MAGTEK®
SECURITY FROM THE INSIDE

iDynamo 5 Gen III

Secure Card Reader
Programmer's Manual (COMMANDS)

June 2024

Document Number:
D998200587-102

REGISTERED TO ISO 9001:2015

MagTek I 1710 Apollo Court I Seal Beach, CA 90740 I Phone: (562) 546-6400 I Technical Support: (562) 546-6800
www.magtek.com

Table 0-1 - Revisions

Rev Number	Date	Notes
100	March 15, 2024	Initial release.
101	May 05, 2024	Add **Property 0x88 USB Packet Delay**
102	June 12, 2024	Add **6.27.1 BYTE Description**

Table of Contents

1 Introduction

1.1 About This Document

This document describes how to communicate with iDynamo 5 Gen III secure card reader (SCR) using the MP application specific integrated circuit (ASIC).

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; **device** refers to iDynamo 5 Gen III.
- **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**.

2 Connection Types

iDynamo 5 Gen III connects to iOS products via a USB-C or Lighting to USB-C cable.

2.1 How to Use Apple iAP2 Connections

This section provides information about developing an iOS app that interfaces with the device via the Lightning or USB connector using iPod Accessory Protocol 2 (iAP2). 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:

- ***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 ***External Accessory*** Framework (for sample code, see Apple's ***EADemo*** app). Upon establishing the connection, the host can begin exchanging data with the device.

2.2 USB Communications

2.3 USB Interface

This USB device conforms to the USB specification revision 2.0 and Human Interface Device (HID) class specification version 1.11. The device is set up as a full-speed, high-powered USB device that draws power from the USB bus. iDynamo5 Gen III identifies itself to the USB host with MagTek's vendor ID **of 0x0801** and Product ID (PID) of **0x0020.** All USB enumeration will include the device serial number.

2.3.1 About USB Reports

HID reports used by the host can be divided into two types:

- **Feature Reports**, which the host uses to send commands to the device and receive responses using **Get Feature** and **Set Feature.**
- **Input Reports** are used by the device to send unsolicited notifications to the host when the device's state changes, or to send asynchronous responses to the host when a command completes. The device commonly uses input reports when reporting card swipes, device events, or when a command takes more time for the device to process than is reasonable for the host to wait on a blocking call for the device to acknowledge completion.

2.3.2 USB Keyboard Emulation

A device in KB mode identifies itself to the USB host as a keyboard and transmits data to the host as ASCII as though it is being typed by a person on an actual keyboard. It does this by mapping each of the possible ASCII characters in the stream to keystrokes. To send an ASCII character to the host, the device looks up the ASCII character in the key and retrieves a combination of a single **Key Usage** which is a unique value assigned to every keyboard key, and a **Key Modifier Byte** and sends them to the host. The key modifier byte modifies the meaning of the key usage ID, by indicating whether any combination of the right or left **Ctrl**, **Shift**, **Alt** or GUI keys [as defined by *Universal Serial Bus (USB) Device Class Definition for Human Interface Devices (HID)*] are pressed at the same time as the key usage ID.
The device transmits ASCII 0 to 31 and 127 as their equivalent control code combinations. For example, for a carriage return value 13 (`0x0D`), the device appears to the host as a keyboard where a person very quickly presses and holds the **Ctrl** key, then presses the **M** key, then releases both keys.
When the keymap contains a Key Usage ID and Key Modifier Byte of `0xFF` for the ASCII value the device wants to send, the device uses **Alt** ASCII code keystrokes instead of key map values, meaning it simulates holding down the **Alt** key on a keyboard and typing the three-digit decimal value of the ASCII character it wants to send. For example, to transmit the ASCII character '?' (063 decimal in the ASCII table), the device sends keypad '0' combined with the **Left Alt** key modifier, then keypad '6' combined with the **Left Alt** key modifier, then keypad '3' combined with the **Left Alt** key modifier.

2.3.2.1 Commands and Responses

Feature reports are used to send commands and receive responses, even when KB mode is active.

2.3.2.2 Notifications (RFU)

Table 2-1 Notification in KB mode

Description		Type	Txt Len	Notes
Message ID = "N001"	Clear	ASCII	4	Notification Msg ID in KB mode
Notification Length	Clear	HEX	4	Notification Length
Notification	Clear	Hex	var	Notification in ASCII-Hex

2.4 USB HID Protocol

This section defines how the device communicates with the host over the USB HID interface.

2.4.1 HID Packeting

The data transmitted in commands, responses, and notifications can be substantial, with a size limit of 64 kilobytes (64K bytes). However, the USB HID interface utilized by V5 restricts messages to 64 bytes. Therefore, to accommodate larger messages, we employ specific methods to segment them into multiple 64-byte packets. The segmentation approach varies depending on the message type to ensure compatibility with legacy V5. Below is the core structure of a USB HID packet.

Table 2-2 HID Packet Definition

Data Offset	ID	Complete Data Len	Data Segment
2 bytes	2 bytes	2 bytes	varies

2.4.2 Sending USB HID Messages to the Host

The device makes data available to the host using one or more Input Reports over a USB Interrupt IN pipe. The host will poll the device at the configured Polling Interval to see if Input Reports are ready. The device must respond to polls with a USB `NAK` when no Input Reports are available.

- Data Messages sent by the device use Input Report ID 0x01.
- Notification Messages sent by the device use Input Report ID 0x02.

2.4.3 Receiving USB Data from the Host

The host sends commands using a Set Feature Report and sends a Get Feature Report to the device to retrieve a synchronous response when appropriate. Feature reports use report ID 0x01.

The host should send both Feature Report types using the default Control pipe using a blocking call to the operating system's native USB libraries. The device NAKs the Status page of a Set Feature Report until it finishes the requested operation, and if it does not respond, the operating system will generally time out and report failure. This method ensures that as soon as the device has fulfilled the command request embedded in the Set Feature Report, the host software can immediately call a follow-up Get Feature Report to retrieve the command.

In very rare cases, the host may simply send a Get Feature Report directly without a preceding Set Feature Report. The Commands documentation specifies these special cases if they exist.

2.4.4 Command Packets

Commands are sent to the device using USB Set Feature Reports using Report ID 1.

Table 2-3 Standard Command Format

Command	Parameter Data Len	Parameter Data
2 bytes	2 bytes	0-64k bytes

Sending commands to the device over a packeted interface (maximum parameter size is 64k (0xFFFF). Standard command format gets broken down into packets for interfaces that do not support unlimited length messages. This is compatible with the V5 extended command protocol.
The number ranges shown in packets assume the interface is USB with a maximum message size of 64 bytes.

Table 2-4 USB HID Command Packet

Header	Packet Data Len (Includes all subsequent field lengths)	Data Offset	Command	Full Parameter Data Len	Partial Parameter Data
0x49	1 Byte (6-58)	2 bytes	2 bytes	2 bytes	0-52 bytes

The Command and Parameter Data length fields stay the same for all packets. Each packet will contain one segment of parameter data up to 52 bytes. The data offset indicates which portion of the parameter data is contained within this packet.

If the required Parameter Data is 52 bytes or shorter, the host can send the entire command using a single command packet. If the Parameter Data is longer than 52 bytes, the host should split the data into multiple packets of 52 or fewer bytes and send multiple command packets. Assuming 52-byte packets, the first packet the host sends should specify Data Offset = 0, the next packet should specify Data Offset = 52, and so on, until the host has sent all the Parameter Data. The device's response to each packet contains either an extended command result code or a standard result code for the command that was sent:

A packet with Data Offset = 0 is considered the start of a new command. If there are still packets pending from a previous command, then that command is cleared, and all data received is erased.

The device will respond to packets with a 0x0B,0x00 when it expects additional packets (i.e., the length of parameter data received is less than the full parameter data length). When all data has been received, then the device will respond with a packet starting with 0x0A and include the response message for the command.

ACK Packet … Send next packet.

Result Code 0x0B - Protocol Request Pending indicates the device is buffering the incoming data and expects the host to send subsequent packets.

Table 2-5 Device to Host – Send next packet.

Packet Result	Data Len
0x0B	0x00

Table 2-6 Device to Host – Last packet received and Response

Pkt Response	Data Len	Response
0x0A	1 byte	First portion of response

Error due to invalid or missing data or due to a timeout while the device is waiting for more packets.

Table 2-7 - Error Invalid or Missing Data

Packet Result	Data Len
0x02	0x00

2.4.5 Response Packets

Response messages are generated after executing a command and made available to the host as a feature report with the report ID set to 1. The host uses the USB Get Feature Report to receive the response or a portion. The following should be compatible with V5 extended responses.

Table 2-8 Standard Response Message Format (aka Extended Response in V5)

Return Code	Response Data Len	Response Data
2-bytes	2-bytes	0...n bytes

After the host receives the initial response packet, it's up to the host to collect the data and send 0x4A commands to get more response packets until all data has been received.

Table 2-9 USB HID Response Packet

Header	Packet Data Len (Includes all subsequent field lengths)	Data Offset	Response Code	Full Response Data Len	Partial Response Data
0x0A	1 Byte (6-58)	2 bytes	2 bytes	2 bytes	0-52 bytes

Table 2-10 Host to Device – Send next packet of Response.

Send Next Packet Command	Data Len
0x4A	0x00

2.4.6 Notifications

Notification messages with a data size longer than 52 bytes will need to be split into multiple packets to accommodate the USB HID interface. Notifications are made available to the host as one or more USB Input Reports with the report ID set to 2.

Table 2-11 Notification Packets

Length	Field Name	Description
1	**Partial Data Length**	The length of the **Data** field contained in the current message. This field is in big endian format. If this value is not equal to the **Complete Data Length**, the device is sending the notification using multiple packets.
2	**Data Offset**	The offset position in bytes within the entire assembled notification where the first byte of the current packet's **Data** field is located. This field is in big endian format. The first byte of the entire notification's Data is at offset zero.
2	**Notification Identifier**	.ID
2	**Complete Data Length**	The total length of data for the entire notification message, summing all **Partial Data Lengths** for multiple packets. This field is in big endian format. If this value is not equal to the **Partial Data Length** of the current packet, the device is sending the data using multiple packets.
Varies	**Data**	May contain part or all the notification data. The size of this field is contained in the **Partial Data Length** field.

2.4.7 Data

Data messages are another type of asynchronous communication and handled in the same manner as a notification, except the report ID is set to 1.

Table 2-12 Data Message Packets

Length	Field Name	Description
1	**Partial Data Length**	The length of the **Data** field contained in the current message. This field is in big endian format. If this value is not equal to the **Complete Data Length**, the device is sending the data using multiple packets.
2	**Data Offset**	The offset position in bytes within the entire assembled notification where the first byte of the current packet's **Data** field is located. This field is in big endian format. The first byte of the entire Data message is at offset zero.
2	**Reserved**	Set to 0000
2	**Complete Data Length**	The total length of data for the entire data message, summing all **Partial Data Lengths** for multiple packets. This field is in big endian format. If this value is not equal to the **Partial Data Length** of the current packet, the device is sending the data using multiple packets.
Varies	**Data**	May contain part or all the data. The size of this field is contained in the **Partial Data Length** field.

2.5 iOS SLIP Format

When connected to an iOS host, the device communicates over what appears to be a simple bidirectional serial line transferring binary data. To manage serial communications, we utilize an old method called SLIP (Serial Line Internet Protocol). This allows the addition of message control characters and a method to distinguish between control and data.

2.5.1 Message Identification

Each message is prefixed by a one-byte message type followed by a two-byte message length. The possible types are:

- 00 - Data (Data is ASCII text to be compatible with KB emulation output)
- 02 - Notification
- 04 - Response
- 05 – Command

Table 2-13 Message with iOS prefix

Message Type	Message Len	Message
1 byte	2 bytes	Length varies

2.5.2 Processing a Message Using SLIP

Messages are framed before and after with a byte value of 0xC0. Because this value can also appear as part of the data message, a few 2-byte sequences are added to distinguish between control and data bytes. The sender modifies the outgoing data, and the receiver translates the incoming data back to the original message. Conversion can be done on the fly during the sending and receiving processes.

- Assemble message.
- Add message type and length to the beginning of the message.
- Replace any bytes with value 0xDB with bytes 0xDB 0xDD.

- Replace any bytes with value 0xC0 with bytes 0xDB 0xDC.
- Start and end the message with bytes 0xC0.

Table 2-14 Message with SLIP framing

Start	Message Type	Message Len	Message	End
0xC0	1 byte	2 bytes	Length varies	0xC0

```
SLIP example
Message 01 DB C0 05
SLIP C0 01 DB DD DB DC 05 C0
```

3 Data Format

Multi-byte values like command and data lengths are always **big-endian**.

3.1 Message Types

3.1.1 Data

Data Messages are formed in response to a cardholder event such as swiping a card or pressing a button. These messages are formatted as a block of ASCII text and compatible with all interfaces.

3.1.2 Commands

Commands consist of a 2-byte command, a 2-byte Parameter Data Length, and Parameter Data as needed.

Table 3-1 Command Structure

Command	Parameter Data Len	Parameter Data
2-bytes	2-bytes	0…n bytes

3.1.3 Responses

Responses consist of a 2-byte return code, a 2-byte response data length, and response data as needed.

Table 3-2 Response Structure

Return Code	Response Data Len	Response Data
2-bytes	2-bytes	0...n bytes

3.1.4 Notifications

Notifications consist of a 2-byte notification code, a 2-byte notification data length, and notification data when needed. They are sent asynchronously due to device events. They can also be used as delayed responses for commands that take longer to process.

Table 3-3 Notification Structure

Notification ID	Notification Data Len	Notification Data
2-bytes	2-bytes	0...n bytes

3.2 Data Output

3.2.1 MSR Track Data

3.2.1.1 Sentinels

Data for each card track is typically bracketed by a start and end sentinel. The sentinel characters can be changed by setting properties. The start sentinel can also indicate what format was used to encode the track.

[SS] Track Data [ES]

If a property value is set to 0, then no character will be sent.

Table 3-4 Track Data Sentinel Properties

Name	Description	Length	Property	Default
T1ISO	Tk1 SS if ISO	1	0x24	0x25 '%'
T2ISO	Tk2 SS if ISO	1	0x25	0x3B ';'
T3ISO	Tk3 SS if ISO	1	0x26	0x2B '+'
T3AMV	Tk3 SS if AAMVA	1	0x27	0x23 '#'
T27BT	Tk2 SS if 7bit	1	0x28	0x40 '@'
T37BT	Tk3 SS if 7bit	1	0x29	0x26 '&'
ES	End sentinel	1	0x2B	0x3F '?'

3.2.1.2 Masking

The PAN field must always be partially masked. The device can be configured to expose the 0-8 leading characters and 0-4 trailing characters. The 8-digit limit for leading characters automatically drops to 6 for cards where the PAN length is less than 16 (e.g. American Express).

Properties 0x07 and 0x08 are used to configure PAN masking.

This device will always mask cardholder name, expiration date, and service code.

3.2.1.3 DUKPT Key Info for Encrypted Data Output

Data messages include DUKPT KEY information fields so the host can derive the correct decryption or MAC key.

Table 3-5 DUKPT Key Derivation Information

Byte	Len	Description	Values	
1	1	DUKPT Key Info Version	00 = Legacy DUKPT 01 = Current (AES) DUKPT	
2	1	For Data item	0-rfu 1-Message MAC 2-MSR Data 3-MP Token	
3	1	Using Mode/operation	0-RFU 1-ENC-CBC-0 2-ENC-CBC-SECURE 3-ENC-CTR 0X10-MAC-CBC-0 0X11-CMAC 0X12-HMAC 0x13-GMAC	
4	1	Derived Key Algorithm	0x00 = 2-key TDEA 0x01 = 3-key TDEA 0x02 = AES 128-bit 0x03 = AES 192 bit 0x04 = AES 256 bit 0x05 = HMAC	Bytes 4,5

Byte	Len	Description	Values	
5-6	2	Generated key length (bits)	Ex: 256 = 0x0100 Ex: 128 = 0x0080	6,7
7-8	2	Derived key Usage (refer to 9.24-3 for all possibilities)	2002 = MAC, both ways 3002 = Data Encryption, both ways FF00 = Legacy PIN Variant FF01 = Legacy MAC, both ways FF02 = Legacy Data Encryption	2,3 AES DUKPT: Use Legacy: FFnn – shift FF left nn bytes

```
Example:
00 02 01 00 0080 FF02 MSR encrypt CBC zero pad with legacy DUKPT 128-
bit TDEA data variant.
```

Note: Derived key algorithm, length, and usage information will be needed to generate the correct AES-DUKPT decryption key.

3.2.2 Data Messages

Data messages will always be made up of ASCII text characters regardless of the interface for compatibility purposes. Text is used to address limitations when using keyboard emulation.

3.2.2.1 Normal Operation – Financial Card Read Message

Any Field with no value will be empty between separator characters.

Table 3-6 Data Message M001 Definition

Field Description	Prot	Type	Txt Len	Notes
Message ID = "M001"	Clear	ASCII	4	MSR Data Message
Track 1 Masked Data	Clear	ASCII	var	Per masking configuration
Track 2 Masked Data	Clear	ASCII	var	Per masking configuration
Track 3 Masked Data	Clear	ASCII	var	Per masking configuration
Track 1 Data	Encrypt	HEX	var	Encrypted with MSR key.
Track 2 Data	Encrypt	HEX	var	Encrypted with MSR key.
Track 3 Data	Encrypt	HEX	var	Encrypted with MSR key.
MP Status Code	Clear	HEX	8	
MP Token	Encrypt	HEX	var	Encrypted with MSR or MP key as configured (see Property 0x15).
Session ID	Encrypt	HEX	16/32	Encrypted with MSR key. Session ID = RTC value.
KSN (MSR)	Clear	HEX	20/24	
DUKPT Key Info (MSR)	Clear	HEX	16	
KSN (MP)	Clear	HEX	0/20/24	Empty when MSR key is used

Field Description	Prot	Type	Txt Len	Notes
DUKPT Key Info (MP)	Clear	HEX	16	Empty when MSR key is used
Device Serial Number	Clear	ASCII	7	Indicates valid range of each hex digit – '0' ~ '9' (0x30 ~ 0x39), 'A' ~ 'F' (0x41 ~ 0x46)
DUKPT Key Info (MAC)	Clear	HEX	16	Using MSR Key
Message Length	Clear	HEX	4	Include all of message except for MAC. Length required for MAC security. High byte first.
MAC	Clear	HEX	16/32 for 2TDES/AES	MSR encryption key used in message. Message padded with zeros by host for MAC calculation.

Note: Text length may vary depending on the key type that is being used.

3.2.2.2 Field Separation

This device uses configurable properties to define characters that get inserted into the message for parsing purposes. The simplest option is to have characters for start of message (SOM), end of message (EOM), and field separation (FS) only.

Table 3-7 Separating text fields for host parsing.

Name	Description	Length	Property	Default	To Disable
SOM	Start of Message	0-7	0x1E	0	0
EOM	End of Message	0-7	0x22	'\r' (0x0D)	0
FS	Field Separator	1	0x23	'|' (0x7C)	0

Note: Portion in bold shows the data included in the output MAC calculations.

```
[SOM]
M001
[FS] Track 1 Masked Data [FS] Track 2 Masked Data [FS] Track 3 Masked
Data
[FS] Track 1 Encrypted Data [FS] Track 2 Encrypted Data [FS] Track 3
Encrypted Data
[FS] [MP Status] [FS] [Encrypted MP Data]
[FS] [Encrypted Session ID]
[FS] [MSR DUKPT Key Serial Number] [FS] [MSR DUKPT Key Info]
[FS] [MP DUKPT Key Serial Number] [FS] [MP DUKPT Key Info]
[FS] [Device Serial Number]
[FS] [MAC DUKPT Key Info] [FS] [MAC message length]
[FS] [MAC] [EOM]

[SOM]
Q001
[FS] [Token DUKPT Key Serial Number] [FS] [Token DUKPT Key Info]
[FS] [QWANTUM Status] [FS] [QWANTUM Token]
```

```
[FS] [Encrypted Session ID]
[FS] [QWANTUM Card ID]
[FS] [Device Serial Number]
[FS] [MAC DUKPT Key Info] [FS] [MAC message length]
[FS] [MAC] [EOM]

[SOM]
Q002
[FS] [Token DUKPT Key Serial Number] [FS] [Token DUKPT Key Info]
[FS] [Encrypted Session ID]
[FS] [Encrypted QWANTUM Data Buffer]
[FS] [Device Serial Number]
[FS] [MAC DUKPT Key Info] [FS] [MAC message length]
[FS] [MAC][EOM]
```

4 Commands

4.1 Command 0000 - Get Property

This command lets the host retrieve a property (see section on properties) from the device using the 1-byte Property ID.

Table 4-1 Get Property Command

Command	Parameter Data Len	Parameter Data
0x0000	0x0001	Property ID (1-byte)

Table 4-2 Get Property Response

Return Code	Response Data Len	Response Data
0x0000	varies	Property Value

Legacy (if needed) for simple & short properties.

Table 4-3 Legacy Get Property

Command	Parameter Data Len	Parameter Data
0x00	0x01	Property ID (1-byte)

Table 4-4 Legacy Response Structure

Return Code	Response Data Len	Response Data
0x00	varies	0 to 58 bytes

4.2 Command 0001 - Set Property

This command sets a property in the device. For secure properties, this command should be the payload for the Send Secured Command.

Table 4-5 Set Property Command

Command	Parameter Data Len	Parameter Data
0x0001	varies	Property ID (1 byte) Property Value (varies)

Table 4-6 Set Property Response

Return Code	Response Data Len	Response Data
0x0000	0x0001	0=Property in effect 1=Needs reset

4.3 Command 0002 - Reset Device

This command is used to reset the device.

Table 4-7 Reset Device Command

Command	Parameter Data Len	Parameter Data
0x0002	0x0000	None

4.4 Command 0x030D - Read Date and Time

The host uses this command to get the date / time from the device's internal clock. The value returned is set to Coordinated Universal Time (UTC). The host is responsible for converting the response to local time.

Table 4-8 Read RTC Command

Command	Parameter Data Len	Parameter Data
0x030D	0x0000	None

Table 4-9 Read RTC Response

Return Code	Response Data Len	Response Data
0x0000	0x0007	See Table

Table 4-10 Read RTC Response Data

Length	Field Name	Value
1	Month	Value from 0x01...0x0C (1-12)
1	Day	Value from 0x01...0x1F (1-31, depends on the month)
1	Hour	Value from 0x00...0x17 (0-23)
1	Minute	Value from 0x00...0x3B (0-59)
1	Second	Value from 0x00...0x3B (0-59)
1	Unused	0x00
1	Year	Value from 0x00 (2008) ...0xFF (2263)

4.5 Command 0x0703 – Get Key Information

This command returns the information about the specified key to the host.

Table 4-11 Get Key Info Command

Command	Parameter Data Len	Parameter Data
0703	0002	Key ID

Table 4-12 Response Data for Get Key Information Command

Offset	Field Name	Description
0	Key Slot Status (1 byte)	0 = Empty 1 = Loaded (Purpose not assigned) 2 = Loaded & active 3 = Exhausted (End of DUKPT key sequence) 4 = Expired (RFU, cert status?) 0xFF = Not supported in this device
1	Slot Type (1 byte)	First byte of Slot ID (Slot ID = Key ID)
2	TK ID (2 byte)	The Key ID used to transport this key (a parent key ID)
The following fields are required if Key Slot is not empty.		
4	Key Environment (1 byte)	'T' for test or 'P' for production
5	X9.143 Attributes (4 bytes)	'Key Usage || Algorithm || Mode of Use' from X9.143 Key Block Header.
9	Key Algorithm (1 byte)	1 = DES 4 = AES128 2 = 2TDES 5 = AES192 3 = 3TDES 6 = AES256
10	KCV (5 or 3 bytes)	5 bytes for AES-CMAC or 3 bytes for TDES-CBCMAC including the KCV algorithm info as defined in X9.143 'KP'/'KC' format (A.5.8 of X9.143 spec)
15 or 13	Length of KSI (1 bytes)	Length of Key Set Identifier
16 or 14	Key Set Identifier (n bytes)	Encoded in Hex-ASCII refer to Table 11 in X9.143 specification.
(16 + n) or (14 + n)	Key Restriction (2 bytes)	16-bit Key Restriction Bitmask (for TK and DKPT keys)
(16 + n + 3) or (14 + n + 3)	Key Configuration (2 bytes)	16-bit Key Configuration Bitmask Currently supports Data Type Configuration only.
(16 + n + 4) or (14 + n + 4)	Timestamp (24 bytes)	Time and date in UTC time format that indicates when the key block was formed.

4.6 Command 0x0711 – Download Firmware File

The iDynamo5 Gen III API and physical interfaces impose a limit on the complete message size, restricting it to less than 64k. Given that the firmware file size is reported to be approximately 128k, the file needs to be transmitted in large chunks using multiple commands. Each command may require further packetization based on the specific interface rules. To ensure compatibility with interface restrictions, the parameter data length should be kept below the maximum allowable size. It is advised to maintain a parameter data length in the range of approximately 1k to 16k.

File loading can be terminated by issuing a Load command with a File Size parameter set to 0. The device should return an error if parameter checking fails. It is recommended to authenticate the file before providing it to the bootloader.

Table 4-13 Download Firmware Command

Command	Parameter Data Len	Parameter Data
0x0711	Varies – includes 8 bytes for file size and index	See Table

Table 4-14 Download Firmware Parameters

Length	Field Name	Value
4	File Size	Size of firmware file
4	File Offset/Index	Offset/Index – Also equal to file downloaded so far.
varies	Part of file	Data size should be (Param Data Len – 8)

As an example, suppose the firmware size is 16K and we aim to transmit it in 4K segments. At the command level:

```
0711 1008 00004000 00000000 <first 4k of firmware file>
0711 1008 00004000 00001000 <2nd 4k>
0711 1008 00004000 00002000 < 3rd 4k>
0711 1008 00004000 00003000 <last 4k>
```

For iOS, the framing and SLIP bytes should be added. For USB HID, each large command must be segmented into smaller packets, following the same process used for breaking down any other large command (using the V5 extended command format)

4.7 Command 0x0712 – Update Firmware from File

Table 4-15 Update Firmware

Command	Parameter Data Len	Parameter Data
0x0712	0x0000	none

This command initiates the device's switch to the bootloader for flash updating. It is expected to fail and return an error under two conditions:

1. If the device has not received an authenticated firmware file or,

2. If the header in the received firmware file indicates that it is an older version than the firmware currently programmed.

5 Response Result Codes

All result codes must be both universal and functional. The code 0x0396 signifies invalid date or time data, indicating that the date or time has not been set. Subsequent attempts to set the property will fail with result code 0x07, indicating a sequence error. The response may be delayed.

Table 5-1 Response Result Codes

Value	Result Code	Description
0x0000	Success	The command completed successfully.
0x0001	Failure	The command failed.
0x0002	Bad Parameter	The command failed due to a bad parameter or command syntax error.
0x0003	Redundant	The command is redundant.
0x0004	Bad Cryptography	A bad cryptography operation occurred.
*0x0005	Delayed	The request is refused because the device is delaying requests as a defense against brute-force hacking.
0x0006	No Keys	No keys are loaded.
0x0007	Invalid Operation	Depends on the context of the command.
0x0008	Response not available	The response is not available.
0x0009	Not enough power	The battery is too low to operate reliably.
0x000D	Not implemented	The command is not implemented.
0x000E	Unarmed tamper, device not ready (Tamper Only)	The tamper device is not ready to be armed.
0x000F	Unarmed tamper, bad signature (Tamper Only)	The tamper is not armed because of a bad signature.
0x0396		
0x0080	DSN not found (in the device)	
0x0081	Incorrect DSN	
0x0082	Max token count reached	
0x0083	Response data length error	
0x0084	Incorrect DSN	
0x0085	Challenge token timed out	
0x0086	Invalid challenge token	
0x0087	Message verification failed	
0x0088	Invalid ECC key format	
0x0089	ECC key format not supported	
0x0085	Challenge token timed out	
0x0086	Invalid challenge token	
0x0087	Message verification failed	
0x0088	Invalid ECC key format	
0x0089	ECC key format not supported	
0x0085	Challenge token timed out	
0x0086	Invalid challenge token	
0x008A	Invalid key block version ID	
0x008B	Key block version not supported	
0x008C	Invalid key usage	
0x008D	Key usage not supported	
0x008E	Invalid algorithm	
0x008F	Algorithm not supported	
0x0090	Invalid mode use	

Value	Result Code	Description
0x0091	Mode use not supported	
0x0092	Key version not supported	
0x0093	Invalid export	
0x0094	Export not supported	
0x0095	Invalid optional block ID	
0x0096	Optional block ID not supported	
0x0097	Invalid KCV algorithm	
0x0098	KCV algorithm not supported	
0x0099	Invalid HIMAC hash algorithm	
0x009A	HMAC Hash algorithm not supported	
0x009B	TR-31 format error	
0x009C	MagTek custom optional block not found	
0x009D	Key Environment not found in Opt Blk	
0x009E	Key Environment not supported	
0x009F	Key ID not found in Opt BlIk	
0x00A0	Key ID not supported	
0x00A1	Key ID of TK not found in Opt Blk	
0x00A2	Transport key not found in key slot	
0x00A3	Wrong transport key (relationship)	
0x00A4	Key Restriction not found in Opt BIk	
0x00A5	Invalid key type restriction	
0x00A6	Invalid data type restriction	
0x00A7	DSN not found in Opt BIk	
0x00A8	Challenge token not found in Opt BIk	
0x00A9	Expiration date/time not found in Opt BIk	
0x00AA	KCV verification failed	
0x00AB	MAC verification failed	
0x00AE	Establish Ephemeral KBPK command is required	
0x00AF	Temporary KBPK not found	
0x00BO	Key ID doesn't match with Cipher Encryption Algorithm property setting.	
0x00B1	Key already exists in the device	
0x00B2	MTK deletion not allowed	
0x00B3	Key doesn't match with the existing key	
0x00B4	Incorrect key environment	
0x00A0	Key ID not supported	
0x00AC	Key ID not found (in the device)	
0x00A0	Key ID not supported	
0x00AC	Key ID not found (in the device)	
0x00AB	MAC verification failed	
0x00Ad	Invalid key configuration	
0x00C0	Error from UCL library	
0x00C1	Failed to save key in NVS	
0x00C2	Key self-check failure	
0x0010	ERR_SequenceNumber	Wrong firmware SequenceNumber
0x0011	ERR_FileID	Wrong firmware FileID
0x0012	ERR_ProductType	Wrong firmware ProductType
0x0013	ERR_OperationType	Wrong firmware OperationType
0x0014	ERR_SignatureLength	Wrong firmware SignatureLength
0x0015	ERR_SignatureMethod	Wrong firmware SignatureMethod

Value	Result Code	Description
0x0016	ERR_CommType	Wrong Interface type
0x0017	TamperTrig	Device tamper triggered while downloading firmware
0x0018	ERR_FwCompareHash	Firmware hash comparison failed
0x0019	Invalid_Iap2offsetadd	iap2 mode update firmware without downloading firmware, error
0x0020	ERR_FlashWrite	Failed to write flash

6 Properties

6.1 About Properties

Properties are used to provide information about the device and how to configure it. Secured properties are set at the factory or by an administrator using software tools supplied by MagTek. Property values take effect immediately unless specified otherwise.

6.2 Property 0x00 – Main Firmware ID

Property Description	
Property ID:	0x00
Property Type:	ASCII
Length:	Varies
Get Property:	Yes
Set Property:	No
Default Value:	Part number of installed firmware

This read-only property returns the main firmware part number, a dash, the major and minor revision number, followed by a dash and the firmware type. {main firmware part number}-{3 character rev}-{type}

Table 6-1 - 0x00 –Main Firmware ID Property Example

Example
1000004854-AD9-PCI

6.3 Property 0x02 - USB Polling Interval

Property Description	
Property ID:	0x02
Property Type:	Byte
Length:	1 byte
Get Property:	Yes
Set Property:	Yes
Default Value:	0x01

This one-byte value (1-255) sets the device's polling interval in milliseconds for the **Interrupt** in Endpoint. The device sends the value of this property as part of USB device enumeration to the host.

6.4 Property 0x03 - Device Serial Number

Property Description	
Property ID:	0x03
Property Type:	ASCII
Length:	7
Get Property:	Yes
Set Property:	No
Default Value:	N/A

This value is also found on the product label. The property contains the 7-character MagTek device serial number. This value is used for USB device enumeration, data messages, and message security.

6.5 Property 0x04 - MagneSafe Version Number

Property Description	
Property ID:	0x04
Property Type:	String
Length:	0 - 7 bytes
Get Property:	Yes
Set Property:	No
Standard Value:	PCIV01

This is a maximum 7-byte read-only property that identifies the MagneSafe Feature Level supported on this device.

6.6 Property 0x05 - Track ID Enable

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

This property is defined as follows:

Table 6-2 Track Enable Property

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 (Generates error if track is blank)

6.7 Property 0x07 - ISO Track PAN Mask

Property Description	
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:

Table 6-3 ISO Track PAN Masking

Length	Description
2	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 “08”.
2	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 “04”.
1	**Masking Character.** This byte specifies which character the device uses for masking.
1	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”.

6.8 Property 0x08 - AAMVA Track Mask

Property Description	
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:

Table 6-4 - Factors for Masking Data on AAMVA Type Cards

Length	Description
2	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	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".
1	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
1	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".

6.9 Property 0x10 - Interface Type

Property Description	
Property ID:	0x10
Property Type:	Byte
Length:	1 byte
Get Property:	Yes
Set Property:	Yes (No for devices that switch connections automatically)
Default Value:	Dependent on device type:

This property represents the device's current connection type.
Valid values for this property are:

- 0x00 = USB HID (HID Only)
- 0x01 = USB Keyboard Emulation (KB) (USB KB Only)
- 0x02 = iAP2
- 0xFF = One-Time Automatic (HID Only | iAP2 Only). When the property is set to this value and the device connects to a host, the device attempts to determine which interface type the host is using. After it successfully detects the interface type, it automatically sets this property to the value that corresponds to that interface type.

6.10 Property 0x1E – Start of Message

Property Description	
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.

6.11 Property 0x22 – End of Message

Property Description	
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 value of this property to the host at the end of the data message. For example, if the host software requires a set of keystrokes to end the process of receiving card data, this property could be set to transmit that keystroke sequence. If the value is 0, the device does not send a termination string.

6.12 Property 0x23 - Field Separator

Property Description	
Property ID:	0x23
Property Type:	Character
Length:	1 byte
Get Property:	Yes
Set Property:	Yes
Default Value:	0x7C ('|')

If the value is 0, the device does not send a delimiter which is not recommended.

6.13 Property 0x24 - Start Sentinel Track 1 (ISO)

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

The device uses this character for the Track 1 start sentinel when it recognizes the track is encoded in the standard ISO format for Track 1.

6.14 Property 0x25 - Start Sentinel Track 2 (ISO ABA)

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

The device uses this character for the Track 2 start sentinel when it recognizes the track is encoded in the standard ISO format for Track 2.

6.15 Property 0x26 - Start Sentinel Track 3 (ISO ABA)

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

The device uses this character for the Track 3 start sentinel when it recognizes the track is encoded in the standard ISO format for Track 3.

6.16 Property 0x27 - Start Sentinel Track 3 (AAMVA)

Property Description	
Property ID:	0x27
Property Type:	Byte
Length:	1 byte
Get Property:	Yes
Set Property:	Yes

Property Description	
Default Value:	0x23 (‘#’)

The device uses this character for the Track 3 start sentinel when it recognizes the track is encoded in the standard Track 3 AAMVA format.

6.17 Property 0x28 - Start Sentinel Track 2 (7-bit)

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

The device uses this character for the Track 2 start sentinel when it recognizes the track is encoded in the 7-bit ISO format, normally used for Track 1.

6.18 Property 0x29 - Start Sentinel Track 3 (7-bit)

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

The device uses this character for the Track 3 start sentinel when it recognizes the track is encoded in the 7-bit ISO format, normally used for Track 1.

6.19 Property 0x2B – Track End Sentinel

Property Description	
Property ID:	0x2B
Property Type:	Byte
Length:	1 byte
Get Property:	Yes
Set Property:	Yes
Default Value:	0x3F (‘?’)

The device uses this character for all track end sentinels.

6.20 Property 0x31 - Mask Other Cards

Property Description	
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 it’s not a QWANTUM card, 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.

6.21 Property 0x34 – Mask AAMVA Card Data

An AAMVA card is typically used for Drivers Licenses and ID cards.

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

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

6.22 Property 0x3A – Boot Firmware ID

Property Description	
Property ID:	0x3A
Property Type:	ASCII
Length:	varies
Get Property:	Yes
Set Property:	No
Default Value:	N/A

This read-only property returns the boot firmware part number, a dash, the major and minor revision number, followed by a dash and the firmware type.

```
{boot fw pn}-{rev}-PCI
Example: 1000004854-AA0-PCI
```

6.23 Property 0x53 - Inter-Key Delay (KB Emulation)

Property Description	
Property ID:	0x53
Property Type:	Byte
Length:	1 byte
Get Property:	Yes
Set Property:	Yes
Default Value:	0x04 (4 ms)

This property controls how long the device pauses between each key report. This delay can be adjusted between 0 and 250 milliseconds. Some host devices cannot handle full speed keyboard input without dropping key presses, so delays can be added. The time needed to send the entire message also increases.

6.24 Property 0x80 – Device Hardware ID

Property Description	
Property ID:	0x80
Property Type:	ASCII
Length:	varies
Get Property:	Yes
Set Property:	No
Default Value:	N/A

Table 6-5 - 0x80 – Device Hardware ID Example

Example
10PCI50U0BA

6.25 Property 0x82 – Time of Day Reset

Property Description	
Property ID:	0x82
Property Type:	Byte
Length:	2 bytes
Get Property:	Yes
Set Property:	Yes (New value takes effect after reset)
Default Value:	0000 (UTC Midnight)

For security purposes, the reader will reset itself once a day. You can choose what time this reset will occur by changing this property. Set the value to HHMM in Coordinated Universal Time (UTC). HH is one byte with a value of 0-23. MM is one byte with a value of 0-59 The new value takes effect after the next device reset or power cycle.

Table 6-6 PCI Time of Day Reset Value

Length	Field Name	Value
1	Hour	Value from 0x00...0x17 (0-23)
1	Minute	Value from 0x00...0x3B (0-59)

6.26 Property 0x84 – Device Model Name

Property Description	
Property ID:	0x84
Property Type:	ASCII
Length:	varies
Get Property:	Yes
Set Property:	No
Standard Value:	'iDynamo 5 Gen III'

For PCI certified devices, this value will match the one found on the PCI website and the product label.

6.27 Property 0x86 – Device State

Property Description	
Property ID:	0x86
Property Type:	Binary
Length:	4 bytes
Get Property:	Yes
Set Property:	No – values come from self-test and security status.
Default Value:	N/A

Note that the device is considered offline (no card reading) until all 4 bytes are zero.

Table 6-7 Device State Definition

Byte	Type	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1	System State	Offline	0	0	Device Keys missing	RTC not active/set	Not Initialized	Security Inactive	Tampered
2	Self-Test failure	Data Integrity	Key Integrity	0	0	0	Registers	Crypto	RNG
3	PCI Info	0	0	0	0	0	HW ID Missing	Model Name Missing	Serial # Missing
4	Unmapped Data Types	0	0	0	0	0	0	0	0

6.27.1 BYTE Description

The primary functions of device states are to inform production software about completed steps and pending tasks. It serves as a checklist and can also assist with remote services and RMAs.

Table 6-8 - BYTE Description

Bytes	Bits	Name	Settings
BYTE 1 – System State	Bit 7	Offline	0 = All bits in this property are set to 0 1 = Not all bits in this property are 0
	Bit 6	Reserved	Always 0
	Bit 5		
	Bit 4	Device Keys Missing	0 = Device keys present 1 = Device keys missing NOTE: "Device Keys" refer to the keys that are injected during MfgCfg such as Transport Keys and MAC Keys. "Device Keys" do not include DUKPT keys (2007, 2002 and 2003). The DUKPT keys are "Financial Keys".
	Bit 3	RTC not Active/Set	0 = RTC active 1 = RTC inactive (not enabled)
	Bit 2	Not Initialized	0 = Non-volatile storage initialized 1= Non-volatile storage not initialized NOTE: "Initialized" means: a. the master keys are generated internally. b. the data & key storage area is initialized with an initialization indicator such as 0xAA55 and its integrity is known good.
	Bit 1	Security Inactive	0 = Security activated 1= Security not activated NOTE: "Security" means Tamper.
	Bit 0	Tampered	0 = Device not tampered 1 = Device tampered
BYTE 2 – Self-Test Failure	Bit 7	Data Integrity	0 = Configuration data in NVS not corrupted 1 = Found data corruption in the configuration data
	Bit 6	Key Integrity	0 = All fixed keys and injected keys are not corrupted 1 = Found key data for one or more keys is corrupted
	Bit 5	Reserved	Always 0

Bytes	Bits	Name	Settings
	Bit 4		
	Bit 3		
	Bit 2	Registers	0 = All hardware registers in uC required for normal operation of the device are initialized/ enabled/configured/ functional; 1 = Any one or more of those registers are not operating correctly
	Bit 1	Crypto	0 = Passed test 1 = Failed test NOTE: Enable Crypto engine in uC and test crypto functions and make sure it returns the expected results. (e.g. KAT – Known Answer Test)
	Bit 0	RNG	0 = Passed test 1 = Failed test NOTE: Enable RNG in uC and read RNG value a few times and make sure RNG returns different values for each read.
BYTE 3 – PCI Info	Bit 7	Reserved	Always 0
	Bit 6		
	Bit 5		
	Bit 4		
	Bit 3		
	Bit 2	PCI HW ID Missing	0 = Found a valid PCI HW ID in Property 0x80 1 = PCI HW ID missing
	Bit 1	Model Name Missing	0 = Found a valid model name in Property 0x84 1 = Model name missing
	Bit 0	Serial # Missing	0 = Found a valid DSN in Property 0x03 1 = DSN missing
BYTE 4 – Unmapped Data Types	Bit 7	Reserved	Always 0
	Bit 6		
	Bit 5		
	Bit 4		
	Bit 3		
	Bit 2		
	Bit 1		
	Bit 0		

6.28 Property 0x87 – Key map status

Property Description	
Property ID:	0x87
Property Type:	Binary
Length:	8 bytes
Get Property:	Yes
Set Property:	No – values come from key injection.
Starting Value:	0 – if no keys loaded

If a key has been loaded into the device, then it's bit indicator should be set to 1.

Table 6-9 Key Map Status Bits

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
1 – TK1	0	MKIFTK	MFGTK	PRODTK	FINTK	DEVTK	MTK	TMPTK
2 – TK2	0	0	0	0	0	0	0	0
3 – MK1	0	0	0	0	0	MFRQMK	MREQMK	FREQMK
4 – MK2	0	0	0	0	0	0	0	0
5 – DK1	DKPTM7	0	0	0	DKPTM3	DKPTM2	0	0
6 – DK2	0	0	0	0	0	0	0	0
7 – DK3	0	0	0	0	0	0	0	0
8 – DK4	0	0	0	0	0	0	0	0

A fully loaded reader would return a keymap of: 7E 00 07 00 8C 00 00 00.

6.29 Property 0x88 USB Packet Delay

Property Description	
Property ID:	0x88
Property Type:	Binary
Length:	1 byte
Get Property:	Yes
Set Property:	Yes
Starting Value:	0x00 (0 ms)

The host can use this property to adjust the device's USB packet delay. The property can be set to 0 to have no delay, or it can be set to a specific value in the range of 1 to 255 ms.

- Adjusting this value does not require security when the device is fully configured.
- Adjusting this value does not affect the USB HOST POLL TIMEOUT, PROPERTY 0x52. In fact, it is mostly independent from the polling interval.

MagTek recommends the following values for various platforms:

- Windows = 0x00 (0 ms)
- Android = 0x19 (25 ms)
- Linux = 0x32 (50 ms)

Appendix A Warranty, Standards and Certifications

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.