details developers will need to communicate with the device using each connection type.

2.1 How to Use Apple iAP Connections (iAP Only)

This section provides information about developing an iOS app that interfaces with the device via the Lightning or USB connector using iPod Accessory Protocol (iAP). For sample code and other supporting materials, see ***99510111 DYNAMAX / EDYNAMO / UDYNAMO / ADYNAMO / IDYNAMO / KDYNAMO / SDYNAMO / TDYNAMO SDK FOR IOS (WEB)***, available from MagTek.

To develop host software for an iOS host that connects to the device, you must know the following device properties, which are specified by the purchaser when ordering, and loaded by the manufacturer:

- ***BundleSeedIDString***, which is a 10-character string assigned by Apple, Inc. to the host software developer
- ***protocolString***, also known as the SDK Protocol, usually in the form of a reverse DNS string unique to the host software developer or the device purchaser.

The host software project must include the protocolString in its ***.plist*** file before compiling. Spelling, including punctuation and capitalization, must exactly match the protocolString of the device.

The host software should initiate a connection to the device using the iOS SDK's ***ExternalAccessory*** Framework (for sample code, see Apple's ***EADemo*** app). Upon establishing the connection, the host can begin exchanging data with the device. Devices may use different formats to send and receive different types of data on different connections, or may change their behavior based on configuration. To determine the data format to use, look up the device and connection type in **Table 1-1**. For details about using **Streaming** format, see section **3.1 How to Use Streaming Format (Streaming Only)**.

On some devices, code upgrade commands are not available through this connection.

3 Data Formats

3.1 How to Use Streaming Format (Streaming Only)

This section describes how the device functions when it is using Streaming format on its current connection to the host. Some device connection types use streaming format for both commands/responses and for magnetic stripe data, while other device connection types use streaming format just for magnetic stripe data. The following sections describe each of these separately.

3.1.1 Magnetic Stripe Card Data In Streaming Format (Swipe Only | Keypad Entry Only)

In streaming format, the device sends **Magnetic Stripe Card Data Sent from Device to Host (MSR Only | Keypad Entry Only)** as a series of potentially variable length fields in a fixed order, separated by delimiters. Many of the delimiters are configurable, which allows the device to output customized sequences of characters to the host.

Streaming data is composed entirely of ASCII characters, but the host should interpret the characters differently depending on the nature of the data. The tables in section **6** that describe where to find these values in the data stream also describe how to decode them:

- **ASCII** fields like **Masked Track Data**, **Device Serial Number**, and **Format Code (Streaming Only)** simply contain the data as ASCII characters.
- **Binary** fields like **Device Encryption Status**, **Encrypted Track Data**, **MagnePrint Status**, **Encrypted MagnePrint Data**, **Encrypted Session ID**, **DUKPT Key Serial Number**, **Clear Text CRC (Streaming Only)**, and **Encrypted CRC (Streaming Only)** are hexadecimal encoded, where the contents consist only of the characters 0123456789ABCDEF, and every two bytes represents the hexadecimal value of the binary byte being sent. The host should decode every two characters as one byte.

The delimiters the device sends between fields are stored as **Properties** in the device's non-volatile memory, which the host can configure using **Command 0x01 - Set Property (MAC)**. **Table 3-1** shows the format the device uses to transmit magnetic stripe data in Streaming mode, where the delimiter properties are abbreviated as a "P" followed by the property number. When transmitting card data to the host, the device replaces each bracketed [P0x##] value with the actual value contained in the specified property, and replaces other bracketed values with card data. For information about a specific property's valid values and effects on device behavior, see its documentation in section **8 Properties**.

Table 3-1 - Card Data Format (Streaming Mode)

Card Data Format
[P0x1E] [P0x20] [P0x24] [**Track 1 Masked Data**] [P0x2B or P0x2D] [P0x21] [P0x20] [P0x1C or P0x25 or P0x28] [**Track 2 Masked Data**] [P0x1D or P0x2B or P0x2E] [P0x21] [P0x20] (Only if exists: [P0x26 or P0x27 or P0x29] [**Track 3 Masked Data**] [P0x2B or P0x2F] [P0x21]) [P0x1F] [P0x23] [**Device Encryption Status**] [P0x23] (Encrypted together: [P0x24] [**Track 1 Encrypted Data**] [P0x2B or P0x2D]) [P0x23] (Encrypted together: [P0x1C or P0x25or P0x28] **Track 2 Encrypted Data**] [P0x1D or P0x2B or P0x2E]) [P0x23] (Encrypted together: [P0x26 or P0x27 or P0x29] [**Track 3 Encrypted Data**] [P0x2B or P0x2F]) [P0x23] [**MagnePrint Status**] [P0x23] [**Encrypted MagnePrint Data**] [P0x23] [**Device Serial Number**] [P0x23] [**Encrypted Session ID**] [P0x23] [**DUKPT Key Serial Number**] [P0x23] [**Remaining MSR Transactions**] (optional, off by default) [P0x23] [**Clear Text CRC (Streaming Only)**] [P0x23] [**Encrypted CRC (Streaming Only)**] [P0x23] [**Format Code (Streaming Only)**] [P0x22]

If the device detects an error on a track, it transmits ASCII character “E” in place of the track data to indicate an error.

The device sends the **Device Encryption Status** value to help the host parse and interpret the incoming stream of data:

- The device only encrypts data if Encryption Enabled (bit 2) and Initial DUKPT Key Injected (bit 1) are set. Otherwise, it instead sends data it would usually encrypt as clear text in ASCII HEX format, and does not include the **DUKPT Key Serial Number**.
- When the DUKPT Keys Exhausted (bit 0) is set, the device no longer reads cards, and the card data format in **Table 3-1** excludes **MagnePrint Status**, **Encrypted MagnePrint Data**, **Masked Track Data**, **Encrypted Track Data**, and all corresponding Pre-Track Strings, Start Sentinels, End Sentinels, and Post-Track Strings. All other delimiters and data elements remain the same.

3.1.2 Commands and Responses In Streaming Format

If section **2 Connection Types** says the connection the device and host should use streaming format to exchange **Commands** and their corresponding responses over the selected connection type, all command and response messages are composed of a series of hexadecimal values encoded as two readable ASCII characters (‘0’ through ‘F’ only) per byte.

Each command should include the data shown in **Table 3-2**, and each response includes the data shown in **Table 3-3**.

Table 3-2 - Command Format for Streaming

Byte	Meaning
0..n	Command Request Message from Table 7-1 in section **7.1 About Commands**
n+1	Carriage Return (ASCII 0x0D)

Table 3-3 - Response Format for Streaming

Byte	Meaning
0..n	Command Response Message from **Table 7-2** in section **7.1 About Commands**
n+1	Carriage Return (ASCII 0x0D)

For example:

- A command with a one byte parameter in the **Command Request Data** field in Table 7-1 would send ASCII ‘0’ (0x30), ASCII ‘1’ (0x31) representing a length of `0x01`; a command with 18 bytes of data would send ASCII ‘1’ (0x31), ASCII ‘2’ (0x32) representing a length of `0x12`.
- To send **Command 0x00 - Get Property** to get **Property 0x03 - Device Serial Number**, the host would send a stream consisting of ASCII ‘0’ (0x30), ASCII ‘0’ (0x30) for Command Number 0x00, ASCII ‘0’ (0x30), ASCII ‘1’ (0x31) for Data Length 0x01, ASCII ‘0’ (0x30), ASCII ‘3’ (0x33) for Data, and a Carriage Return (0x0D) to signal the end of the message.
- The device’s responses are encoded similarly.

4 Security Levels

Devices can be configured to operate at different Security Levels, which affects **Magnetic Stripe Card Data Sent from Device to Host (MSR Only | Keypad Entry Only)**, the host software's ability to modify **Properties**, and the host software's ability to execute certain **Commands**. The Security Level can be increased by sending commands to the device, but can never be decreased. The sections below provide details about how each security level affects device behavior.

4.1 About Message Authentication Codes (MAC)

Commands in this manual that are tagged "MAC" are **privileged commands**. If the device is set to a Security Level higher than **Security Level 2**, the host software must calculate and append a four-byte Message Authentication Code ("MAC") to the Data field of the message, extending the length of the field by 4 bytes, to prove the sender is authorized to execute that command. If the device is set to **Security Level 2**, the device ignores the MAC field and the Device Serial Number field and the host can set them to all zeroes. If a MAC is required but not present or incorrect, the device returns `0x07`.

In most cases, the host must calculate the MAC using the current DUKPT Key (which can be retrieved using **Command 0x09 - Get Current TDES DUKPT KSN** to get a reference to the key). In some cases, documented in the commands that are affected by it, the host must compute the MAC using the UIK installed in the device. In cases where only MagTek knows the UIK, MagTek must be involved to populate the MAC field.

The host must calculate the MAC over the whole command per ***ISO 9797-1***, MAC Algorithm 3, Padding Method 1, using the **Message Authentication, request or both ways** variant as specified in ***ANS X9.24-1:2009, Annex A***. Data supplied to the MAC algorithm should be provided in raw binary form, not converted to ASCII-hexadecimal.

Upon successfully completing a MACed command that used the DUKPT key, the device advances the DUKPT Key.

The serial number value included with MACs is always 16 bytes long. The 16th byte always contains 0x00. If the serial number is less than 15 bytes, it is left-justified and padded with binary zeroes.

4.2 Security Level 2

Security Level 2 is the least secure mode. In this mode, keys are loaded but the device does not require the host software to use them for most operations: Keys are used/needed to load new keys and to move to Security Level 3 or 4, but all other properties and commands are freely usable. The host can use **Command 0x15 - Get / Set Security Level (MAC)** to determine the device's current security level.

4.3 Security Level 3

At Security Level 3, many commands require security; most notably **Command 0x01 - Set Property (MAC)**. See section **4.1 About Message Authentication Codes (MAC)** for details. The host can use **Command 0x15 - Get / Set Security Level (MAC)** to determine the device's current security level.

Security Level 3 also enables encryption of data and inclusion of encrypted data where it may have been left out at a lower security level. For a list of specific data the device encrypts at this security level and how the host can decrypt it, see section **5 Encryption, Decryption, and Key Management**.

4.4 Security Level 4 (MSR Only)

When the device is at Security Level 4, the device requires the host to successfully complete an Authentication Sequence before it will transmit data from a magnetic stripe card swipe (see section **7.3.6**

Command 0x10 - Activate Authenticated Mode). Correctly executing the Authentication Sequence also causes the green LED to blink, alerting the operator that the device is being controlled by a host with knowledge of the keys—that is, an Authentic Host. The host can use **Command 0x15 - Get / Set Security Level (MAC)** to determine the device's current security level.

4.5 Command Behaviors By Security Level

Table 4-1 shows the commands that are affected by the device's security level. Commands that are not affected by the security level are not listed. The key is as follows:

- **Y** means the command can run at the specified security level.
- **N** means the command is prohibited at the specified security level.
- **C** means the customer may specify **Y** or **S** for that command when ordering.
- **S** means the command is secured [may require MACing, see section **4.1 About Message Authentication Codes (MAC)**].
- * indicates **Command 0x02 - Reset Device** has special behavior. If an Authentication sequence has failed, only a correctly MACed **Command 0x02 - Reset Device (MAC)** can be used to reset the device. This is to prevent a dictionary attack on the keys and to minimize a denial of service (DoS) attack.

Table 4-1 - Command Behaviors At Each Security Level

Command	Level 2	Level 3	Level 4 (MSR Only)
Any command that is not listed in this table works the same at all Security Levels.	Y	Y	Y
Command 0x01 - Set Property (MAC)	Y	S	S
Command 0x02 - Reset Device (MAC)	Y	*	*
Command 0x10 - Activate Authenticated Mode	N	Y	Y
Command 0x11 - Activation Challenge Response	N	Y	Y
Command 0x12 - Deactivate Authenticated Mode	N	Y	Y
Command 0x15 - Get / Set Security Level (MAC)	S	S	S

5 Encryption, Decryption, and Key Management

5.1 About Encryption and Decryption

Some data exchanged between the device and the host is encrypted. This includes **Encrypted Track Data**, **Encrypted MagnePrint Data**, **Encrypted Session ID**, and **Encrypted CRC (Streaming Only)**. To decrypt this data, the host must first determine what key to use, then decrypt the data.

5.2 How to Determine the Key

When the device is encrypting data (see **Security Levels**), the host software must do the following to generate a key (the "derived key") to use for decryption:

1) **Determine the value of the Initial Key loaded into the device**. The lookup methods the host software uses depend on the overall solution architecture, and are outside the scope of this document. However, most solutions do this in one of two ways, both of which use the Initial Key Serial Number that arrives with the encrypted data (see **Command 0x09 - Get Current TDES DUKPT KSN** for details about interpreting the KSN):
    a) Look up the value of the Base Derivation Key using the Initial KSN portion of the current KSN as an index value, then use TDES DUKPT algorithms to calculate the value of the Initial Key; or
    b) Look up the value of the Initial Key directly, using the Initial KSN portion of the current KSN as an index value.
2) **Derive the current key**. Apply TDES DUKPT algorithms to the Initial Key value and the encryption counter portion of the KSN that arrives with the encrypted data.
3) **Determine which variant of the current key the device used to encrypt**. The variants are defined in ***ANS X9.24-1:2009 Annex A***, which programmers of host software must be familiar with. Which variant the host should use depends on the type of data the host is decrypting or encrypting, and on device settings:
    a) **Encrypted CRC (Streaming Only)** data is always encrypted using the **Message Authentication, request or both ways** variant.
    b) **Encrypted MagnePrint Data** is encrypted according to the setting in **Property 0x54 - Card Data Encryption Variant (MSR Only, Configurable MSR Variants Only)**, if the device supports it. Otherwise, it is encrypted using the **PIN Encryption variant**.
    c) **Encrypted Track Data** and **Encrypted Session ID** is encrypted according to the setting in **Property 0x54 - Card Data Encryption Variant (MSR Only, Configurable MSR Variants Only)**, if the device supports it. Otherwise, it is encrypted using the **PIN Encryption variant**.
4) Use the variant algorithm with the current key to calculate that variant.
5) Decrypt the data according to the steps in section **5.3 How to Decrypt Data**.

5.3 How to Decrypt Data

For **Encrypted Track Data**, the device begins by encrypting the first 8 bytes of clear text track data. The 8-byte result of this encryption is placed in an encrypted data buffer. The process continues using the DES 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 device continues until all clear text bytes have been encrypted. If the final block of clear text contains fewer than 8 bytes, the device pads the end of the block to make 8 bytes. After the final clear text block is XORed with the prior 8 bytes of encrypted data, the device encrypts it and places it in the encrypted data value. No Initial Vector is used in the process.

The host must decrypt the data in 8 byte blocks, ignoring any final unused bytes in the last block. When a value consists of more than one block, the host should use the CBC method to decrypt the data by following these steps:

1) Start decryption on the last block of 8 bytes (call it block N) using the key.
2) XOR the result of the decryption with the next-last block of 8 bytes (block N-1).
3) Repeat until reaching the first block.
4) Do not XOR the first block with anything.
5) Concatenate all blocks.
6) Determine the expected length of the decrypted data. In some cases this may be a standard field length, and in other cases the expected data length may accompany the encrypted data. When decrypting track data where no length is available, the host software can use the End Sentinel to find the actual end of the data (ignoring the padding at the end, which contains all zeroes).
7) Truncate the end of the decrypted data block to the expected data length, which discards the padding at the end.

6 Magnetic Stripe Card Data Sent from Device to Host (MSR Only | Keypad Entry Only)

The device sends card swipe data to the host even if it can not fully decode the data. How the host interprets incoming messages to find the data detailed in this section depends on the connection type (see section **2 Connection Types**) and the data format (see section **3 Data Formats**). Each subsection is tagged with the features, connection types, and data formats for which it is relevant. **Table 6-1** provides a convenient summary / index of all available values and their offsets.

Table 6-1 - List of Magnetic Stripe Data Sorted By GATT/SLIP Offset

Data	HID Usage	GATT/SLIP Offset
Track 1 Encrypted Data	0x30	7..118
Track 2 Encrypted Data	0x31	119..230
Track 3 Encrypted Data	0x32	231..342
MagnePrint Status	0x23	344..347
Encrypted MagnePrint Data	0x33	349..476
Device Serial Number	0x40	477..492
Device Encryption Status	0x42	493..494
DUKPT Key Serial Number (KSN)	0x46	495..504
Track 1 Masked Data	0x4A	508..619
Track 2 Masked Data	0x4B	620..731
Track 3 Masked Data (3-Track Only)	0x4C	732..843
Encrypted Session ID	0x50	844..851
Remaining MSR Transactions	0x55	856..858
Clear Text CRC (Streaming Only)	N/A	N/A
Encrypted CRC (Streaming Only)	N/A	N/A
Format Code (Streaming Only)	N/A	N/A

6.1 About Track Data

After the host receives and decrypts **Encrypted Track Data** from the device, or receives clear text track data (based on device settings or state), or receives **Masked Track Data**, it may need to parse each track into individual values embedded in the tracks. The device can read multiple card formats, which vary even between different issuers and payment brands using the same underlying standards. Describing all possible formats is beyond the scope of this document, but this section describes how to parse data from tracks 1, 2, and 3 in a generic ISO/ABA compliant format as an example.

Table 6-2 shows an example of ISO/ABA track data the device sends to the host, using unmasked placeholder numbers to make it easier to see the relative positions of the values embedded in the track data. It is important to note that some cards do not include Track 3 data, and some devices do not read or transmit Track 3 data (see section **1.5 About Device Features**).

Table 6-2 – Example Generic ISO/ABA Track Data Format

Generic ISO/ABA Track Data Format	
Track 1 Data	%75555555555555555^CARDHOLDER NAME/^333388800044440000006?
Track 2 Data	;5555555555555555=33338880004444006?
Track 3 Data	;5555555555555555=333388800044440000006?

The example track data in **Table 6-2** can be interpreted as follows:

- The %, ?, and ; are Sentinels / delimiters, and are taken directly from the data on the card, except when using Streaming format, where they may be overridden by **Properties** as described in section **3.1.1 Magnetic Stripe Card Data In Streaming Format (Swipe Only | Keypad Entry Only)**.
- The 7 at the beginning of Track 1 data is the card format code. For swiped credit / debit cards, this comes from the card and is generally B.
- The string of 5s is the Account Number / License Number / PAN.
- The carets ^ are a standard ISO track 1 delimiter surrounding the Cardholder Name.
- CARDHOLDER NAME/ is the Cardholder Name.
- The string of 3s is the Expiration Date.
- The string of 8s is the Service Code. For swiped credit / debit cards, this comes from the card.
- The remaining characters (0s, 4s, and 6) are Discretionary Data. For swiped debit / credit cards this data is of varying length and content and comes from the card, and must be interpreted according to the standards established by issuers, payment brands, and so on.

6.2 Device Encryption Status

This two-byte value contains the Device Encryption Status in big endian byte order. Byte 1 is the least significant byte; the LSB of byte 1 is status bit 0, and the LSB of byte 2 is status bit 15.

If the **Encryption Enabled** bit or **Initial DUKPT Key Injected** bit are not set, the device sends card data it would usually encrypt as clear text, and does not include a valid **DUKPT Key Serial Number**.

When the **DUKPT Keys Exhausted** bit is set, the device no longer reads cards, but continues to send **Magnetic Stripe Card Data Sent from Device to Host (MSR Only | Keypad Entry Only)** to report status. The data it sends to the host in this case does not include valid **MagnePrint Status**, **Encrypted MagnePrint Data**, **Masked Track Data**, or **Encrypted Track Data**.

Format	Where to Find Value
HID	Usage 0x42
Streaming	See **Table 3-1** p. **22**. Hex-encoded binary.
TLV	Data Object 8001
GATT/SLIP	Offset 493..494

The Device Encryption Status is defined as follows:

Bit	Meaning
0	DUKPT keys exhausted (1 = Exhausted, 0 = Keys available)
1	Initial DUKPT key injected, always set to 1
2	Encryption Enabled, always set to 1
3	Authentication Required
4	Timed out waiting for cardholder to swipe card
5	Reserved
6	Reserved
7	Reserved
8	No MSR Transactions Remaining [see **Command 0x1C - Get Remaining MSR Transactions Counter (MSR Only)**]
9	Reserved
10	Reserved
11	(Configurable MSR Variants Only) DUKPT Key Variant used to encrypt **Encrypted Track Data**. 0 = PIN Encryption. 1 = Data Encryption, request or both ways
12	Reserved
13	Reserved
14	Unused (always set to 0)

Bit	Meaning
15	Unused (always set to 0)

6.3 Encrypted Track Data

If decodable track data exists for a given track, the device returns it in the corresponding **Track x Encrypted Data** value, described in the subsections below.

When the device is transmitting data in HID, GATT, or SLIP format, the **Encrypted Data** values are always 112 bytes long, which is the maximum number of bytes that can be encoded on a card. However, the length of actual valid data in each value may be less than 112 bytes, and is stored in the corresponding **Encrypted Data Length** value. The host software should ignore data located beyond the data length reported by the device.

The device decodes the data from each track on the card and converts it to ASCII, and includes all data starting with the start sentinel and ending with the end sentinel. For additional information about configuration-specific or card-type-specific start and end sentinel behavior, see sections **8.18**, **8.19**, **8.20**, **8.21**, **8.22**, **8.23**, **8.24**, **8.26**, **8.27**, and **8.28**.

If the device is in a security level below **Security Level 3**, it sends the resulting track data in the **Track x Encrypted Data** values unencrypted. If the device is in **Security Level 3** or **Security Level 4**, it encrypts the data before sending. For information about how the device encrypts the data and how the host should decrypt it, see section **5 Encryption, Decryption, and Key Management**.

6.3.1 Track 1 Encrypted Data

For information about the contents of track data, see section **6.1 About Track Data**. For information about decryption, see section **5 Encryption, Decryption, and Key Management**.

Format	Where to Find Value
HID	Usage 0x30
Streaming	See **Table 3-1** p. **22**. Hex-encoded binary.
TLV	Data Object 8215
GATT/SLIP	Offset 7..118

6.3.2 Track 2 Encrypted Data

For information about the contents of track data, see section **6.1 About Track Data**. For information about decryption, see section **5 Encryption, Decryption, and Key Management**.

Format	Where to Find Value
HID	Usage 0x31
Streaming	See **Table 3-1** p. **22**. Hex-encoded binary.
TLV	Data Object 8216
GATT/SLIP	Offset 119..230

6.3.3 Track 3 Encrypted Data

On 2-track devices (see **Table 1-2 - Device Features**), this value is included in incoming data as a null value.

For information about the contents of track data, see section **6.1 About Track Data**. For information about decryption, see section **5 Encryption, Decryption, and Key Management**.

Format	Where to Find Value
HID	Usage 0x32
Streaming	See **Table 3-1** p. **22**. Hex-encoded binary.
TLV	Data Object 8217
GATT/SLIP	Offset 231...342

6.4 MagnePrint Status

This four-byte value contains 32 bits of MagnePrint status information. The first byte, byte 1, is the least significant byte and its least significant bit is status bit 0. The final byte, byte 4, is the most significant byte and its most significant bit is status bit 31. For an example, see **Table 6-3** on page **33** which shows how to interpret MagnePrint Status bits for a value of `A1050000`.

If the device is set to a security level below **Security Level 3**, it uses the current value of **Property 0x15 - MagnePrint Flags** to determine the behavior of this value.

- Bit 0 = MagnePrint capable flag
- Bits 1 to 15 = Product revision & mode
- Bit 16 = Reserved
- Bit 17 = Reserved for noise measurement
- Bit 18 = Swipe too slow
- Bit 19 = Swipe too fast
- Bit 20 = Reserved
- Bit 21 = Actual card swipe direction (0 = Forward, 1 = Reverse)
- Bits 22-31 = Reserved

Format	Where to Find Value
HID	Usage 0x23
Streaming	See **Table 3-1** p. **22**. Hex-encoded binary.
TLV	Data Object 8263
GATT/SLIP	Offset 344..347

Table 6-3 - MagnePrint Status Example for Value A1050000

Byte #	Nibble #	Bit #	Hex Value	Binary Value	Bit Meaning
Byte 1 = A1	1	7	A	1	Product Revision/Mode
		6		0	Product Revision/Mode
		5		1	Product Revision/Mode
		4		0	Product Revision/Mode
	2	3	1	0	Product Revision/Mode
		2		0	Product Revision/Mode
		1		0	Product Revision/Mode
		0		1	MagnePrint capable
Byte 2 = 05	3	15	0	0	Product Revision/Mode
		14		0	Product Revision/Mode
		13		0	Product Revision/Mode
		12		0	Product Revision/Mode
	4	11	5	0	Product Revision/Mode
		10		1	Product Revision/Mode
		9		0	Product Revision/Mode
		8		1	Product Revision/Mode
Byte 3 = 00	5	23	0	0	Reserved
		22		0	Reserved
		21		0	Direction
		20		0	Reserved
	6	19	0	0	Too Fast
		18		0	Too Slow
		17		0	Reserved for noise measurement
		16		0	Reserved
Byte 4 = 00	7	31	0	0	Reserved
		30		0	Reserved
		29		0	Reserved
		28		0	Reserved
	8	27	0	0	Reserved
		26		0	Reserved
		25		0	Reserved
		24		0	Reserved

6.5 Encrypted MagnePrint Data

This value contains Encrypted MagnePrint data, which when decrypted generally yields a 54-byte value. The least significant bit of the first byte of data in the decrypted value corresponds to the first bit of MagnePrint data.

If the device is set to a security level below **Security Level 3**, it uses the current value of **Property 0x15 - MagnePrint Flags** to determine the behavior of this value.

To derive a decrypted MagnePrint value to authenticate a card, the host should decrypt the data block.

Format	Where to Find Value
HID	Usage 0x33
Streaming	See **Table 3-1** p. **22**. Hex-encoded binary.
TLV	Data Object 8218
GATT/SLIP	Offset 349..476

6.6 Device Serial Number

This 16-byte ASCII value contains the device serial number in a null-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 **Property 0x03 - Device Serial Number**. This value is stored in non-volatile memory, so it persists when the device is power cycled.

Format	Where to Find Value
HID	Usage 0x40
Streaming	See **Table 3-1** p. **22**. ASCII.
TLV	Data Object 8102
GATT/SLIP	Offset 477..492

6.7 Masked Track Data

6.7.1 About Masking

If decodable track data exists for a given track, the device uses the **Track x Masked Track Data** value for that track to send a masked version of the data. The masked version includes one byte of data for each character decoded from the track, starting with the Start Sentinel and ending with the End Sentinel.

In masked track data, the device sends a specified mask character instead of the actual character read from the track. Only ISO/ABA (Financial Cards with ***ISO/IEC 7813*** Format code B) and AAMVA cards are selectively masked; all other card types are either sent entirely masked or entirely unmasked. More detail about masking is included in the sections below about each specific track.

There are separate masking settings for ISO/ABA format cards and AAMVA format cards (See **Property 0x07 - ISO Track Mask** and **Property 0x08 - AAMVA Track Mask** for more information). Each of these settings allows the host software 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 a correction should be applied to make the Mod 10 (Luhn algorithm) digit at the end of the PAN be correct.

Table 6-4 provides an example of data from tracks 1, 2, and 3 of a swiped ISO/ABA card after it has been decrypted or if the device has sent it in the clear. **Table 6-5** shows the same data as it might appear with a specific set of **Masked Track Data** rules applied.

Table 6-4 – Sample ISO/ABA Swiped Track Data, Clear Text / Decrypted

Sample ISO/ABA Swiped Track Data, Clear Text / Decrypted	
Track 1	%B6011000995500000^ TEST CARD ^15121015432112345678?
Track 2	;6011000995500000=15121015432112345678?
Track 3	;6011000995500000=1512101543211234567833333333333333333333?

Table 6-5 – Sample ISO/ABA Swiped Track Data, Masked

Sample ISO/ABA Swiped Track Data, Masked	
Track 1	%B6011000020000000^ TEST CARD ^15120000000000000000?
Track 2	;6011000020000000=15120000000000000000?
Track 3	;6011000020000000=0000000000000000000000000000000000000000?

Data Formats with fixed Data field lengths (such as USB HID format, GATT format, and SLIP format, which are fixed at 112 bytes) include a **Masked Track Data Length** value for each track, which the host should use to truncate and ignore undefined data past the end of the track data. Formats where the host can easily determine where masked track data begins and ends (such as formats with delimiters or with data length built in to the format itself) do not include explicit masked track data lengths.

6.7.2 Track 1 Masked Data

This value contains the masked track data for track 1. All characters are transmitted. For information about the contents of track data, see section **6.1 About Track Data**. For general information about masking, see section **6.7.1 About Masking** and **Appendix B Identifying ISO/ABA and AAMVA Cards For Masking (MSR Only)**.

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

- The number of initial characters and trailing characters specified by **Property 0x07 - ISO Track Mask** is sent unmasked. If Mod 10 correction is specified (see section **8.6 Property 0x07 - ISO Track Mask**), all but one of the intermediate characters of the PAN are set to zero; one of them is set such that the 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.
- Cardholder Name and the Expiration Date are sent unmasked.
- The Service Code is always unmasked on newer devices (Never Mask Service Code Only). On legacy devices, the Service Code is always masked.
- 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 all characters read from the card.

Format	Where to Find Value
HID	Usage 0x4A (112 bytes fixed, must be truncated)
Streaming	See **Table 3-1** p. **22**. ASCII.
TLV	Data Object 8221
GATT/SLIP	Offset 508..619

6.7.3 Track 2 Masked Data

This 112-byte value contains the masked track data for track 2. For general information about masking, see section **6.7.1 About Masking** and **Appendix B Identifying ISO/ABA and AAMVA Cards For Masking (MSR Only)**.

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

- The number of initial characters and trailing characters specified by **Property 0x07 - ISO Track Mask** is sent unmasked. If Mod 10 correction is specified (see **Property 0x07 - ISO Track Mask**), all but one of the intermediate characters of the PAN are set to zero; one of them is set such that the 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 sent unmasked.
- The Service Code is always unmasked on newer devices (Never Mask Service Code Only). On legacy devices, the Service Code is always masked.
- 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 (see **Property 0x08 - AAMVA Track Mask**), all but one of the intermediate characters of the DL/ID#PAN are set to zero; one of them is 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.

Format	Where to Find Value
HID	Usage 0x4B (112 bytes fixed, must be truncated)
Streaming	See **Table 3-1** p. **22**. ASCII.
TLV	Data Object 8222
GATT/SLIP	Offset 620-731

6.7.4 Track 3 Masked Data (3-Track Only)

This 112-byte value contains the Masked Track Data for track 3. On 2-track devices (see **Table 1-2 - Device Features**), this value is not included in the incoming data. For general information about masking, see section **6.7.1 About Masking** and **Appendix B Identifying ISO/ABA and AAMVA Cards For Masking (MSR Only)**.

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

- The number of initial characters and trailing characters specified by **Property 0x07 - ISO Track Mask** is sent unmasked. If Mod 10 correction is specified (see section **8.6 Property 0x07 - ISO Track Mask**), all but one of the intermediate characters of the PAN are set to zero; one of them is 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 all characters read from the card.

Format	Where to Find Value
HID	Usage 0x4C (112 bytes fixed, must be truncated)
Streaming	See **Table 3-1** p. **22**. ASCII.
TLV	Data Object 8223
GATT/SLIP	Offset 732-843

6.8 Encrypted Session ID

This 8-byte value contains the encrypted version of the current Session ID. Its primary purpose is to prevent replays. After a card is read, this property is encrypted, along with the card data, and supplied as part of the transaction message. The clear text version is never transmitted. To avoid replay, the host software should set the Session ID property before a transaction, and verify that the Encrypted Session ID returned with card data decrypts to the original value it set.

Format	Where to Find Value
HID	Usage 0x50
Streaming	See **Table 3-1** p. **22**. Hex-encoded binary.
TLV	Data Object 8309
GATT/SLIP	Offset 844..851

6.9 DUKPT Key Serial Number (KSN)

This 80-bit value contains the TDES DUKPT **Key Serial Number** (KSN) associated with encrypted values included in the same message. For details about how to interpret this value, see section **7.3.4 Command 0x09 - Get Current TDES DUKPT KSN**. If no keys are loaded, all bytes have the value `0x00`.

Format	Where to Find Value
HID	Usage 0x46
Streaming	See **Table 3-1** p. **22**. Hex-encoded binary.
TLV	Data Object 8301
GATT/SLIP	Offset 495..504

6.10 Remaining MSR Transactions

This 3-byte value contains the number of MSR transactions remaining at the end of the current transaction. The value is also sometimes referred to as the transaction threshold. See **Command 0x1C - Get Remaining MSR Transactions Counter (MSR Only)** and **Property 0x30 - Send Remaining MSR Transactions Counter (Streaming Only, MSR Only)** for more information.

Format	Where to Find Value
HID	Usage 0x55
Streaming	See **Table 3-1** p. **22**. Hex-encoded binary.
TLV	Data Object 810A
GATT/SLIP	Offset 856..858

6.11 Clear Text CRC (Streaming Only)

This two-byte value contains a clear text version of a Cyclical Redundancy Check (CRC-16 CCITT, polynomial 0x1021), with the 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 host software 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 host software can have high confidence that all the data was received correctly. **Property 0x19 - CRC Flags (Streaming Only, MSR Only)** controls whether this value is sent.

Format	Where to Find Value
HID	N/A
Streaming	See **Table 3-1** p. **22**. Hex-encoded binary.
TLV	N/A
GATT/SLIP	N/A

6.12 Encrypted CRC (Streaming Only)

This 8-byte value 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; the remaining bytes are unused. The host software 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 host software can have high confidence that all the data was received correctly. **Property 0x19 - CRC Flags (Streaming Only, MSR Only)** controls whether this value is sent.

Format	Where to Find Value
HID	N/A
Streaming	See **Table 3-1** p. **22**. Hex-encoded binary.
TLV	N/A
GATT/SLIP	N/A

6.13 Format Code (Streaming Only)

This four-character ASCII value contains the Format Code [stored in **Property 0x2C - Format Code (Streaming Only, MSR Only)**], which provides information for the host software to locate values within the incoming message:

Format	Where to Find Value
HID	N/A
Streaming	See **Table 3-1** p. **22**. ASCII.
TLV	N/A
GATT/SLIP	N/A

7 Commands

This section describes the commands available on the device. Each command’s section heading indicates the **Connection Types**, **Data Formats**, and device features (see section **1.5 About Device Features**) that are relevant to it.

7.1 About Commands

Regardless of connection type and data format, all MagneSafe V5 devices use common structures to receive command request messages from the host and to send command response messages back. For information about connection-specific wrappers for these commands, see section **2 Connection Types**.

Table 7-1 - Command Request Message (Host Sends to Device to Initiate a Command)

Offset	Field Name
0	Command Number
1	Command Request Data Length
2..n Maximum / fixed length depends on device and connection type.	Command Request Data

Command Number is a one byte value that contains the requested command number. Section **7 Commands** lists all available commands.

Command Request Data Length is a one byte value that contains the length of the **Command Request Data** field.

Command Request Data contains command data as defined in the documentation for the selected command in section **7 Commands**.

Table 7-2 - Command Response Message (Host Sends to Device to Retrieve Data or Responses)

Offset	Field Name
0	Result Code
1	Command Response Data Length
2..n	Command Response Data

Result Code is a one-byte value the device sends to indicate success or the failure mode of the command. Section **7.2 About Result Codes** provides more detail.

Command Response Data Length is a one byte value that contains the length of the **Command Response Data** field.

Command Response Data contains response data as defined in the documentation for the selected command in section **7 Commands**.

7.2 About Result Codes

There are two types of **Result Code** values the device can return in its response: **Generic** result codes (listed in **Table 7-3**), which have the same meaning for all commands, and **command-specific** result codes, which can have different meanings for different commands, and are listed with every command in this section. Generic result codes always have the most significant bit set to zero, and command-specific result codes always have the most significant bit set to one.

Table 7-3 - Generic Result Codes

Value (Hex)	Result Code	Description
0x00	Success	The command completed successfully.
0x01	Failure	The command failed.
0x02	Bad Parameter	The command failed due to a bad parameter or command syntax error.
0x03	Redundant	The command is redundant.
0x04	Bad Cryptography	A bad cryptography operation occurred.
0x05	Delayed	The request is refused because the device is delaying requests as a defense against brute-force hacking.
0x06	No Keys	No keys are loaded.
0x07	Invalid Operation	Depends on the context of the command.
0x08	Response not available	The response is not available.
0x09	Not enough power	The battery is too low to operate reliably.
0x0A	Reserved	Reserved
0x0B	Reserved	Reserved
0x0C	Reserved	Reserved
0x0D	Not implemented	The command is not implemented.
0x0E	Reserved	Reserved
0x0F	Reserved	Reserved
0x55	Failure communicating with Embedded V5 IntelliHead (Embedded V5 Head Only)	The device's main microcontroller is unable to communicate with the device's embedded MagneSafe V5 IntelliHead. This indicates a hardware failure. Return the device to the manufacturer for service.

7.3 General Commands

7.3.1 Command 0x00 - Get Property

This command gets a property from the device. For details about properties, see section **8 Properties**.

Most properties have a firmware default value that may be changed during manufacturing or the order fulfillment process to support different customer needs.

Table 7-4 - Request Data for Command 0x00 - Get Property

Data Offset	Value
0	Property ID

Table 7-5 - Response Data for Command 0x00 - Get Property

Data Offset	Value
0..n	Property Value

Property ID is a one-byte value that identifies the property. A full list of properties can be found in section **8 Properties**.

Property Value consists of the multiple-byte value of the property. The number of bytes in this value depends on the type of property and the length of the property. **Table 7-6** describes the available property types.

Table 7-6 - Property Types

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

The result codes for the **Get Property** command can be any of the generic result codes listed in **Table 7-3** on page **41**.

7.3.2 Command 0x01 - Set Property (MAC)

This command sets a property in the device. For security purposes, this command is privileged. When the Security Level is set to higher than 2 (see section **4 Security Levels**), this command must be MACed to be accepted [see section **4.1 About Message Authentication Codes (MAC)**]. The command is logically paired with **Command 0x00 - Get Property**. For details about properties, see section **8 Properties**.

Some properties require the device to be reset using **Command 0x02 - Reset Device (MAC)** or power cycled to take effect. In those cases, the documentation for the property indicates what is required.

Table 7-7 - Request Data for Command 0x01 - Set Property (MAC)

Data Offset	Value
0	Property ID
1..n	Property Value

Response Data: None

The result codes for the **Set Property** command can be any of the generic result codes listed in **Table 7-3** on page **41**. If the **Set Property** command gets a result code of `0x07`, it means the required MAC was absent or incorrect.

Property ID is a one-byte value that identifies the property. A full list of properties can be found in section **8 Properties**.

Property Value consists of multiple bytes containing the value of the property. The number of bytes in this value depends on the property. **Table 7-4** describes the available property types.

Table 7-8 - Response Data for Command 0x01 - Set Property (MAC)

Property Type	Description
Byte	This is a one-byte value. The range of valid values depends 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 data length listed in the tables for each property does not include the terminating NULL character.

7.3.3 Command 0x02 - Reset Device (MAC)

This command is used to reset the device, and can be used to make property changes take effect without power cycling the device.

(MSR Only) If the device is in the midst of an Authentication Sequence initiated by **Command 0x10 - Activate Authenticated Mode (MSR Only)**, the device does not honor the Reset Device command until after the Authentication Sequence has successfully completed, or a cardholder swipes a card, or the device is power cycled. If the Authentication Sequence fails, the device initiates anti-hack mode and will require that the host MAC the Reset Device command (see section **4 Security Levels**). This prevents a dictionary attack on the keys and reduces the potential impact of denial of service attacks.

In rare instances, devices may optionally be configured at the manufacturer to require a MAC for every Reset Device command call, not just when anti-hack behavior is active.

Request Data Field: None

Response Data Field: None

Result codes:
0x00 = Success
0x07 = Incorrect MAC, or authentication sequence is pending

Example Request (Hex)

Cmd Num	Data Len	Data
02	00	

Example Response (Hex)

Result Code	Data Len	Data
00	00	

7.3.4 Command 0x09 - Get Current TDES DUKPT KSN

The host uses this command to get the current Triple Data Encryption Standard (TDES) DUKPT Key Serial Number (KSN) on demand.

This 80-bit value contains the TDES DUKPT **Key Serial Number** (KSN) associated with encrypted values included in the same message. The rightmost 21 bits are the current value of the encryption counter. The leftmost 59 bits are the device's **Initial KSN**, which is a combination of the **Key Set ID** that identifies the Base Derivation Key (BDK) injected into the device during manufacture, and the device's serial number (DSN); how those two values are combined into the 59 bit Initial KSN is defined by a convention the customer defines when architecting the solution, with support from MagTek. For example, one common scheme is to concatenate a 7 hex digit (28 bit) Key Set ID, a 7 hex digit (28 bit) Device Serial Number, and 3 padding zero bits. In these cases, the key can be referenced by an 8-digit MagTek part number ("key ID") consisting of the 7 hex digit Key Set ID plus a trailing "0."

Request Data: None

Table 7-9 - Response Data for Command 0x09 - Get Current TDES DUKPT KSN

Offset	Field Name	Description
0	Current Key Serial Number	80-bit TDES DUKPT KSN

Result codes:
0x00 = Success
0x02 = Bad Parameter - The Data field in the request 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

7.3.5 Command 0x0A - Set Session ID (MSR Only)

This command is used to set the current Session ID, which the device transmits to the host in the **Encrypted Session ID**. The new Session ID stays in effect until one of the following occurs:

- The host sends the device another Set Session ID command.
- The device is powered off.
- The device 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 the device reads a card, it encrypts the Session ID along with the card data, and supplies it as part of the **Magnetic Stripe Card Data Sent from Device to Host**. The device never transmits a clear text version of this data.

Table 7-10 - Request Data for Command 0x0A - Set Session ID (MSR Only)

Offset	Field Name	Description
0	New Session ID	This eight byte value may be any value the host software wishes.

Response Data: None

Result codes:
0x00 = Success
0x02 = Bad Parameter **-** The Data field in the request is not the correct length. The Session ID is an 8-byte value, 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	

7.3.6 Command 0x10 - Activate Authenticated Mode (MSR Only)

This command is used by the host software to activate Authenticated Mode, and is the only way to enter that mode. When the device is set to Security Level 4 (see section **4.4 Security Level 4**), it does not gather and transmit card data after a swipe until Authenticated Mode has been established with the host, indicating both devices have established a direct two-way trust relationship. The general sequence of events for entering Authenticated Mode is as follows:

1) The cardholder or operator performs an action as a lead-in to swiping a card, such as signing in to a web page that interacts with the device.
2) The host software is aware of the cardholder action, and in response it sends the Activate Authenticated Mode command to the device. As part of this command, the host software specifies a PreAuthentication Time Limit parameter in units of seconds. The device uses this time limit in subsequent steps. The device interprets any value less than 120 seconds to mean 120 seconds.
3) The device responds to the host with the current Key Serial Number (KSN) and two challenges (Challenge 1 and Challenge 2), which it encrypts using a custom variant of the current DUKPT Key (Key XOR F0F0 F0F0 F0F0 F0F0 F0F0 F0F0 F0F0 F0F0). Challenge 1 contains 6 bytes of random numbers followed by the last two bytes of the KSN. Challenge 2 contains 8 bytes of random numbers.
4) The device waits up to the PreAuthentication Time Limit. If the device times out waiting for the host to respond, the Authentication attempt fails and the device may activate anti-hacking behavior. See below for details.
5) The host software decrypts Challenge 1 and Challenge 2 and compares the last two bytes of the KSN with the last two bytes of the clear text KSN to authenticate the device.
6) The host software completes the Activate Authentication sequence using **Command 0x11 - Activation Challenge Response**, including the length of time the device should keep Authenticated Mode active without a swipe.
7) The device determines whether the Activation Challenge Reply is valid. If it is valid, the device activates Authenticated Mode and allows transmission of swiped card data to the host. The device may optionally indicate to the operator that the host and the device are mutually authenticated. See below for information about device behavior when the Activation Challenge Reply is not valid.
8) Authenticated mode stays active until the timeout previously specified by the host in **Command 0x11 - Activation Challenge Response**, or until the device sends valid swipe data to the host, at which point the device deactivates Authenticated Mode.

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 device activates anti-hacking behavior by enforcing an increasing delay between incoming Activate Authenticated Mode commands. The first delay is 10 seconds, increasing by 10 seconds up to a maximum delay of 10 minutes. The operator may deactivate anti-hacking mode at any time by swiping any encoded magnetic stripe card. When the device is in this anti-hacking mode, it requires the host to take additional steps to call **Command 0x02 - Reset Device**

To support use of Authenticated Mode, the host software can use **Command 0x14 - Get Device State (MSR Only)** at any time to determine the current state of the device.

Table 7-11 - Request Data for Command 0x10 - Activate Authenticated Mode (MSR Only)

Offset	Field Name	Description
0	PreAuthentication Time Limit (msb)	Most significant byte of the PreAuthentication Time Limit in seconds (120 seconds or greater)

Offset	Field Name	Description
1	PreAuthentication Time Limit (lsb)	Least significant byte of the PreAuthentication Time Limit in seconds (120 seconds or greater)

Table 7-12 – Response Data for Command 0x10 - Activate Authenticated Mode (MSR Only)

Offset	Field Name	Description
0	Current Key Serial Number	This eighty-bit value includes the Initial Key Serial Number in the leftmost 59 bits and the value of the encryption counter in the rightmost 21 bits.
10	Challenge 1	The host should use this eight-byte challenge later in **Command 0x11 - Activation Challenge Response**, and to authenticate the device.
18	Challenge 2	The host should use this eight-byte challenge later in **Command 0x12 - Deactivate Authenticated Mode**.

Result codes:
0x00 = Success
0x03 = Redundant - the device 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 (see section **4 Security Levels**)
0x80 = No MSR Transactions Remaining [see **Command 0x1C - Get Remaining MSR Transactions Counter (MSR Only)**]

Example Request (Hex)

Cmd Num	Data Len	Data
10	00	

Example Response (Hex)

Result Code	Data Len	Data
00	1A	FFFF 0123 4567 8000 0003 9845 A48B 7ED3 C294 7987 5FD4 03FA 8543

7.3.7 Command 0x11 - Activation Challenge Response (MSR Only)

This command is used as the second part of an Activate Authentication sequence following **Command 0x10 - Activate Authenticated Mode**. In this command, the host software sends the first 6 bytes of Challenge 1 (received in response to **Command 0x10 - Activate Authenticated Mode**) plus two bytes of timeout information, and (optionally) an eight byte Session ID encrypted with the a custom variant of the current DUKPT Key (Key XOR 3C3C 3C3C 3C3C 3C3C 3C3C 3C3C 3C3C 3C3C).

The time information contains the maximum number of seconds the device should remain in Authenticated Mode. Regardless of the value of this timer, a card swipe in the Authenticated Mode ends the Authenticated Mode. The maximum time allowed is 3600 seconds (one hour). For example, for a full hour, use `0x0E10`; for 3 minutes, use `0x012C`. A value of `0x00` forces the device to stay in Authenticated Mode until a card swipe or power down occurs (no timeout).

If the host includes Session ID information and the command is successful, it changes the Session ID in the device in the same way as calling **Command 0x0A - Set Session ID**.

If the device decrypts the Challenge Response correctly, Activate Authenticated Mode has succeeded. If the device can not decrypt the Challenge Response correctly, Activate Authenticated Mode fails and the TDES DUKPT Key Serial Number advances.

Table 7-13 - Request Data for Command 0x11 - Activation Challenge Response (MSR Only)

Offset	Field Name	Description
0	Response to Challenge 1	First 6 bytes of Challenge 1 plus a two-byte timeout (MSB first), 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 Data field in the request is not a correct length
0x04 = Bad Data - the encrypted reply data could not be verified
0x07 = Sequence - not expecting this command

Example Request (Hex)

Cmd Num	Data Len	Data
11	08	8579827521573495

Example Response (Hex)

Result Code	Data Len	Data
00	00	

7.3.8 Command 0x12 - Deactivate Authenticated Mode (MSR Only)

This command is used to exit Authenticated Mode initiated by **Command 0x10 - Activate Authenticated Mode**. It can be used to exit the mode with or without incrementing the DUKPT transaction counter (lower 21 bits of the Key Serial Number). The host software must send the first 7 bytes of Challenge 2 (from the response to **Command 0x10 - Activate Authenticated Mode**) and the Increment flag (0x00 indicates no increment, 0x01 indicates increment the KSN) encrypted with a custom variant of the current DUKPT Key (Key XOR 3C3C 3C3C 3C3C 3C3C 3C3C 3C3C 3C3C 3C3C).

If the device decrypts Challenge 2 successfully, it exits Authenticated Mode, and depending on the Increment flag, may increment the KSN.

If the device cannot decrypt Challenge 2 successfully, it stays in Authenticated Mode until either the time specified in **Command 0x10 - Activate Authenticated Mode** elapses or the cardholder or operator swipes a card. This behavior is intended to discourage denial of service attacks. Exiting Authenticated Mode by timeout or card swipe always increments the KSN; exiting Authenticated Mode using **Command 0x12 - Deactivate Authenticated Mode** may increment the KSN.

Table 7-14 - Request Data for Command 0x12 - Deactivate Authenticated Mode (MSR Only)

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

Response Data: None

Result codes:
0x00 = Success
0x02 = Bad Parameters - the Data field in the request is not the correct length
0x03 = Bad Data - the encrypted reply data could not be verified
0x07 = Sequence - not expecting this command

Example Request (Hex)

Cmd Num	Data Len	Data
12	08	8579827521573495

Example Response (Hex)

Result Code	Data Len	Data
00	00	

7.3.9 Command 0x14 - Get Device State (MSR Only)

When the device is set to **Security Level 4 (MSR Only)**, it requires mutual authentication with the host [see **Command 0x10 - Activate Authenticated Mode (MSR Only)**]. The host can use this command to determine the state of Authenticated Mode at a given point in time. For convenience, this manual refers to states with the notation ***State:Antecedent*** (e.g., **WaitActAuth:BadSwipe**), showing the current state and the state that led to it. Lists of possible states and their definitions are provided in the device response tables below.

In most cases, the host software can also track the state of Authenticated Mode by inference. As the host software interacts with the device, most state transitions are marked by the messages exchanged with the device. The exception is the transition from **WaitActRply:x** to **WaitActAuth:TOAuth**, which happens if the device times out waiting for the host to send **Command 0x11 - Activation Challenge Response (MSR Only)**, which the device does not report to the host. To cover this case, the host must be aware that a timeout could occur, in which case the device responds to **Command 0x11 - Activation Challenge Response (MSR Only)** with Result Code 0x07 (Sequence Error).

Example 1 – Power Up followed by Authentication and good swipe:

1) Device powers on. Host software should send this command to discover the current state of the device is WaitActAuth:PU.
2) Host sends a valid **Command 0x10 - Activate Authenticated Mode (MSR Only)**. Device responds with result code 0x00, inferring the transition to the WaitActRply:PU state.
3) Host sends a valid **Command 0x11 - Activation Challenge Response (MSR Only)**. Device responds with result code 0x00, inferring the transition to the WaitSwipe:PU state.
4) Cardholder swipes a card correctly. Device sends card data to the host, inferring the transition to the WaitActAuth:GoodSwipe state.

Example 2 – Device times out waiting for swipe:

1) Device waiting after a good swipe. Host software may send this command to discover the current state of the device is WaitActAuth:GoodSwipe.
2) Host sends valid **Command 0x10 - Activate Authenticated Mode (MSR Only)**. Device responds with result code 0x00, inferring the transition to the WaitActRply:GoodSwipe state.
3) Host sends a valid **Command 0x11 - Activation Challenge Response (MSR Only)**. Device responds with result code 0x00, inferring the transition to the WaitSwipe:GoodSwipe state.
4) Authenticated mode times out before a swipe occurs. Device sends mostly empty card data to the host to report the timeout in Device Encryption Status. The host infers the transition to the WaitActAuth:TOSwipe state.

Example 3 – Host sends invalid Command 0x11 - Activation Challenge Response (MSR Only):

1) Device waiting after a good swipe. Host software may send this command to discover the current state of the device is WaitActAuth:GoodSwipe.
2) Host sends valid **Command 0x10 - Activate Authenticated Mode (MSR Only)**. Device responds with result code 0x00, inferring the transition to the WaitActRply:GoodSwipe state.
3) Host sends invalid **Command 0x11 - Activation Challenge Response (MSR Only)**. Device responds with result code 0x02 or 0x04, inferring the transition to the WaitActAuth:FailAuth state.

Example 4 – Host waits too long sending Command 0x11 - Activation Challenge Response (MSR Only):

1) Device waiting after a good swipe. Host software may send this command to discover the current state of the device is WaitActAuth:GoodSwipe.
2) Host sends valid **Command 0x10 - Activate Authenticated Mode (MSR Only)**. Device responds with result code 0x00, inferring the transition to the WaitActRply:GoodSwipe state.
3) Device times out waiting for host to send **Command 0x11 - Activation Challenge Response (MSR Only)** (State => WaitActAuth:TOAuth). Host doesn't know because the device does not send any message.
4) Host eventually sends **Command 0x11 - Activation Challenge Response (MSR Only)** (State remains WaitActAuth:TOAuth). Device responds with result code 0x07, inferring the previous transition to WaitActAuth:TOAuth state.

Request Data: None

Table 7-15 - First Byte, Response Data for Command 0x14 - Get Device State (MSR Only)

Current Device State		
Value	Name	Meaning
0x00	WaitActAuth	Waiting for Activate Authenticated Mode. The device requires the host to authenticate using **Command 0x10 - Activate Authenticated Mode** before it accepts swipes.
0x01	WaitActRply	Waiting for Activation Challenge Reply. The host has started to authenticate, and the device is waiting for **Command 0x11 - Activation Challenge Response**.
0x02	WaitSwipe	Waiting for swipe. The device is waiting for the cardholder or operator to swipe a card.
0x03	WaitDelay	Waiting for Anti-Hacking Timer. Two or more previous attempts to Authenticate have failed; the device is waiting for the Anti-Hacking timer to expire before it accepts **Command 0x10 - Activate Authenticated Mode**.

Table 7-16 - Second Byte, Response Data for Command 0x14 - Get Device State (MSR Only)

Current State Antecedent		
Value	Name	Meaning
0x00	PU	Just Powered Up. The device has had no swipes and has not been Authenticated since it was powered up.
0x01	GoodAuth	Authentication Activation Successful. The host has sent the device a valid **Command 0x11 - Activation Challenge Response**.
0x02	GoodSwipe	Good Swipe. The cardholder swiped a valid card correctly.
0x03	BadSwipe	Bad Swipe. The cardholder swiped a card incorrectly or the card is not valid.
0x04	FailAuth	Authentication Activation Failed. The most recent **Command 0x11 - Activation Challenge Response** failed.
0x05	FailDeact	Authentication Deactivation Failed. A recent **Command 0x12 - Deactivate Authenticated Mode** failed.

Current State Antecedent		
0x06	TOAuth	Authentication Activation Timed Out. The host failed to send **Command 0x11 - Activation Challenge Response** in the time period specified by **Command 0x10 - Activate Authenticated Mode**.
0x07	TOSwipe	Swipe Timed Out. The cardholder failed to swipe a card in the time period specified in **Command 0x11 - Activation Challenge Response**.
0x08	Reserved	Reserved

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

7.3.10 Command 0x15 - Get / Set Security Level (MAC)

This command is used to set or get the device's current Security Level (see section **4 Security Levels**). The host can use this to raise the Security Level, but can not lower it.

When using this command to set the device's security level, the host should include the specified data in the request, and the device will not return an explicit response. When using this command to get the device's current security level, the host should include no data, and the device will return a response.

Table 7-17 - Request Data for Command 0x15 - Get / Set Security Level (MAC)

Offset	Field Name	Description
0	Security Level	Optional: if present must be either `0x03` or `0x04`. If absent, this is a query for the current Security Level.
1	MAC	Four byte MAC to secure the command [see section **4.1 About Message Authentication Codes (MAC)**]. If the host does not include a value for Security Level, it should not include the MAC value.

Table 7-18 - Response Data for Command 0x15 - Get / Set Security Level (MAC)

Offset	Field Name	Description
0	Security Level	Only present if there was no Data in the request. This value gives the current Security Level.

Result codes:
0x00 = Success
0x02 = Bad Parameters. The Data field in the request is not a correct length OR the specified Security Level is invalid; OR the current Security Level is 4.
0x07 = Incorrect MAC; command not authorized

Example Set Security Level Request (Hex)

Cmd Num	Data Len	Data
15	05	03 xx xx xx xx, where xx xx xx xx is a valid MAC

Example Set Security Level Response (Hex)

Result Code	Data Len	Data
00	00	

Example Get Request (Hex)

Cmd Num	Data Len	Data
15	00	

Example Get Security Level Response (Hex)

Result Code	Data Len	Data
00	01	03

7.3.11 Command 0x1C - Get Remaining MSR Transactions Counter (MSR Only)

This command is used to get the maximum number of remaining card swipe transactions or correctly completed Authentication sequences (**Command 0x10 - Activate Authenticated Mode** followed by a correct **Command 0x11 - Activation Challenge Response**) that the device can process. The value it returns is also sometimes referred to as the transaction threshold.

The value has three possible states:

- **Disabled** - value 0xFFFFFF - In this state there is no limit to the number of transactions that can be performed.
- **Expired** - value 0x000000 - This state indicates MSR transactions and Authentication commands are prohibited.
- **Active** - value 1 to 1,000,000 (0x000001 to 0x0F4240) - In this state, each transaction or successful Authentication sequence causes the value to be decremented and allows transactions to be processed. If an Authentication sequence decrements this value to 0, the device permits one final encrypted card swipe.

Some devices are configured to only allow the manufacturer to call this command.

Request Data: None

Table 7-19 - Response Data for Command 0x1C - Get Remaining MSR Transactions Counter (MSR Only)

Offset	Field Name	Description
0	Device Serial Number	16 bytes of device serial number. If the serial number is shorter than 15 bytes, this value is left-justified and padded with binary zeroes. At least one byte (usually the last one) must contain binary zero.
16	Remaining MSR Transactions	This three byte value contains the current number of remaining MSR transactions.

Result codes:
0x00 = Success
0x02 = Invalid length

Example Request (Hex)

Cmd Num	Data Len	Data
1C	00	

Example Response (Hex)

Result Code	Data Len	Data
00	13	544553542053455455502030303031000007F1 (2033 MSR Transactions Remaining)

8 Properties

8.1 About Properties

MagneSafe V5 devices have a number of programmable configuration properties stored in non-volatile memory. Most of the programmable properties pertain to data formats other than vendor-defined HID, but some of the properties deal with the device regardless of format (for information about changing formats and making format-specific properties visible, see **Property 0x10 - Interface Type**). These properties can be configured at the factory or by an administrator using software tools supplied by MagTek. Changing these configuration properties requires low-level communication with the device. Details for communicating with the device to read or change programmable properties are provided in section **7.3.1 Command 0x00 - Get Property** and section **7.3.2 Command 0x01 - Set Property (MAC)**.

8.2 Property 0x00 - Firmware ID

Property ID: `0x00`
Property Type: String
Length: Fixed at 11 bytes
Get Property: Yes
Set Property: No
Default Value: Part number of installed firmware

This is an 11 or 13 byte read-only property that identifies the firmware part number and revision installed on the device. The first 8 or 10 bytes represent the part number, the next byte represents the firmware major revision number, and the final two bytes represent an internal build number. For example, this property might be “21042812D01”.

(Embedded V5 Head Only)
For products that have a MagneSafe V5 IntelliHead embedded, two firmware IDs are available: One in the device itself, and one in the embedded IntelliHead. This property returns the values for the embedded IntelliHead. To get the values for the device, use **Property 0x3A - Firmware ID 2 (Embedded V5 Head Only)**.

Example Get Request (Hex)

Cmd Num	Data Len	Property ID
00	01	00

Example Get Response (Hex)

Result Code	Data Len	Property Value
00	0B	32 31 30 34 32 38 31 32 44 30 31

8.3 Property 0x03 - Device Serial Number

Property ID: `0x03`
Property Type: String
Length: 0 - 15 bytes
Get Property: Yes
Set Property: Yes (Once)
Default Value: Null string / ASCII device serial number set when the device is configured.

The property contains the device serial number, and is 0 to 15 bytes long. The device sends the value of this property (if any) to the host in the Device Serial Number field of **Magnetic Stripe Card Data Sent from Device to Host**. This property may be Set only once; attempts to Set the property again fail with RC = 0x07 (Sequence Error).

This property is stored in non-volatile memory, so it persists when the device is power cycled. When this property is changed, the device must be reset (see **Command 0x02 - Reset Device**) or powered off for at least 30 seconds, then powered on, before the changes will take effect.

Example Set Request (Hex)

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

Example Set Response (Hex)

Result Code	Data Len	Data
00	00	

Example Get Request (Hex)

Cmd Num	Data Len	Property ID
00	01	03

Example Get Response (Hex)

Result Code	Data Len	Property Value
00	03	31 32 33

8.4 Property 0x04 - MagneSafe Version Number

Property ID: `0x04`
Property Type: String
Length: 0 - 7 bytes
Get Property: Yes
Set Property: No
Default Value: “V05”

This is a maximum 7-byte read-only property that identifies the MagneSafe Feature Level supported on this device. Attempts to set this property fail with RC = 0x01.

Example Get Request (Hex)

Cmd Num	Data Len	Property ID
00	01	04

Example Get Response (Hex)

Result Code	Data Len	Property Value
00	03	56 30 35

8.5 Property 0x05 - Track ID Enable (MSR Only)

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

This property is defined as follows:

Bit Position	7	6	5	4	3	2	1	0
	id	0	T_3	T_3	T_2	T_2	T_1	T_1

id = 0: Decodes standard ISO/ABA cards only
id = 1: Decodes AAMVA and 7-bit cards also

If the id flag is set to 0, only tracks that conform to the ISO card data format allowed for that track are decoded. If the track cannot be decoded by the ISO method, the device reports a decode error.

For each pair of track bits, valid values are as follows:
$T_\#$ = 00: Track Disabled
$T_\#$ = 01: Track Enabled
$T_\#$ = 10: Track Enabled and Required (Error if blank)

This property is stored in non-volatile memory, so it persists when the device is power cycled. When this property is changed, the device must be reset (see **Command 0x02 - Reset Device**) or powered off for at least 30 seconds, then powered on, before the changes will take effect.

Example Set Request (Hex)

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

Example Set Response (Hex)

Result Code	Data Len	Data
00	00	

Example Get Request (Hex)

Cmd Num	Data Len	Property ID
00	01	05

Example Get Response (Hex)

Result Code	Data Len	Property Value
00	01	95

8.6 Property 0x07 - ISO Track Mask

Property ID: `0x07`
Property Type: String
Length: 6 bytes
Get Property: Yes
Set Property: Yes
Default Value: “04040Y”

This property specifies how the device should mask data on ISO/ABA type cards: Each byte in the sequence has the following meaning:

Offset	Description
0..1	These bytes are an ASCII representation of a decimal value that specifies how many of the leading characters of the PAN the device sends unmasked. The range is from “00” to “99”.
2..3	These bytes are an ASCII representation of a decimal value that specifies how many of the trailing characters of the PAN the device sends unmasked. The range is from “00” to “99”.
4	**Masking Character.** This byte specifies which character the device uses for masking. If this byte contains the uppercase letter ‘V’, the following rules apply: 1) The device masks the PAN using character ‘0’ 2) The device leaves all data after the PAN unmasked, leaving Discretionary Data (“DD”) and other non-PAN data available for the host to read.
5	This byte specifies whether the device applies Mod 10 Correction to the PAN. “Y” means Yes, “N” means No. This option is only effective if the Masking Character specified by this command is “0”.

This property is stored in non-volatile memory, so it persists when the device is power cycled. When this property is changed, the device must be reset (see **Command 0x02 - Reset Device**) or powered off for at least 30 seconds, then powered on, before the changes will take effect.

8.7 Property 0x08 - AAMVA Track Mask (MSR Only)

Property ID: `0x08`
Property Type: String
Length: 6 bytes
Get Property: Yes
Set Property: Yes
Default Value: `"04040Y"`

This property specifies the factors for masking data on AAMVA type cards. Each byte in the property has the following meaning:

Offset	Description
0..1	These bytes are an ASCII representation of a decimal value that specifies how many of the leading characters of the Driver's License/ID Number (DL/ID#) the device sends unmasked. The range is from "00" to "99".
2..3	These bytes are an ASCII representation of a decimal value that specifies how many of the trailing characters of the DL/ID# sends unmasked. The range is from "00" to "99".
4	**Masking Character.** This byte specifies which character the device uses for masking. If this byte contains the uppercase letter 'V', the following rules apply: • The device masks the PAN according to the rules of this property (**Property 0x34 - Send AAMVA Card Data** is ignored) • The device uses '0' for masking the PAN • The device sends all data after the PAN without masking
5	This byte specifies whether the device applies Mod 10 Correction to the DL/ID#. "Y" means Yes, "N" means No. This option is only effective if the masking character specified in this command is "0".

This property is stored in non-volatile memory, so it persists when the device is power cycled. When this property is changed, the device must be reset (see **Command 0x02 - Reset Device**) or powered off for at least 30 seconds, then powered on, before the changes will take effect.

8.8 Property 0x10 - Interface Type

Property ID: `0x10`
Property Type: Byte
Length: 1 byte
Get Property: Yes
Set Property: Yes (No for devices that switch connections automatically)
Default Value: 0x02

This property represents the device's current connection type (see section **2 Connection Types**) and data format (see section **3 Data Formats**):

- Valid values for this property are
  - `0x02` = iAP / RS-232 / UART (RS-232 Only | UART Only | iAP Only)
- On devices that have only one possible value for this property, the property is read-only.
- On devices that support multiple values for this property and do not handle connection switching automatically, the host can use this property to change the device's behavior. MagTek strongly recommends the host set this property before setting other properties, and immediately power cycle or reset the device (see **Command 0x02 - Reset Device)**, because it changes which other properties are available.
- (Bluetooth LE Only | iAP Only) This property only governs behavior of connections that are NOT Bluetooth LE or iAP (see section **1.4 About Connections and Data Formats**). Those connections are governed separately.

This property is stored in non-volatile memory, so it persists when the device is power cycled. When this property is changed, the device must be reset (see **Command 0x02 - Reset Device**) or powered off for at least 30 seconds, then powered on, before the changes will take effect.

Example Set Request (Hex)

Cmd Num	Data Len	Property ID	Property Value
01	02	10	00

Example Set Response (Hex)

Result Code	Data Len	Data
00	00	

Example Get Request (Hex)

Cmd Num	Data Len	Property ID
00	01	10

Example Get Response (Hex)

Result Code	Data Len	Property Value
00	01	00

8.9 Property 0x14 - Track Data Send Flags (KB Only | Streaming Only, MSR Only)

Property ID: `0x14`
Property Type: Byte
Length: 1 byte
Get Property: Yes
Set Property: Yes
Default Value: 0x63

The host uses this property to alter the formatting of **Magnetic Stripe Card Data Sent from Device to Host (MSR Only | Keypad Entry Only)** as follows:

Bit Position	7	6	5	4	3	2	1	0
	ICL	SS	ES	LRC	MKR	LC	Er	

Er = 00: The device does not send card data when a decode error occurs. Not currently implemented.
Er = 01: The device does not send track data when a decode error occurs.
Er = 11: Send the single character 'E' as the track data for each track with a decode error.

This property is stored in non-volatile memory, so it persists when the device is power cycled. When this property is changed, the device must be reset (see **Command 0x02 - Reset Device**) or powered off for at least 30 seconds, then powered on, before the changes will take effect.

8.9.1 Streaming Flags (Streaming Only)

SS = 0: Don't send Start Sentinel for each track.
SS = 1: Send Start Sentinel for each track.

ES = 0: Don't send End Sentinel for each track.
ES = 1: Send End Sentinel for each track.

The LRC is the unmodified LRC from the track data. If the host software needs to verify the LRC, it would need to restore the original Start Sentinels, then convert the track data from ASCII to card data format, and apply the LRC calculation algorithm appropriate for the card format (e.g., ISO or AAMVA). The LRC property only applies to track data sent in Streaming mode and completely in the clear (Security Level 2).

- LRC = 0: Don't send LRC for each track.
- LRC = 1: Send LRC for each track.

8.10 Property 0x15 - MagnePrint Flags (MSR Only)

NOTICE

(Streaming Only)
Using this command adds or suppresses fields in the data stream between the device and host, which changes the position of all subsequent data elements in the stream. This may render the device incompatible with host software that expects to parse a fixed format, rather than using Property 0x2C - Format Code (Streaming, MSR Only) **to determine the position of data in the stream.**

Property ID: `0x15`
Property Type: Byte
Length: 1 byte
Get Property: Yes
Set Property: Yes
Default Value: 0x00

The host uses this property to direct the device to either include or exclude MagnePrint data in **Magnetic Stripe Card Data Sent from Device to Host (MSR Only | Keypad Entry Only)** when the device is in **Security Level 2**. At higher security levels, the device always sends encrypted MagnePrint data.

Bit Position	7	6	5	4	3	2	1	0
	0	0	0	0	0	0	0	S

S = 0: Device does not include MagnePrint Data
S = 1: Device includes MagnePrint Data

Setting S to 1 directs the device to send **MagnePrint Status** and **Encrypted MagnePrint Data** with each swipe when it is in **Security Level 2**.

Setting S to 0 directs the device to zero-fill these values. (Streaming Only) When the device is using Streaming format, it omits these values altogether, along with the programmable field separators that precede each value.

Some devices are configured to only allow the manufacturer to modify this property.

This property is stored in non-volatile memory, so it persists when the device is power cycled. When this property is changed, the device must be reset (see **Command 0x02 - Reset Device**) or powered off for at least 30 seconds, then powered on, before the changes will take effect.

8.11 Property 0x19 - CRC Flags (Streaming Only, MSR Only)

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

This property specifies the behavior of the values **Clear Text CRC (Streaming Only)** and **Encrypted CRC (Streaming Only)** within **Magnetic Stripe Card Data Sent from Device to Host (MSR Only | Keypad Entry Only)**.

Bit Position	7	6	5	4	3	2	1	0
	0	0	0	0	0	0	E	S

E = 0: Device does NOT send Encrypted CRC.
E = 1: Device sends the Encrypted CRC.

S = 0: Device does NOT send the Clear Text CRC.
S = 1: Device sends the Clear Text CRC.

With the default setting `0x01`, the device sends only the Clear Text CRC. If both E and S are set to 0, the device still sends the Programmable Field Separator that precedes each of these fields.

This property is stored in non-volatile memory, so it persists when the device is power cycled. When this property is changed, the device must be reset (see **Command 0x02 - Reset Device**) or powered off for at least 30 seconds, then powered on, before the changes will take effect.

8.12 Property 0x1E - Pre Card String (Streaming Only, MSR Only)

Property ID: `0x1E`
Property Type: String
Length: 0 - 7 bytes
Get Property: Yes
Set Property: Yes
Default Value: Null string

The device sends the value of this property to the host before all other card data. For example, if the host software requires a set of keystrokes to begin the process of receiving card data, this property could be set to transmit that keystroke sequence.

This property is stored in non-volatile memory, so it persists when the device is power cycled. When this property is changed, the device must be reset (see **Command 0x02 - Reset Device**) or powered off for at least 30 seconds, then powered on, before the changes will take effect.

Example Set Request (Hex)

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

Example Set Response (Hex)

Result Code	Data Len	Data
00	00	

Example Get Request (Hex)

Cmd Num	Data Len	Property ID
00	01	1E

Example Get Response (Hex)

Result Code	Data Len	Property Value
00	03	31 32 33

8.13 Property 0x1F - Post Card String (Streaming Only, MSR Only)

Property ID: `0x1F`
Property Type: String
Length: 0 - 7 bytes
Get Property: Yes
Set Property: Yes
Default Value: Null string.

The device sends the host the value of this property after all other card data.

This property is stored in non-volatile memory, so it persists when the device is power cycled. When this property is changed, the device must be reset (see **Command 0x02 - Reset Device**) or powered off for at least 30 seconds, then powered on, before the changes will take effect.

Example Set Request (Hex)

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

Example Set Response (Hex)

Result Code	Data Len	Data
00	00	

Example Get Request (Hex)

Cmd Num	Data Len	Property ID
00	01	1F

Example Get Response (Hex)

Result Code	Data Len	Property Value
00	03	31 32 33

8.14 Property 0x20 - Pre Track String (Streaming Only, MSR Only)

Property ID: `0x20`
Property Type: String
Length: 0-7 bytes
Get Property: Yes
Set Property: Yes
Default Value: Null string

The device sends the host the value of this property before the data for each track. If the value is 0, the device does not send a pre-track string.

This property is stored in non-volatile memory, so it persists when the device is power cycled. When this property is changed, the device must be reset (see **Command 0x02 - Reset Device**) or powered off for at least 30 seconds, then powered on, before the changes will take effect.

Example Set Request (Hex)

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

Example Set Response (Hex)

Result Code	Data Len	Data
00	00	

Example Get Request (Hex)

Cmd Num	Data Len	Property ID
00	01	20

Example Get Response (Hex)

Result Code	Data Len	Property Value
00	03	31 32 33

8.15 Property 0x21 - Post Track String (Streaming Only, MSR Only)

Property ID: `0x21`
Property Type: String
Length: 0-7 bytes
Get Property: Yes
Set Property: Yes
Default Value: Null string.

The device sends the host the value of this property after the data for each track. If the value is 0, the device does not send a pre-track string.

This property is stored in non-volatile memory, so it persists when the device is power cycled. When this property is changed, the device must be reset (see **Command 0x02 - Reset Device**) or powered off for at least 30 seconds, then powered on, before the changes will take effect.

Example Set Request (Hex)

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

Example Set Response (Hex)

Result Code	Data Len	Data
00	00	

Example Get Request (Hex)

Cmd Num	Data Len	Property ID
00	01	21

Example Get Response (Hex)

Result Code	Data Len	Property Value
00	03	31 32 33

8.16 Property 0x22 - Termination String (Streaming Only, MSR Only)

Property ID: `0x22`
Property Type: String
Length: 0-7 bytes
Get Property: Yes
Set Property: Yes
Default Value: 0x0D (carriage return)

The device sends the host the value of this property after the all the data for a transaction. If the value is 0, the device does not send a termination string.

This property is stored in non-volatile memory, so it persists when the device is power cycled. When this property is changed, the device must be reset (see **Command 0x02 - Reset Device**) or powered off for at least 30 seconds, then powered on, before the changes will take effect.

Example Set Request (Hex)

Cmd Num	Data Len	Property ID	Property Value
01	02	22	0D

Example Set Response (Hex)

Result Code	Data Len	Data
00	00	

Example Get Request (Hex)

Cmd Num	Data Len	Property ID
00	01	22

Example Get Response (Hex)

Result Code	Data Len	Property Value
00	01	0D

8.17 Property 0x23 - Field Separator (Streaming Only, MSR Only)

Property ID: `0x23`
Property Type: Byte
Length: 1 byte
Get Property: Yes
Set Property: Yes
Default Value: 0x7C (‘|’)

This property stores the character the device uses for P35 (see section **3.1 How to Use Streaming Format**). If the value is in the range 1 - 127, the device sends the equivalent ASCII character. If the value is 0, the device does not send a delimiter, which is inadvisable.

This property is stored in non-volatile memory, so it persists when the device is power cycled. When this property is changed, the device must be reset (see **Command 0x02 - Reset Device**) or powered off for at least 30 seconds, then powered on, before the changes will take effect.

Example Set Request (Hex)

Cmd Num	Data Len	Property ID	Property Value
01	02	23	7C

Example Set Response (Hex)

Result Code	Data Len	Data
00	00	

Example Get Request (Hex)

Cmd Num	Data Len	Property ID
00	01	23

Example Get Response (Hex)

Result Code	Data Len	Property Value
00	01	7C

8.18 Property 0x24 - Start Sentinel Track 1 ISO ABA (Streaming Only, MSR Only | Keypad Entry Only)

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

This property stores the character the device uses to replace the Track 1 Start Sentinel for cards where it recognizes Track 1 encoded in ISO/ABA format. The default value is the standard ISO/ABA Track 1 Start Sentinel, meaning no replacement. If the value is in the range 1 - 127, the device sends the equivalent ASCII character. If the value is 0, the device does not send a character.

This property is stored in non-volatile memory, so it persists when the device is power cycled. When this property is changed, the device must be reset (see **Command 0x02 - Reset Device**) or powered off for at least 30 seconds, then powered on, before the changes will take effect.

8.19 Property 0x25 - Start Sentinel Track 2 ISO ABA (Streaming Only, MSR Only)

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

This property stores the character the device uses to replace the Track 2 Start Sentinel for cards where it recognizes Track 2 encoded in ISO/ABA format. The default value is the standard ISO/ABA Track 2 Start Sentinel, meaning no replacement. If the value is in the range 1 - 127, the device sends the equivalent ASCII character. If the value is 0, the device does not send a character.

This property is stored in non-volatile memory, so it persists when the device is power cycled. When this property is changed, the device must be reset (see **Command 0x02 - Reset Device**) or powered off for at least 30 seconds, then powered on, before the changes will take effect.

8.20 Property 0x26 - Start Sentinel Track 3 ISO ABA (Streaming Only, MSR Only)

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

This property stores the character the device uses as a Track 3 Start Sentinel for cards where it recognizes Track 3 encoded in ISO/ABA format. If the value is in the range 1 - 127, the device sends the equivalent ASCII character. If the value is 0, the device does not send a character.

This property is stored in non-volatile memory, so it persists when the device is power cycled. When this property is changed, the device must be reset (see **Command 0x02 - Reset Device**) or powered off for at least 30 seconds, then powered on, before the changes will take effect.

8.21 Property 0x27 - Start Sentinel Track 3 AAMVA (Streaming Only, MSR Only)

Property ID: `0x27`
Property Type: Byte
Length: 1 byte
Get Property: Yes
Set Property: Yes
Default Value: 0x23 (‘#’)

This property stores the character the device uses as a Track 3 Start Sentinel for cards where it recognizes Track 3 encoded in AAMVA format. If the value is in the range 1 - 127, the device sends the equivalent ASCII character. If the value is 0, the device does not send a character.

This property is stored in non-volatile memory, so it persists when the device is power cycled. When this property is changed, the device must be reset (see **Command 0x02 - Reset Device**) or powered off for at least 30 seconds, then powered on, before the changes will take effect.

8.22 Property 0x28 - Start Sentinel Track 2 7bits (Streaming Only, MSR Only)

Property ID: `0x28`
Property Type: Byte
Length: 1 byte
Get Property: Yes
Set Property: Yes
Default Value: 0x40 (‘@’)

This property stores the character the device uses to replace the Track 2 Start Sentinel for cards where it recognizes Track 2 encoded in the 7-bit ISO format used for Track 1. If the value is in the range 1 - 127, the device sends the equivalent ASCII character. If the value is 0, the device does not send a character.

This property is stored in non-volatile memory, so it persists when the device is power cycled. When this property is changed, the device must be reset (see **Command 0x02 - Reset Device**) or powered off for at least 30 seconds, then powered on, before the changes will take effect.

8.23 Property 0x29 - Start Sentinel Track 3 7bits (Streaming Only, MSR Only, 3-Track Only)

Property ID: `0x29`
Property Type: Byte
Length: 1 byte
Get Property: Yes
Set Property: Yes
Default Value: 0x26 (‘&’)

This property stores the character the device uses to replace the Track 3 Start Sentinel for cards where it recognizes Track 3 encoded in the 7-bit ISO format used for Track 1. If the value is in the range 1 - 127, the device sends the equivalent ASCII character. If the value is 0, the device does not send a character.

This property is stored in non-volatile memory, so it persists when the device is power cycled. When this property is changed, the device must be reset (see **Command 0x02 - Reset Device**) or powered off for at least 30 seconds, then powered on, before the changes will take effect.

8.24 Property 0x2B - End Sentinel (Streaming Only, MSR Only)

Property ID: `0x2B`
Property Type: Byte
Length: 1 byte
Get Property: Yes
Set Property: Yes
Default Value: 0x3F ('?')

This property stores the character the device sends as the End Sentinel for all tracks in any card data format [unless it is overridden, track by track, by **Property 0x2D - End Sentinel Track 1 (Streaming Only, MSR Only)**, **Property 0x2E - End Sentinel Track 2 (Streaming Only, MSR Only)**, or **Property 0x2F - End Sentinel Track 3 (Streaming Only, MSR Only, 3-Track Only)**]. In tracks that have a standard end sentinel embedded, it replaces them. If the value is in the range 1 - 127, the device sends the equivalent ASCII character. If the value is 0, the device does not send a character.

This property is stored in non-volatile memory, so it persists when the device is power cycled. When this property is changed, the device must be reset (see **Command 0x02 - Reset Device**) or powered off for at least 30 seconds, then powered on, before the changes will take effect.

8.25 Property 0x2C - Format Code (Streaming Only, MSR Only)

Property ID: `0x2C`
Property Type: String
Length: 4 bytes
Get Property: Yes
Set Property: Yes
Default Value: "0000"

This property specifies the Format Code the device includes when it sends card swipe data to the host [see section **6.13 Format Code (Streaming Only)**], and is designed to allow programmers of host software to populate the final three characters as "notes" the host software can use to determine how to parse or interpret the accompanying data. When setting this property, the host must send four characters: The device ignores the first character, which is reserved for MagTek/device use, and changes the final three characters of the property to the final three characters the host specified.

The value can be interpreted as follows:

- By default, the Format Code is "0000".
- If the manufacturer configures **Property 0x30 - Send Remaining MSR Transactions Counter (Streaming Only, MSR Only)** to send the MSR Transactions Remaining counter, the device changes the Format Code from "0000" to "0001."
- If the host directly sets the value of the Format Code property, the new value overrides the factory set value. The host can change the final three characters, but making such a change automatically causes the first character to change to "1".
- If an administrator or host software changes a setting that automatically updates Format Code, the first character of the Format Code changes to a "1". Such settings include:
    - **Property 0x15 - MagnePrint Flags (MSR Only)**

- **Property 0x19 - CRC Flags (Streaming Only, MSR Only)**
- **Property 0x1E - Pre Card String (Streaming Only, MSR Only)**
- **Property 0x1F - Post Card String (Streaming Only, MSR Only)**
- **Property 0x20 - Pre Track String (Streaming Only, MSR Only)**
- **Property 0x21 - Post Track String (Streaming Only, MSR Only)**
- **Property 0x22 - Termination String (Streaming Only, MSR Only)**
- **Property 0x23 - Field Separator (Streaming Only, MSR Only)**
- **Property 0x24 - Start Sentinel Track 1 ISO ABA (Streaming Only, MSR Only | Keypad Entry Only)**
- **Property 0x25 - Start Sentinel Track 2 ISO ABA (Streaming Only, MSR Only)**
- **Property 0x26 - Start Sentinel Track 3 ISO ABA (Streaming Only, MSR Only)**
- **Property 0x27 - Start Sentinel Track 3 AAMVA (Streaming Only, MSR Only)**
- **Property 0x28 - Start Sentinel Track 2 7bits (Streaming Only, MSR Only)**
- **Property 0x29 - Start Sentinel Track 3 7bits (Streaming Only, MSR Only, 3-Track Only)**
- **Property 0x2B - End Sentinel (Streaming Only, MSR Only)**

This property is stored in non-volatile memory, so it persists when the device is power cycled. When this property is changed, the device must be reset (see **Command 0x02 - Reset Device**) or powered off for at least 30 seconds, then powered on, before the changes will take effect.

8.26 Property 0x2D - End Sentinel Track 1 (Streaming Only, MSR Only)

Property ID:	0x2D
Property Type:	Byte
Length:	1 byte
Get Property:	Yes
Set Property:	Yes
Default Value:	0xFF

This property specifies the character the device sends as the end sentinel for Track 1 with any card format. In tracks that have standard end sentinels embedded, it replaces them. If the value is in the range 1 - 127, the device sends the equivalent ASCII character. If the value is 0xFF, the device uses the value of **Property 0x2B - End Sentinel** instead. If the value is 0, the device does not send a character.

This property is stored in non-volatile memory, so it persists when the device is power cycled. When this property is changed, the device must be reset (see **Command 0x02 - Reset Device**) or powered off for at least 30 seconds, then powered on, before the changes will take effect.

8.27 Property 0x2E - End Sentinel Track 2 (Streaming Only, MSR Only)

Property ID:	0x2E
Property Type:	Byte
Length:	1 byte
Get Property:	Yes
Set Property:	Yes
Default Value:	0xFF

This property specifies the character the device sends as the end sentinel for Track 2 with any card format. In tracks that have standard end sentinels embedded, it replaces them. If the value is in the range 1 - 127, the device sends the equivalent ASCII character. If the value is `0xFF`, the device uses the value of **Property 0x2B - End Sentinel** instead. If the value is 0, the device does not send a character.

This property is stored in non-volatile memory, so it persists when the device is power cycled. When this property is changed, the device must be reset (see **Command 0x02 - Reset Device**) or powered off for at least 30 seconds, then powered on, before the changes will take effect.

8.28 Property 0x2F - End Sentinel Track 3 (Streaming Only, MSR Only, 3-Track Only)

Property ID: `0x2F`
Property Type: Byte
Length: 1 byte
Get Property: Yes
Set Property: Yes
Default Value: 0xFF

This property specifies the character the device sends as the end sentinel for Track 3 with any card format. In tracks that have standard end sentinels embedded, it replaces them. If the value is in the range 1 - 127, the device sends the equivalent ASCII character. If the value is `0xFF`, the device uses the value of **Property 0x2B - End Sentinel** instead. If the value is 0, the device does not send a character.

This property is stored in non-volatile memory, so it persists when the device is power cycled. When this property is changed, the device must be reset (see **Command 0x02 - Reset Device**) or powered off for at least 30 seconds, then powered on, before the changes will take effect.

8.29 Property 0x30 - Send Remaining MSR Transactions Counter (Streaming Only, MSR Only)

Property ID: `0x30`
Property Type: Byte
Length: 1 byte
Get Property: Yes
Set Property: Yes
Default Value: 0x00 (Don't send Remaining MSR Transactions counter)

This property specifies whether device sends the Remaining MSR Transactions counter (sometimes known as the transaction threshold, see **Command 0x1C - Get Remaining MSR Transactions Counter (MSR Only)** for details) as part of a Streaming message. If the property is set to `0x00`, the device sends neither the remaining MSR transactions counter nor the field separator. If the property is set to `0x01`, the device sends the remaining MSR transactions counter immediately following the **DUKPT Key Serial Number (KSN)** in a swipe message.

If this property is set to 0x01 and **Property 0x2C - Format Code (Streaming Only, MSR Only)** is currently "0001", the device changes **Property 0x2C - Format Code (Streaming Only, MSR Only)** to "0002".

This property is stored in non-volatile memory, so it persists when the device is power cycled. When this property is changed, the device must be reset (see **Command 0x02 - Reset Device**) or powered off for at least 30 seconds, then powered on, before the changes will take effect.

8.30 Property 0x31 - Mask Other Cards (MSR Only)

Property ID: 0x31
Property Type: Byte
Length: 1 byte
Get Property: Yes
Set Property: Yes
Default Value: 0x00 (Don't Mask Other cards)

This property designates whether cards which do not decode as either ISO/ABA (Financial) or AAMVA (Driver License) format should be sent with their data masked or unmasked. The default value (0x00) is to send the data unmasked. If this property is set to 0x01, the device sends the track(s) to the host using a "0" for each byte of track data the device reads from the card.

If a card is encoded according to ISO/ABA rules (Track 1 in 7 bit format, Tracks 2 and Track 3 in 5 bit format), and Track 1 does not begin with the character 'B', the device always sends the **Track 1 Masked Data** value unmasked, regardless of the value of this property. See **Appendix E** for details.

This property is stored in non-volatile memory, so it persists when the device is power cycled. When this property is changed, the device must be reset (see **Command 0x02 - Reset Device**) or powered off for at least 30 seconds, then powered on, before the changes will take effect.

8.31 Property 0x34 - Send AAMVA Card Data Unmasked (MSR Only)

Property ID: 0x34
Property Type: Byte
Length: 1 byte
Get Property: Yes
Set Property: Yes
Default Value: 0x00

This property controls how the device sends AAMVA card data when the security level is higher than **Security Level 2**:

- 0 = Send masked AAMVA card data.
- 1 = Send clear AAMVA card data.

This property is stored in non-volatile memory, so it persists when the device is power cycled. When this property is changed, the device must be reset (see **Command 0x02 - Reset Device**) or powered off for at least 30 seconds, then powered on, before the changes will take effect.

Example Set Request (Hex)

Cmd Num	Data Len	Property ID	Data
01	02	34	01

Example Set Response (Hex)

Result Code	Data Len	Data
00	00	

Example Get Request (Hex)

Cmd Num	Data Len	Property ID
00	01	34

Example Get Response (Hex)

Result Code	Data Len	Data
00	01	01

8.32 Property 0x3A - Firmware ID 2 (Embedded V5 Head Only)

Property ID: `0x3A`
Property Type: String
Length: Fixed at 11 bytes
Get Property: Yes
Set Property: No
Default Value: Part number, major revision number, and build number of installed firmware

This 11-byte or 13 byte read-only property returns the part number, major revision number, and build number of secondary firmware installed on the device. The first 8 or 10 bytes represent the firmware part number, the next byte represents the firmware major revision number, and the final two bytes represent a firmware internal build number. For example, this property might be “21042812D01” where 21042812 is the part number, D is the major revision number, and 01 is the internal build number. To get the device’s primary firmware ID, use **Property 0x00 - Firmware ID**.

(Embedded V5 Head Only)
For devices with embedded MagneSafe V5 IntelliHeads, two firmware IDs are available: One for the device, and one for the embedded IntelliHead. This property returns values for the device.

Example Get Request (Hex)

Cmd Num	Data Len	Property ID
00	01	3A

Example Get Response (Hex)

Result Code	Data Len	Property Value
00	0B	32 31 30 34 32 38 31 32 44 30 31

8.33 Property 0x54 - Card Data Encryption Variant (MSR Only, Configurable MSR Variants Only)

Property ID: `0x54`
Property Type: Byte
Length: 1 byte
Get Property: Yes
Set Property: Yes
Default Value: 0x00 (PIN Variant)

This property specifies which variant of the current DUKPT Key the device uses to encrypt magnetic stripe **Track 1 Encrypted Data**, **Track 2 Encrypted Data**, **Track 3 Encrypted Data**, and **Encrypted Session ID**:

- 0x00 = Use **PIN Encryption** variant
- 0x01 = Use **Data Encryption, request or both ways** variant

The host software should use this value to determine how to create the correct Derived Key to decrypt **Encrypted Track Data** (see section **5 Encryption, Decryption, and Key Management**). The algorithms for creating the Derived Key fitting each of the possible variants are fully specified in ***ANS X9.24-1:2009***.

Some devices are configured to only allow the manufacturer to modify this property.

This property is stored in non-volatile memory, so it persists when the device is power cycled. When this property is changed, the device must be reset (see **Command 0x02 - Reset Device**) or powered off for at least 30 seconds, then powered on, before the changes will take effect.

Example Set Request (Hex)

Cmd Num	Data Len	Property ID	Data
01	02	54	01

Example Set Response (Hex)

Result Code	Data Len	Data
00	00	

Example Get Request (Hex)

Cmd Num	Data Len	Property ID
00	01	54

Example Get Response (Hex)

Result Code	Data Len	Data
00	01	01

Appendix A Examples

This section includes direct command examples and information about using demonstration software. In addition to the examples here, source code with detailed comments is included with the demo software and can be used as a guide for custom software development.

The book ***USB Complete*** by Jan Axelson is also a good guide for software developers, especially the chapter “Human Interface Devices: Host Applications.”

A.1 Command Examples

This section provides examples of command sequences and cryptographic operations. Each example shows a sequence as it actually runs, so developers of custom software can check their code against the examples step-by-step to make sure the software is parsing and computing values correctly.

A.1.1 Example: Streaming Card Swipe In Security Level 2, Not SureSwipe (Streaming Only, MSR Only)

This example shows how to interpret card data received on a device set to **Security Level 2** transmitting in streaming format (see section **3.1 How to Use Streaming Format**).

The incoming streaming data is:

```
Byte    Content
  0     %B5452000000007189^HOGAN/PAUL      ^08040000000000
 50     000000000?;5452000000007189=080400000000000000?+51
100     63000050000445=000000000000?|0200|%B54523005512271
150     89^HOGAN/PAUL      ^08043210000007250000000?|;5452
200     300551227189=080432100000007250?|+5163499080020445
250     =000000000000?||||0000000000000000||6F36||1000
```

The information in section **3.1 How to Use Streaming Format (Streaming Only)** provides a basic template showing the expected order of fields in the data:

```
[P0x1E]
[P0x20] [Tk1 SS] [Tk1 Masked Data] [ES] [P0x21]
[P0x20] [Tk2 SS] [Tk2 Masked Data] [ES] [P0x21]
[P0x20] [Tk3 SS] [Tk3 Masked Data] [ES] [P0x21]
[P0x1F]
[P0x23] [Device Encryption Status]
[P0x23] [Tk1 Encrypted Data (including TK1 SS and ES)]
[P0x23] [Tk2 Encrypted Data (including TK2 SS and ES)]
[P0x23] [Tk3 Encrypted Data (including TK3 SS and ES)]
[P0x23] [MagnePrint Status]
[P0x23] [Encrypted MagnePrint data]
[P0x23] [Device serial number]
[P0x23] [Encrypted Session ID]
[P0x23] [DUKPT Key Serial Number (KSN) / counter]
[P0x23] [Clear Text CRC]
[P0x23] [Encrypted CRC]
[P0x23] [Format Code]
[P0x22]
```

Each of the Pxx elements has the default value in this configuration, so this can be re-interpreted as:

```
%[Tk1 Masked Data]?
;[Tk2 Masked Data]?
+[Tk3 Masked Data]?
|[Device Encryption Status]
|[Tk1 Encrypted Data (including TK1 SS and ES)]
|[Tk2 Encrypted Data (including TK2 SS and ES)]
|[Tk3 Encrypted Data (including TK3 SS and ES)]
|[MagnePrint Status]
|[Encrypted MagnePrint data]
|[Device serial number]
|[Encrypted Session ID]
```

```
|[DUKPT Key Serial Number (KSN) / counter]
|[Clear Text CRC]
|[Encrypted CRC]
|[Format Code]
<ENTER>
```

Using the above as a template and filling in with the received raw swipe data yields the following:

```
%B5452000000007189^HOGAN/PAUL      ^08040000000000000000000?
;5452000000007189=080400000000000000?
+5163000050000445=000000000000?
|0200
|%B5452300551227189^HOGAN/PAUL      ^08043210000000725000000?
|;5452300551227189=080432100000007250?
|+5163499080020445=000000000000?
|
|
|
|0000000000000000
|
|6F36
|
|1000
```

The Device Serial Number value is empty because the DSN has not been set.

The MagnePrint Status, the MagnePrint Data, the DUKPT Key Serial Number (KSN) / counter and Encrypted CRC values are empty because this device is at Security Level 2 (encryption not enabled).

When the device is set to Security Level 2, the following values are represented as ASCII characters:

- Masked Track data
- Encrypted Track data
- Format Code

All other values are represented as hexadecimal data (two ASCII characters together specify the value of a single byte).

A.1.2 Example: Swipe Decryption, Streaming Mode, Device In Security Level 3 or 4 (Streaming Only, MSR Only)

This example shows the data received in streaming format for a device using streaming format that is set to **Security Level 3** or **Security Level 4**, with KSN Count = 8 (see **Command 0x09 - Get Current TDES DUKPT KSN**). It includes steps that show how to decrypt the incoming data.

The incoming streaming data is:

```
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
```

The expected order of fields in the data:

```
[P0x1E]
[P0x20] [Tk1 SS] [Tk1 Masked Data] [ES] [P0x21]
[P0x20] [Tk2 SS] [Tk2 Masked Data] [ES] [P0x21]
[P0x20] [Tk3 SS] [Tk3 Masked Data] [ES] [P0x21]
[P0x1F]
[P0x23] [Device Encryption Status]
[P0x23] [Tk1 SS] [Tk1 Encrypted Data] [ES]
[P0x23] [Tk1 SS] [Tk2 Encrypted Data] [ES]
[P0x23] [Tk1 SS] [Tk3 Encrypted Data] [ES]
[P0x23] [MagnePrint Status]
[P0x23] [Encrypted MagnePrint data]
[P0x23] [Device serial number]
[P0x23] [Encrypted Session ID]
[P0x23] [DUKPT serial number/counter]
[P0x23] [Clear Text CRC]
[P0x23] [Encrypted CRC]
[P0x23] [Format Code]
[P0x22]
```

The device has the default configuration for each of the Pxx elements, so the host software can interpret the format above as:

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

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

Using the above as a template and filling in with the received raw swipe data yields the following data:

```
%B5452000000007189^HOGAN/PAUL      ^08040000000000000000000?
;5452000000007189=08040000000000000?
+5163000050000445=000000000000?
|0600
|C25C1D1197D31CAA87285D59A892047426D9182EC11353C051ADD6D0F072A6CB34365
60B3071FC1FD11D9F7E74886742D9BEE0CFD1EA1064C213BB55278B2F12
|724C5DB7D6F901C7F0FEAE7908801093B3DBFE51CCF6D483E789D7D2C007D539499BA
ADCC8D16CA2
|E31234A91059A0FBFE627954EE21868AEE3979540B67FCC40F61CECA54152D1E
|A1050000
|8628E664C59BBAA232BA90BFB3E6B41D6F4B691E633C311CBE6EE7466B81196EC07B1
2648DCAC4FD7FD0E212B479C60BAD8C74F82F327667
|
|21685F158B5C6BE0
|FFFF9876543210E00008
|B78F
|
|0000
```

The Device Serial Number value is empty because the DSN has not been set.

The Encrypted CRC value is empty because the default configuration is to send it empty.

At Security Level 3, these values are represented as ASCII characters:

- Masked Track data
- Format Code

All other values are represented as hexadecimal data (two ASCII characters together specify the value of a single byte).

To decrypt this data, the host software would first examine the KSN field `FFFF9876543210E00008`, and break it down into base key `FFFF9876543210E` and the key counter is `0x00008` (see section **6.9 DUKPT Key Serial Number** for details). The host would use this information to calculate encryption key 27F66D5244FF621E AA6F6120EDEB427F, which is also provided in the ANSI standard documentation’s examples for convenience.

There are five encrypted values: Track 1 encrypted data, track 2 encrypted data, track 3 encrypted data, encrypted MagnePrint data, and encrypted session ID. The remainder of this section details the procedure for decrypting these data values.

The track 1 encrypted data is:

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

Section **5 Encryption, Decryption, and Key Management** tells us to decrypt the last block first: C213BB55278B2F12 TDES Decrypt with 27F66D5244FF621E AA6F6120EDEB427F gets E98ED0F0D1EA1064, XOR D9BEE0CFD1EA1064 gets 3030303F00000000, which is the decrypted last block.

Continuing in reverse block order, D9BEE0CFD1EA1064 TDES Decrypt with 27F66D5244FF621E AA6F6120EDEB427F gets E12DA84C41B85772, XOR D11D9F7E74886742 gets 3030373235303030, which is decrypted block 7.

Continuing in reverse block order, D11D9F7E74886742 TDES Decrypt with 27F66D5244FF621E AA6F6120EDEB427F gets 0704673B0041CC2F, XOR 3436560B3071FC1F gets 3332313030303030, which is decrypted block 6.

Continuing in reverse block order, 3436560B3071FC1F TDES Decrypt with 27F66D5244FF621E AA6F6120EDEB427F gets 718DF68EC04A96FF, XOR 51ADD6D0F072A6CB gets 2020205E30383034, which is decrypted block 5.

Continuing in reverse block order, 51ADD6D0F072A6CB TDES Decrypt with 27F66D5244FF621E AA6F6120EDEB427F gets 0989597B8D3373E0, XOR 26D9182EC11353C0 gets 2F5041554C202020, which is decrypted block 4.

Continuing in reverse block order, 26D9182EC11353C0 TDES Decrypt with 27F66D5244FF621E AA6F6120EDEB427F gets BF110311E7D5453A, XOR 87285D59A8920474 gets 38395E484F47414E, which is decrypted block 3.

Continuing in reverse block order, 87285D59A8920474 TDES Decrypt with 27F66D5244FF621E AA6F6120EDEB427F gets F2692820A5E12B9B, XOR C25C1D1197D31CAA gets 3035353132323731, which is decrypted block 2.

Continuing in reverse block order, C25C1D1197D31CAA TDES Decrypt with 27F66D5244FF621E AA6F6120EDEB427F gets 2542353435323330, which is decrypted block 1.

The host software can ignore the last four bytes because they are all hex 0x00, and are located after the End Sentinel. Ordering the decrypted blocks first to last while ignoring the null padding at the end yields:

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

The resulting ASCII string is:

```
%B5452300551227189^HOGAN/PAUL      ^08043210000000725000000?
```

The track 2 encrypted data is:

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

Section **5 Encryption, Decryption, and Key Management** tells us to decrypt the last block first: `499BAADCC8D16CA2` TDES Decrypt with `27F66D5244FF621E AA6F6120EDEB427F` gets `D0BBE2E2FF07D539`, XOR `E789D7D2C007D539` gets `373235303F000000`, which is the decrypted last block.

Continuing in reverse block order, `E789D7D2C007D539` TDES Decrypt with `27F66D5244FF621E AA6F6120EDEB427F` gets `82EBCE61FCC6E4B3`, XOR `B3DBFE51CCF6D483` gets `3130303030303030`, which is decrypted block 4.

Continuing in reverse block order, `B3DBFE51CCF6D483` TDES Decrypt with `27F66D5244FF621E AA6F6120EDEB427F` gets `C9C39E4138B423A1`, XOR `F0FEAE7908801093` gets `393D303830343332`, which is decrypted block 3.

Continuing in reverse block order, `F0FEAE7908801093` TDES Decrypt with `27F66D5244FF621E AA6F6120EDEB427F` gets `47796C85E4CE30FF`, XOR `724C5DB7D6F901C7` gets `3535313232373138`, which is decrypted block 2.

Continuing in reverse block order, `724C5DB7D6F901C7` TDES Decrypt with `27F66D5244FF621E AA6F6120EDEB427F` gets `3B35343532333030`, which is decrypted block 1.

The host software can ignore the last four bytes because they are all hex 0x00, and are located after the End Sentinel. Ordering the decrypted blocks first to last while ignoring the null padding at the end yields:

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

The resulting ASCII string for track 2 is:

```
;5452300551227189=080432100000007250?
```

The track 3 encrypted data is:

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

Section **5 Encryption, Decryption, and Key Management** tells us to decrypt the last block first: 0F61CECA54152D1E TDES Decrypt with 27F66D5244FF621E AA6F6120EDEB427F gets DE0949643B57C3C4, XOR EE3979540B67FCC4 gets 3030303030303F00, which is the decrypted last block.

Continuing in reverse block order, EE3979540B67FCC4 TDES Decrypt with 27F66D5244FF621E AA6F6120EDEB427F gets CB5F4964DE11B6BA, XOR FE627954EE21868A gets 353D303030303030, which is decrypted block 3.

Continuing in reverse block order, FE627954EE21868A TDES Decrypt with 27F66D5244FF621E AA6F6120EDEB427F gets D32A0499226994CF, XOR E31234A91059A0FB gets 3038303032303434, which is decrypted block 2.

Continuing in reverse block order, E31234A91059A0FB TDES Decrypt with 27F66D5244FF621E AA6F6120EDEB427F gets 2B35313633343939, which is decrypted block 1.

Ordering the decrypted blocks first to last gives:

```
HEX                   ASCII
2B35313633343939      +5163499
3038303032303434      08002044
353D303030303030      3=000000
3030303030303F00      000000?
```

The host software can ignore the last byte because it is hex 0x00 and is located after the End Sentinel. The resulting ASCII string for track 3 is:

```
+5163499080020443=000000000000?
```

The MagnePrint data is:

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

Section **5 Encryption, Decryption, and Key Management** tells us to decrypt the last block first: `AD8C74F82F327667` TDES Decrypt with `27F66D5244FF621E AA6F6120EDEB427F` gets `09162DCA11E5C60B`, XOR `7FD0E212B479C60B` gets `76C6CFD8A59C0000`, which is the decrypted last block.

Continuing in reverse block order, `7FD0E212B479C60B` TDES Decrypt with `27F66D5244FF621E AA6F6120EDEB427F` gets `AE81BFA4A2C80006`, XOR `C07B12648DCAC4FD` gets `6EFAADC02F02C4FB`, which is decrypted block 6.

Continuing in reverse block order, `C07B12648DCAC4FD` TDES Decrypt with `27F66D5244FF621E AA6F6120EDEB427F` gets `AAC8D06ACCF27E6D`, XOR `BE6EE7466B81196E` gets `14A6372CA7736703`, which is decrypted block 5.

Continuing in reverse block order, `BE6EE7466B81196E` TDES Decrypt with `27F66D5244FF621E AA6F6120EDEB427F` gets `01D78CB7D1DAEA95`, XOR `6F4B691E633C311C` gets `6E9CE5A9B2E6DB89`, which is decrypted block 4.

Continuing in reverse block order, `6F4B691E633C311C` TDES Decrypt with `27F66D5244FF621E AA6F6120EDEB427F` gets `0D2620B051231748`, XOR `32BA90BFB3E6B41D` gets `3F9CB00FE2C5A355`, which is decrypted block 3.

Continuing in reverse block order, `32BA90BFB3E6B41D` TDES Decrypt with `27F66D5244FF621E AA6F6120EDEB427F` gets `41499B60A6AAD427`, XOR `8628E664C59BBAA2` gets `C7617D0463316E85`, which is decrypted block 2.

Continuing in reverse block order, `8628E664C59BBAA2` TDES Decrypt with `27F66D5244FF621E AA6F6120EDEB427F` gets `010002D4B69CD2C0`, which is decrypted block 1.

Ordering the decrypted blocks first to last gives:

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

The host software can ignore the last two bytes because by definition MagnePrint data is 54 bytes long:

```
010002D4B69CD2C0C7617D0463316E853F9CB00FE2C5A3556E9CE5A9B2E6DB8914A637
2C A77367036EFAADC02F02C4FB76C6CFD8A59C0000
```

The encrypted session ID data is:

```
21685F158B5C6BE0
```

As this is a simple eight byte block, we only need decrypt it with the appropriate key:
21685F158B5C6BE0 TDES Decrypt with 27F66D5244FF621E AA6F6120EDEB427F gets 0000000000000000. All zeroes is the expected value because in this example the host software did not specify a session ID.

A.1.3 Example: Changing from Security Level 2 to Security Level 3

```
; This script demonstrates changing from Security Level 2 to Security
Level 3.
; It assumes the device is at Security Level 2 with the ANS X9.24
Example
; key loaded and the KSN counter set to 1.
09 00        ; Get current KSN (should be FFFF9876543210E00001)
Request      : CMND=09, LEN=00, DATA=
Response     : RC=  00, LEN=0A, DATA=FF FF 98 76 54 32 10 E0 00 01

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

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

Delay        : (waited 5 seconds)
```

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

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

A.1.4 Example: Changing from Security Level 2 to Security Level 4 (MSR Only)

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

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

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

Delay         : (waited 5 seconds)
```

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

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

A.1.5 Example: Changing from Security Level 3 to Security Level 4 (MSR Only)

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

; For KSN 2, MAC Key: C46551CEF9FDDBB0 AA9AD834130DC4C7
;
; The command to change Security Level looks like: 15 05 04 nnnnnnnn
;  where nnnnnnnn is the MAC.
;
; The data to be MACd is: 15 05 04
; Data to be MACd must be in blocks of eight bytes, so we left justify
and
; zero fill the block to get: 15 05 04 00 00 00 00 00 (This is the
block to MAC)
; For convenience show it as the compacted form: 1505040000000000
;
; The MAC algorithm run with this data uses the following
cryptographic
; operations:
;
;  Single DES Encrypt the data to be MACd with the left half of the
MAC Key:
;     1505040000000000 1DES Enc with C46551CEF9FDDBB0 =
735323A914B9482E
;
;  Single DES Decrypt the result with the right half of the MAC Key:
;     735323A914B9482E 1DES Dec with AA9AD834130DC4C7 =
390E2E2AC8CB4EE6
;
;  Single DES Encrypt the result with the left half of the MAC Key:
;     390E2E2AC8CB4EE6 1DES Enc with C46551CEF9FDDBB0 =
D9B7F3D8064C4B26
;
; The leftmost four bytes of the final result are the MAC = D9B7F3D8
;
; Send the MACd Set Security Level command
15 05 04 D9B7F3D8
Request      : CMND=15, LEN=05, DATA=04 D9 B7 F3 D8
Response     : RC=  00, LEN=00, DATA=

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

Delay        : (waited 5 seconds)
```

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

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

A.1.6 Example: Authentication (MSR Only)

In this example, the device is already in **Security Level 3** or **Security Level 4**. The script puts the device into Authenticated Mode, leaves it in that mode for a time, then deactivates it.

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

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

09 00        ; Get current KSN (should be FFFF9876543210E00002)

; Send the Activate Authenticated Mode command (4 minutes)
10 02 00F0
Request      : CMND=10, LEN=02, DATA=00 F0
Response     : RC=  00, LEN=1A, DATA=FF FF 98 76 54 32 10 E0 00 03 AA
AA AA AA AA AA AA AA DD DD DD DD DD DD DD DD
                                     |------- Current KSN -------| |--
-- Challenge 1 ----| |---- Challenge 2 ----|
Response     : RC=  00, LEN=1A, DATA=FF FF 98 76 54 32 10 E0 00 03 BE
5C 98 35 17 7E 45 2A A7 2D 2D B2 36 BF 29 D2
;  Challenge 1 Encrypted: BE5C9835177E452A
;  Challenge 2 Encrypted: A72D2DB236BF29D2

; Note that the KSN now ends with a counter of 3!
; Decrypt Challenge 1 using variant of Current Encryption Key
;  (Current Encryption Key XOR with F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0)
;
;  Current Key   0DF3D9422ACA561A 47676D07AD6BAD05
;          XOR   F0F0F0F0F0F0F0F0 F0F0F0F0F0F0F0F0
;            =   FD0329B2DA3AA6EA B7979DF75D9B5DF5
;
;     BE5C9835177E452A TDES Decrypt with FD0329B2DA3AA6EA
B7979DF75D9B5DF5 = 7549AB6EB4840003
;
;  Note that the final two bytes of the result = 0003, matching the
KSN as
;   transmitted in the clear.  This provides Authentication to the
host that
;   the device is what it claims to be (proves key knowledge).
;
; Decrypt Challenge 2 using Current Encryption Key variant as above
;     A72D2DB236BF29D2 TDES Decrypt with FD0329B2DA3AA6EA
B7979DF75D9B5DF5 = 34DB9230698281B4
;
;
```

```
; Build an Activation Challenge Reply command (cmd, len, cryptogram)
;  11 08 XXXXXXXXXXXXXXXX
;
;  The clear text input for the cryptogram is composed of the first
six bytes
;  of the decrypted Challenge 1 followed by two bytes specifying how
long to
;  stay in the Authenticated Mode.
;
;     CCCCCCCCCCCC TTTT
;
;     Time examples:
;        For 30 seconds use 001E
;        For 99 seconds use 0063
;        For 480 seconds use 01E0
;        For 1200 seconds use 04B0
;
;  These values are concatenated to form an eight byte block, we will
use 480
;  seconds:
;
;     CCCCCCCCCCCC01E0
;
;  The block is encrypted using a variant of the Current Encryption
Key
;  (Current Encryption Key XOR with 3C3C3C3C3C3C3C3C3C3C3C3C3C3C3C3C)
;
;  Current Key   0DF3D9422ACA561A 47676D07AD6BAD05
;          XOR   3C3C3C3C3C3C3C3C 3C3C3C3C3C3C3C3C
;            =   31CFE57E16F66A26 7B5B513B91579139
;
;     7549AB6EB48401E0 TDES Enc with 31CFE57E16F66A26 7B5B513B91579139
= A30DDE3BFD629ACD
;
; Send the Activation Challenge Reply Command
11 08 A30DDE3BFD629ACD

; Build a Deactivate Authenticated Mode command (cmd, len, cryptogram)
;  12 08 XXXXXXXXXXXXXXXX
;
;  The clear text input for the cryptogram is composed of the first
seven bytes
;  of the decrypted Challenge 2 followed by one byte specifying
whether to
;  increment the DUKPT KSN or not (00 = no increment, 01 = increment).
;
;     DDDDDDDDDDDDDD II
;
;  These values are concatenated to form an eight byte block, we will
specify
;  No Increment:
```

```
;
;     DDDDDDDDDDDDDD00
;
;  The block is encrypted using a variant of the Current Encryption
Key
;  (Current Encryption Key XOR with 3C3C3C3C3C3C3C3C3C3C3C3C3C3C3C3C)
;
;  Current Key   0DF3D9422ACA561A 47676D07AD6BAD05
;          XOR   3C3C3C3C3C3C3C3C 3C3C3C3C3C3C3C3C
;            =   31CFE57E16F66A26 7B5B513B91579139
;
;     34DB923069828100 TDES Enc with 31CFE57E16F66A26 7B5B513B91579139
= CA CB BD 5F 58 D5 C9 50
;
; Send the Deactivate Authenticated Mode command
12 08 CACBBD5F58D5C950
```

A.2 About the SDKs and Additional Examples

MagTek provides SDKs and corresponding documentation for many programming languages and operating systems that enable software developers to quickly develop custom host software that communicates with this device, without having to deal with the complexities of platform APIs for direct communication across the various available connection types, connecting using the various available communication protocols, and parsing the various available data formats.

The SDKs and corresponding documentation include:

- Functions for sending the direct commands described in this manual
- Wrappers for commonly used commands and properties that further simplify development
- Detailed compilable examples of processing incoming payment data and using the direct commands and properties described in this manual

To download the SDKs and documentation, search www.magtek.com for “SDK” and select the SDK and documentation for the programming languages and platforms you need, or contact MagTek Support Services for assistance.

Appendix B Identifying ISO/ABA and AAMVA Cards For Masking (MSR Only)

B.1 ISO/ABA Financial Card

The device uses the rules below to determine if a card is an ISO/ABA card (per ***ISO 7811-2,2001***), which affects the masking used for **Masked Track Data**. ISO defines a particular and different bit-level character encoding format of the data on each of the three tracks of the card. The format of the card depends on decisions made by the entity that issued the card. For example, some organizations may choose to use the ISO Track 1 encoding format for Track 2 data, or other permutations that do not conform to the standard. The device only considers ISO Financial masking for cards it classifies as ISO, which it determines according to the following rules:

1) If the low level decoding algorithm determines the bit level character encoding for every track conforms to the ISO format defined for that track, the card is classified as ISO. Otherwise the device attempts to classify the card as an **AAMVA Driver's License**, and if the card fails that test, the device classifies the card as **Other**. A properly encoded ISO card has the following properties:
   a) At least one track must be decodable.
   b) Track 1 must be 7 bits per character.
   c) If Track 2 or Track 3 exist, they must be 5 bits per character.
2) If the device determines the card is ISO encoded, it then determines the masking behavior for each track independently. One track may qualify for masking and another may not, according to the following rules.
3) For Track 1, the device's intent is to send the card's Format Code unmasked, the PAN partially masked, the Name and Expiration Date unmasked, and the rest of the track masked, with exceptions:
   a) The Service Code is always unmasked on newer devices (Never Mask Service Code Only). On legacy devices, the Service Code is always masked.
   b) If the card's Format Code, PAN, Name, or Expiration Date are not correctly structured, the device transmits the rest of the track unmasked starting with the point of discrepancy. The device defines "correct structure" for Track 1 as follows:
      i) The card's Format Code, PAN, Name, or Expiration Date do not contain the '?' character (End Sentinel).
      ii) The Format Code is the first character on the track and is the character 'B'.
      iii) The PAN has a maximum of 19 digits and ends with character '^' (Field Separator).
      iv) The Cardholder Name has a maximum of 26 characters and is ended by the character '^' (Field Separator).
      v) The Expiration Date has 4 characters.
      vi) The Service Code has 3 characters.
4) For Track 2, the device's intent is to send the PAN partially masked, the Expiration Date unmasked, and most of the rest of the track masked, with exceptions:
   a) The Service Code is always unmasked on newer devices (Never Mask Service Code Only). On legacy devices, the Service Code is always masked.
   b) If the PAN or Expiration Date are not correctly structured, the device sends the rest of the track unmasked starting at the point of discrepancy. The device defines "correct structure" as follows:
      i) The PAN or Expiration Date does not contain the '?' character (End Sentinel).
      ii) The PAN has a maximum of 19 digits and ends with the character '=' (Field Separator).
      iii) The Expiration Date has 4 characters.

        iv) The Service Code has 3 characters.

5) For Track 3, the device's intent is to send the PAN partially masked and the rest of the track masked, with exceptions:
    a) If the PAN is not correctly structured, the device sends the rest of the track unmasked, starting at the point of discrepancy. The device defines "correct structure" as follows:
        i) The PAN does not contain the '?' character (End Sentinel).
        ii) The PAN has a maximum of 19 digits and ends with character '=' (Field Separator).

B.2 AAMVA Driver's License

The device uses the following rules to determine if a card is an AAMVA card:

1) If the device 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*** [sic.] rules, the card is considered to be an AAMVA card. Some MagTek devices do not support reading of Track 3, so this rule does not apply on such devices.
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 are "604425" or contain values in the range "636000" to "636062" inclusive, the card is considered to be an AAMVA card.

AAMVA card masking, when enabled, works as follows:

1) The device sends track 1 and track 3 entirely masked; all character positions are filled with zeroes.
2) Track 2 is treated as follows:
    a) The device's intent is to send the Driver License ID (DLID) partially masked, the Expiration Date unmasked, the Birth Date unmasked, and the rest of the track masked.
    b) If the DLID, Expiration Date, or Birth Date are not correctly structured, the rest of the track, starting at the point of discrepancy, is sent unmasked. The device defines "correctly structured" as follows:
        i) If the DLID, Expiration Date, or Birth Date contain the '?' character (End Sentinel), the field is not correctly structured.
        ii) A correctly structured DLID has a maximum of 19 digits and is terminated by the character '=' (Field Separator).
        iii) A correctly structured Expiration Date has 4 characters.
        iv) A correctly structured Birth Date has 8 characters.