MAGTEK®
SECURITY FROM THE INSIDE

DynaProx Products

EMV Contactless/NFC Card Readers
Programmer's Manual (COMMANDS)



April 2025

Document Number:
D998200489-102

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

Copyright © 2006 - 2025 MagTek, Inc.
Printed in the United States of America

INFORMATION IN THIS PUBLICATION IS SUBJECT TO CHANGE WITHOUT NOTICE. MAGTEK CANNOT BE HELD LIABLE FOR ANY USE OF THE CONTENTS OF THIS DOCUMENT. ANY CHANGES OR IMPROVEMENTS MADE TO THIS PRODUCT WILL BE INCLUDED IN THE NEXT PUBLICATION RELEASE. IF YOU HAVE QUESTIONS ABOUT SPECIFIC FEATURES AND FUNCTIONS OR WHEN THEY WILL BECOME AVAILABLE, PLEASE CONTACT YOUR MAGTEK REPRESENTATIVE.

MagTek®, MagnePrint®, and MagneSafe® are registered trademarks of MagTek, Inc.
Magensa™ is a trademark of MagTek, Inc.

AAMVA™ is a trademark of AAMVA.
American Express® and EXPRESSPAY FROM AMERICAN EXPRESS® are registered trademarks of American Express Marketing & Development Corp.
D-PAYMENT APPLICATION SPECIFICATION® is a registered trademark of Discover Financial Services CORPORATION
MasterCard® is a registered trademark and PayPass™ and Tap & Go™ are trademarks of MasterCard International Incorporated.
Visa® and Visa payWave® are registered trademarks of Visa International Service Association.

ANSI®, the ANSI logo, and numerous other identifiers containing "ANSI" are registered trademarks, service marks, and accreditation marks of the American National Standards Institute (ANSI).
ISO® is a registered trademark of the International Organization for Standardization.
UL™ and the UL logo are trademarks of UL LLC.
PCI Security Standards Council® is a registered trademark of the PCI Security Standards Council, LLC.
EMV® is a registered trademark in the U.S. and other countries and an unregistered trademark elsewhere. The EMV trademark is owned by EMVCo, LLC. The Contactless Indicator mark, consisting of four graduating arcs, is a trademark owned by and used with permission of EMVCo, LLC.
The *Bluetooth*® word mark and logos are registered trademarks owned by Bluetooth SIG, Inc. and any use of such marks by MagTek is under license.

Google Play™ store, Google Wallet™ payment service, and Android™ platform are trademarks of Google LLC.
Apple Pay®, iPhone®, iPod®, Mac®, and OS X® are registered trademarks of Apple Inc., registered in the U.S. and other countries. iPad™ is a trademark of Apple. Inc. App Store℠ is a service mark of Apple Inc., registered in the U.S. and other countries. IOS is a trademark or registered trademark of Cisco in the U.S. and other countries and is used by Apple Inc. under license.
Microsoft®, Windows®, and .NET® are registered trademarks of Microsoft Corporation.
MIFARE, the MIFARE logo, MIFARE Ultralight, MIFARE Plus, MIFARE Classic, MIFARE FleX, DESFire, and MIFARE4Mobile are registered trademarks of NXP B.V.

All other system names and product names are the property of their respective owners.

Table 1.1-1 - Revisions

Rev Number	Date	Notes
10	Oct 11, 2021	Initial DRAFT release from master rev 30v74 plus in-place corrections.
30	Nov 11, 2021	Release derived from master rev 30
50	Sep 8, 2022	Generate Programmers manual from D100005000 rev 50 Release to Rev 50 from rev 31
60	March 6, 2023	Generate Programmers manual from D100005000 rev 60.
61	March 27, 2023	Generate Programmers manual from D100005000 rev 61.
62	April 7, 2023	Updated **Table 3.2-4** Request Operation Failed, Modified value of DFDF1B in **Table 4.7-1 – EMV Batch Data (DynaPro Format) Type** ; Update DF1B Example Value in **Table 4.10-2** Updated Table 6.1-9 - Response Example**,** Table 6.1-13 - Response Example**, Update Request and Response examples in** Command 0xD111 - Set Property**, Add note "For additional support, please contact MagTek Support." To sections 6.6.3, 6.6.4, 6.6.5, 6.6.6, 6.7.1, 6.9.2, 6.9.3, 6.9.4, 6.10.1; Add Appendix C Erasing EMV Configurations,** Updated **Table 3.2-4 - Operation Status Detail Codes** with 02 Requested Operation Failed; Change examples in **Command 0xD101 - Get Property** and **Command 0xD111 - Set Property** Misc. clarifications and corrections
63	May 16, 2023	Added definition of 00 03 00 00 in **7.3.1 Notification 0x1001 - Device Information Update** ; Added NFC UID support in **4.16 NFC UID Type (EMV Contactless Only) , Table 7.1-1 - Notification Detail Codes** , **Table 7.1-2 - Notification Payload for Data Update, ARQC Update (Quick Chip), Data Attached , Table 7.1-7 - Notification Detail Codes;** add detection and removal user event notification in **7.3.1 Notification 0x1001 - Device Information Update** ; Add **Property 2.3.1.2.1.5 Battery Charger Status** Clarified Message size in **Table 2.1-2 - Multi-Packet Head Format** ; Add DynaFlex II Go USB PID and Get/Set Property command, Device Family ID, Product ID; Updated notification code for Battery Charge complete to 00 02 02 00 in **Table 7.3-1 - Notification Detail Codes** ; updated **Table 7.3-1 - Notification Detail Codes** to distinguish 1K or 4K Mifare Classic cards; Add DynaFlex II Go on page 1; Add NFC pass through flow description in **Command 0x1001 - Start Transaction**; Add BCR encryption bit to **8.3.6.4 Property 1.2.7.1.2.1 User Event Notification Controls Enable;** Updated NFC Removal Code from - x50 to 0x05 in **Table 7.1-7 - Notification Detail Codes** ; **Table 7.3-1 - Notification Detail Codes**, Add **Table 7.3-7 - Notification Payload for UID**, **Table 7.3-8 – Notification Payload for UID Example**; Misc. clarifications and corrections;
64	September 28, 2023	Add tags 82 and 9F6E in the clear text portion of the ARQC Message - **EMV ARQC (DynaPro Format) Type** ; Removed Tag 84 on these payloads message to match firmware **Table 7.4-2, Table 7.4-3**; Add **24 Hour Automatic Reset PCI Requirement** and **Property 1.2.7.1.1.3 Device Reset Will Occur Soon Notification Control**; Misc. clarifications and corrections.

Rev Number	Date	Notes
65	October 10, 2023	Add Extended AID Selection Feature - **7.4.1 Notification 0x1803 - User Interface Host Action Request and 8.3.1.1 Property 1.2.1.1.1.2 Application Selection Behavior**; Add support for Mifare Classic 1K and 4K - **6.2.2 Command 0x1101 – Pass Through Command for MIFARE Classic;** Update Key Injection Sequence in **Command 0xEF01 - Load Key Using TR-31** added **Table 6.9-7 - Device Key ID / Slot**
66	April 24, 2024	Add Section **8.4.1.4 Property 2.1.2.1.2.4 Boot0 Firmware Part** Number; Update section **4.3 Display Strings** to include missing strings. Add reference to **Command 0xE001 - Get Challenge** to **Command 0xEF01 - Load Key Using TR-31,** Update Table 4.18-6 - Key Slot IDs, change ID from 200x to 0x2000 to 0x201F , Update **0 DUKPT Slot IDs** with TR31 Module support from 16 to 32 and Slot IDs to 0x2000 to 0x201F, Update **Table 8.6-4 - Property 2.3.1.1.1.2 Transaction Key Status** with updated Valid Values, Add **Property 2.3.1.2.1.6 Device Temperature**. Add **Property 1.2.7.1.4.1 Device Low Temperature Notification Level**. Add **Property 1.2.7.1.4.2 Device High Temperature Notification Level**. Add **Property 1.2.7.1.4.3 Device Temperature Notification Repeat Interval**. Add **Section 6.2.3 Command 0x1102 – Pass Through Command for MIFARE DESFire, Type 4**, Update section **7.1.1 Notification 0x0101 - Transaction Information Update** to include Mifare DESfire info, Update section **7.3.1 Notification 0x1001 - Device Information Update** to include Mifare DESFire info, Add **Property 1.1.1.1.1.10 Google Smart Tap Collector ID Slot 1** through **Property 1.1.1.1.1.15 Google Smart Tap Collector ID Slot 6**. Add **Property 1.1.1.1.1.1A Google Smart Tap POS Capabilities**.Add Google Smart Tap to **Notification 0x0101 – Transaction Information Update**. Add Google Smart Tap data to **EMV ARQC Type** and **EMV Batch Data Type**. Add **Property** 1.2.2.2.1.2 USB Configuration Type; Add **Property 1.1.1.1.1.D Apple iAP2 AppBundleID;** Revised **5.1.1 MACs for EMV Data** to add AES method and simplified TDES MACing spec by referencing ANSI x9.24; Add **Command 0x1805 - Buzzer** and update **Notification 0x1805 - User Interface Operation** Complete to include Buzzer result info, add **Property 1.2.1.1.3.1 Contactless Low Power Card Detect (Contactless Only),** Clarified max number of leading unmasked characters **Property 1.1.2.2.1.3 PAN Number of Leading Unmasked Characters** ; Added **Appendix G - Physical Button (DynaFlex Only);** Added low battery error to Table 3.2-4 in **Response Message**; Added event notifications are disabled when battery is 5% or less in **Property 1.2.7.1.2.1 User Event Notification Controls Enable**, Added note that firmware can not be updated when the batter charge is less than or equal to 5 percent in **Command 0xD801 - Load Firmware File**; Add definition of Tag DF29 in **Table 4.5-2 - EMV ARQC (DynaPro Format) DFDF59 Decrypted Contents** and **Table 4.7-2 – EMV Batch Data (DynaPro Format) DFDF59 Decrypted Contents** ; Misc. clarifications and corrections; Added buzzen in progress error to Table 3.2-4 in **Response Message**; Add: Request file from MagTek services Set: No

Rev Number	Date	Notes
100	September 3, 2024	Add Apple VAS and GWST in **Table 1.6-1 - Device Features**; Add bit definitions to enable Apple / Google VAS in **Command 0x1001 - Start Transaction**; Add tag /83 to **Table 6.1-1 - Request Data for Command 0x1001 - Start Transaction**; Add **Table 7.4-7 – Default User Interface String IDs and Strings**; Add **Property 1.1.1.1.4.1 EMV Configuration Filename**; Add **Property 1.1.1.1.4.2 CA Public Key Configuration Filename**; Add **Property Subgroup 2.1.2.8.nn Custom** UI**;** Add NFC Tag Support Flow and Audio Transducer Beep Flow to **6.1.1 Command 0x1001 - Start Transaction;** Update references for Tag 81 in Table **6.2-1 - Request Data for Command 0x1100 – Pass Through Command For NTag/MIFARE Ultralight, Type 2.**; Update table caption to "NTag Commands" in **Table 6.2-2 – NTag Commands**; Add **Table** 6.2-3 – MIFARE Ultralight EV1 Commands and **Table 6.2-4 – MIFARE Ultralight C Commands**, Add **6.2.1.1 Encrypted Data Format** and **6.2.2.1 Encrypted Data Format** and **6.2.3.1 Encrypted Data Format**; Add **8.3.1.4 Property 1.2.1.1.4.1 MIFARE Ultralight C 2keys3DES**; Update Notificaton detail codes to include Ntag/MIFARE Ultralight in **Table 7.1-1 - Notification Detail Codes** and **Table 7.3-1 - Notification Detail Codes**, add **Table 6.1-3 - Response Example for Command 0x1001 – Start Transaction Command not executed due to Battery Charge State**, Update **Table 3.2-3 - Response Message Format** with **Table 6.6-4 - Response Example for Command 0xD801 Battery Charge State** battery Charge Status content, Update **Table 3.2-5 - Notification Message Format** with battery charge status content; added **4.16 NFC UID Type (EMV Contactless Only)** , add notification for GPO Response in **7.1.1Notification 0x0101 - Transaction Information Update**, add additional note on Tag 85 regarding GPO Response = 6985 on **Table 7.4-2 - Notification Payload for Display Message Request Notifications (No Display Only):** Add **Table 6.2-4 – MIFARE Ultralight C Commands, Table** 6.2-3 – MIFARE Ultralight EV1 Commands**, Table 6.2-5 – MIFARE Ultralight AES Commands,** Add **Property 1.2.1.1.4.2 MIFARE Ultralight AES DataProtKey.Property 1.2.1.1.4.3 MIFARE Ultralight AES UIDRetrKey. Property 1.2.1.1.4.4 MIFARE Ultralight AES OriginalityKey.** Add note for random ID. Add **6.6.7 Command 0xD831 – Delete File from Device.** Add **Command 0xD112 - Set Property (Secured)**, Add **2.2 How to Use Apple iAP2 Connections (iAP2 Only);** update **Table 4.5-1 - EMV ARQC (DynaPro Format) Type** with encrypted card data content; Add **Table 4.5-3 - EMV ARQC (DynaPro Format) DFDFDF37 Decrypted Contents**; Update Table 1.5-1 - Device Connection Types / Data Formats to include iAP2; add **2.2 How to Use Apple iAP2 Connections (iAP2 Only)**

Rev Number	Date	Notes
101	January 20, 2025	Updated HW configuration values for USB/iAP2. Add **Property 2.1.2.5.6.7 Available Access Points** ; Add support for CUP, JCB and Interac in - **EMV Processing Configuration File Type** and **EMV Entry Point Configuration File Type** ; Misc. clarifications and corrections; updated **Notification 0x0101 - Transaction Information Update** and **Notification 0x1001 - Device Information Update** to support MIFARE Plus. Add tags to **ARQC** (cleartext): 0xDF2A – Tip Mode Sale Amount Entered 0xDF2B – Tip Mode Total Amount Add tag to **OID property 1.1.1.1.3.4 – Other TLV**: 0x9F02 – Default Amount for Quickchip Add to CA revocation list, exception file list and DPAS data storage to **Table 6.6-1 - File Types**; Delete **Command 0xD112 - Set Property (Secured)** request data, response data and example; Add **Property 2.1.2.3.6.5 Entry Point Checksum** and **Property 2.1.2.3.A.5 Entry Point Checksum.** Added **Table 4.5 9 – Fleet Data Container Payload.;** Removed Enhanced ARQC section; Removed Enhanced Batch Data Section; Add "common kernel only" tag to Fleet, JCB, CUP and Interac OIDs. replaced radar mode with updated terminology changed to Proximity Detection Mode

Rev Number	Date	Notes
102	April 15, 2025	Make **Device Lock Feature** customer facing; Add **Property 1.1.1.1.2.6 Tip Mode Enable Submit on Amount Button Press**; Add response codes to group 2 subgroup 3: **Attempt to display Flexible UI page while touchscreen UI notifications enabled** and Request is invalid while card emulation is in progress to in **Table 3.2-4 - Operation Status Detail Codes;** Updated **Property 1.2.3.2.1.1 System Volume Control** to allow value of 0 (no sound); Added Apollo client cert support; Add Card Emulation to **Table 1.6-1 - Device Features**; Add **Card Emulation**; add **Command 0x1840 – Card Emulation,** add Module 0x06 Card Emulation to **Table 7.4-4 - Notification Detail Codes**. Update **Command 0x1102 – Pass Through Command for MIFARE DESFire, Type 4** to include MIFARE DESFire EV1/EV2/EV3; Update **Command 0x1102 – Pass Through Command for MIFARE DESFire, Type 4** to include MIFARE Plus EV1/EV2/SE/X SL1 (Security Level 1) commands; Update **Command 0x1103 – Pass Through Command for MIFARE Plus, Type 2** to include MIFARE Plus EV1/EV2/SE/X SL3 (Security Level 3) commands; Update **Notification 0x0101 - Transaction Information Update** and **Notification 0x1001 - Device Information Update** to include MIFARE DESFire EV1/EV2/EV3; Add **Property 1.2.1.1.4.5 MIFARE Plus AES_Key1. Property 1.2.1.1.4.6 MIFARE Plus AES_Key2. Property 1.2.1.1.4.7 MIFARE Plus AES_Key3. Property 1.2.1.1.4.8 MIFARE Plus AES_Key4. Property 1.2.1.1.4.9 MIFARE Plus AES_Key5. Property 1.2.1.1.4.A MIFARE Plus AES_Key6.** Update **Command 0x1100 – Pass Through Command For NTag/MIFARE Ultralight, Type 2** to include an option for READ command to do a fast read.

Table of Contents

Table of Contents 8
1 Introduction 17
1.1 About This Document 17
1.2 About This Document 17
1.3 About Terminology 18
1.4 About SDKs and Sample Code 18
1.5 About Connections and Data Formats 20
1.6 About Device Features 21
2 Connection Types 23
2.1 How to Use USB Connections (USB Only) 23
2.1.1 About USB Reports, Usages, Usage Pages, and Usage IDs 23
2.1.2 How to Connect to a Device Using the USB Connection 23
2.1.3 How to Send Command Requests Using the USB Connection 24
2.1.4 How to Receive Data Using the USB Connection (HID Only) 26
2.2 How to Use Apple iAP2 Connections (iAP2 Only) 27
3 Messages (Requests, Responses, Notifications, and Files) 28
3.1 About Messages 28
3.2 Message Format 29
3.2.1 Tag-Length-Value (TLV) Encoding 29
3.2.1.1 About TLV Encoding 29
3.2.1.2 TLV Example 30
3.2.1.3 How to Read TLV Tables 31
3.2.2 Message Structure 32
3.2.2.1 Request Message 32
3.2.2.2 Response Message 33
3.2.2.3 Notification Message 38
3.2.2.4 Data File Message 39
4 Data Types and Shared TLV Data Objects 41
4.1 Primitive Data Types 41
4.2 About Track Data 41
4.3 Display Strings 42
4.4 Encryption Type 44
4.5 EMV ARQC Type 44
4.5.1 EMV ARQC (DynaPro Format) Type 45
4.6 EMV ARPC Type 54
4.7 EMV Batch Data Type 54
4.7.1 EMV Batch Data (DynaPro Format) Type 55
4.7.2 Merchant Data Container 58

4.8 EMV Terminal Configuration File Type 58
4.9 EMV Processing Configuration File Type 60
4.10 EMV Entry Point Configuration File Type 62
4.10.1 Mastercard MCL Entry Point Table 63
4.10.2 Visa payWave Entry Point Table 66
4.10.3 American Express Expresspay Entry Point Table 69
4.10.4 Discover D-PAS Entry Point Table 71
4.10.5 China Unionpay Entry Point Table (Common Kernel Only) 73
4.10.6 JCB Entry Point Table (Common Kernel Only) 75
4.10.7 Interac Flash Entry Point Table (Common Kernel Only) 77
4.11 EMV Configuration CA Public Keys File Type 78
4.12 EMV American Express DRL Configuration File Type (Not Supported on Expresspay 4.x) 80
4.13 Security Operation Type 80
4.14 Security Parameters Type 83
4.15 Key Information Type 84
4.16 NFC UID Type (EMV Contactless Only) 84
4.17 GPO Response Type (EMV Contactless Only) 84
4.18 TR-31 Key Block Type 84
4.18.1 DUKPT Key Mapping 87
4.18.1.1 Terms and Definitions 87
4.18.2 Restrictions of a DUKPT Slot ID 88
4.18.3 The Rules of Key Mapping 89
4.18.4 The settings of DUKPT Slot IDs injected through TR31 90
4.18.5 The Allowed Key Mapping Table 90
4.18.5.1 Examples of Key Mapping 90
4.19 Common File Structure 91
4.20 Certificate File Types 91
4.21 Certificate Signing Request (CSR) File Types 92
4.22 Card Emulation 92
4.22.1 Card Emulation Overview 92
4.22.2 Device Compatibility 92
5 Security 93
5.1 About Message Authentication Codes (MAC) 93
5.1.1 MACs for EMV Data 93
5.2 About Key Serial Numbers (KSNs) 93
5.3 About Encryption and Decryption 94
5.4 How to Determine the Key 94
5.5 How to Decrypt Data 94
5.6 24 Hour Automatic Reset PCI Requirement 95
5.7 Device Lock Feature 95

6 Commands 97
6.1 Command Group 0x10nn - Transactions 97
6.1.1 Command 0x1001 - Start Transaction 97
6.1.2 Command 0x1004 - Resume Transaction 103
6.1.3 Command 0x1008 - Cancel Transaction 104
6.2 Command Group 0x11nn - NFC/MIFARE Pass Through Commands (Contactless Only) 105
6.2.1 Command 0x1100 – Pass Through Command For NTag/MIFARE Ultralight, Type 2 105
6.2.1.1 Encrypted Data Format 117
6.2.2 Command 0x1101 – Pass Through Command for MIFARE Classic/MINI®/Plus SL1 (Security Level 1), Type 2 118
6.2.2.1 Encrypted Data Format 122
6.2.3 Command 0x1102 – Pass Through Command for MIFARE DESFire, Type 4 123
6.2.3.1 Encrypted Data Format 124
6.2.4 Command 0x1103 – Pass Through Command for MIFARE Plus, Type 2 124
6.2.4.1 Encrypted Data Format 137
6.3 Command Group 0x18nn - User Interface 137
6.3.1 Command 0x1802 - Report Cardholder Selection 137
6.3.2 Command 0x1804 - Read Barcode (BCR Only) 139
6.3.3 Command 0x1805 - Buzzer 140
6.3.4 Command 0x1840 – Card Emulation 141
6.4 Command Group 0x1Fnn - Device Control 143
6.4.1 Command 0x1F01 - Reset Device 143
6.4.2 Command 0x1F02 - Set Notification Subscriptions 143
6.5 Command Group 0xD1nn - Settings and Information 145
6.5.1 Command 0xD101 - Get Property 145
6.5.2 Command 0xD111 - Set Property (Unsecured) 149
6.5.3 Command 0xD112 - Set Property (Secured) 152
6.6 Command Group 0xD8nn - File Operations 152
6.6.1 About Files 152
6.6.2 Command 0xD801 - Load Firmware File 153
6.6.3 Command 0xD811 - Start Send File to Device (Secured) 155
6.6.4 Command 0xD812 - Start Send File to Device (Unsecured) 157
6.6.5 Command 0xD821 - Start Get File from Device 159
6.6.6 Command 0xD825 - Get File Info from Device 161
6.6.7 Command 0xD831 – Delete File from Device 162
6.7 Command Group 0xD9nn - Process Files 163
6.7.1 Command 0xD901 - Commit Firmware from File 163
6.8 Command Group 0xDFnn - Diagnostics and Utilities 165

6.8.1 Command 0xDF01 - Echo 165
6.9 Command Group 0xEnnn - Security 165
6.9.1 Command 0xE001 - Get Challenge 165
6.9.2 Command 0xEEEE - Send Secured Command to Device 167
6.9.3 Command 0xEF01 - Load Key Using TR-31 168
6.9.4 Command 0xEF06 – Change Device Lock State 170
6.9.5 Command 0xEF07 – Change Device Lock Passcode 171
6.9.6 Command 0xEF11 - Get Key Info 172
6.10 Command Group 0xFnnn - Manufacturing 174
6.10.1 Command 0xF017 - Establish Ephemeral KBPK 174
7 Notifications 178
7.1 Notification Source 0x01nn - Notifications from Transactions 178
7.1.1 Notification 0x0101 - Transaction Information Update 178
7.1.2 Notification 0x0105 - Transaction Operation Complete 183
7.2 Notification Source 0x09nn - Notifications from Firmware Updates 187
7.2.1 Notification 0x0905 - Firmware Update Successful 187
7.2.2 Notification 0x0906 - Firmware Update Failed 187
7.2.3 Notification 0x0907 - Firmware is Up to Date 187
7.3 Notification Source 0x10nn - Notifications from Device 188
7.3.1 Notification 0x1001 - Device Information Update 188
7.3.1.1 Notification 0x1001 – Low Battery Warning 194
7.3.1.2 Notification 0x1001 - Low Battery Shutdown 194
7.3.1.3 Notification 0x1001 - Battery Charged 195
7.4 Notification Source 0x18nn - Notifications from User Interface 196
7.4.1 Notification 0x1803 - User Interface Host Action Request 196
7.4.2 Notification 0x1805 - User Interface Operation Complete 198
8 Configuration 204
8.1 About Configuration 204
8.2 Property Group 1.1.nnnn Financial Settings 204
8.2.1 Property Subgroup 1.1.1.nnn EMV Settings 204
8.2.1.1 Property 1.1.1.1.1.1 EMV Terminal Identification 204
8.2.1.2 Property 1.1.1.1.1.2 EMV ARQC Message Tag List 205
8.2.1.3 Property 1.1.1.1.1.3 EMV Batch Data Tag List 206
8.2.1.4 Property 1.1.1.1.1.4 EMV Reversal Data Tag List (EMV Contact Only) 207
8.2.1.5 Property 1.1.1.1.1.5 ARPC Receive Timeout 208
8.2.1.6 Property 1.1.1.1.1.6 ARPC Retry Attempts 209
8.2.1.7 Property 1.1.1.1.1.7 Apple VAS Merchant ID and URL Slot 1 210
8.2.1.8 Property 1.1.1.1.1.8 Apple VAS Merchant ID and URL Slot 2 211
8.2.1.9 Property 1.1.1.1.1.9 Apple VAS Merchant ID and URL Slot 3 212

8.2.1.10 Property 1.1.1.1.1.A Apple VAS Merchant ID and URL Slot 4 213
8.2.1.11 Property 1.1.1.1.1.B Apple VAS Merchant ID and URL Slot 5 214
8.2.1.12 Property 1.1.1.1.1.C Apple VAS Merchant ID and URL Slot 6 215
8.2.1.13 Property 1.1.1.1.1.D Apple iAP2 AppBundleID 216
8.2.1.14 Property 1.1.1.1.1.10 Google Smart Tap Collector ID Slot 1 217
8.2.1.15 Property 1.1.1.1.1.11 Google Smart Tap Collector ID Slot 2 218
8.2.1.16 Property 1.1.1.1.1.12 Google Smart Tap Collector ID Slot 3 219
8.2.1.17 Property 1.1.1.1.1.13 Google Smart Tap Collector ID Slot 4 220
8.2.1.18 Property 1.1.1.1.1.14 Google Smart Tap Collector ID Slot 5 221
8.2.1.19 Property 1.1.1.1.1.15 Google Smart Tap Collector ID Slot 6 222
8.2.1.20 Property 1.1.1.1.1.1A Google Smart Tap POS Capabilities 222
8.2.1.21 Property 1.1.1.1.2.6 Tip Mode Enable Submit on Amount Button Press 224
8.2.1.22 Property 1.1.1.1.4.1 EMV Configuration Filename 225
8.2.1.23 Property 1.1.1.1.4.2 CA Public Key Configuration Filename 226
8.2.2 Property Subgroup 1.1.2.nnn SRED Settings 227
8.2.2.1 Property 1.1.2.2.1.2 ISO/ABA Masking Character 227
8.2.2.2 Property 1.1.2.2.1.3 PAN Number of Leading Unmasked Characters 229
8.2.2.3 Property 1.1.2.2.1.4 PAN Number of Trailing Unmasked Characters 230
8.2.2.4 Property 1.1.2.2.1.5 PAN MOD 10 Check Digit Correction 231
8.2.2.5 Property 1.1.2.4.1.2 Key Mapping of Account Data 232
8.2.2.6 Property 1.1.2.4.1.3 Key Mapping of MAC 232
8.2.2.7 Property 1.1.2.4.1.5 Key Mapping of MagTek Token 234
8.2.2.8 Property 1.1.2.4.1.6 Key Mapping of User Data 1 234
8.3 Property Group 1.2.nnnn Device Settings 234
8.3.1 Property Subgroup 1.2.1.nnn Transaction Settings 234
8.3.1.1 Property 1.2.1.1.1.2 Application Selection Behavior (Contactless Only) 234
8.3.1.2 Property 1.2.1.1.2.1 Signature Capture Control 235
8.3.1.3 Property 1.2.1.1.3.1 Contactless Low Power Card Detect (Contactless Only) 236
8.3.1.4 Property 1.2.1.1.4.1 MIFARE Ultralight C 2keys3DES 237
8.3.1.5 Property 1.2.1.1.4.2 MIFARE Ultralight AES DataProtKey. 238
8.3.1.6 Property 1.2.1.1.4.3 MIFARE Ultralight AES UIDRetrKey. 239
8.3.1.7 Property 1.2.1.1.4.4 MIFARE Ultralight AES OriginalityKey. 240
8.3.1.8 Property 1.2.1.1.4.5 MIFARE Plus AES_Key1. 241
8.3.1.9 Property 1.2.1.1.4.6 MIFARE Plus AES_Key2. 242
8.3.1.10 Property 1.2.1.1.4.7 MIFARE Plus AES_Key3. 243
8.3.1.11 Property 1.2.1.1.4.8 MIFARE Plus AES_Key4. 244
8.3.1.12 Property 1.2.1.1.4.9 MIFARE Plus AES_Key5. 245
8.3.1.13 Property 1.2.1.1.4.A MIFARE Plus AES_Key6. 246

8.3.2 Property Subgroup 1.2.2.2.nn USB Settings (USB Only) 247
8.3.2.1 Property 1.2.2.2.1.1 Reduce Power During USB Suspend 247
8.3.2.2 Property 1.2.2.2.1.2 USB Configuration Type 248
8.3.3 Property Subgroup 1.2.3.nnn User Interface Settings 249
8.3.3.1 Property 1.2.3.1.3.1 Enable Card Logos Page at Startup 249
8.3.3.2 Property 1.2.3.1.3.2 Display Card Logo While Authorizing 250
8.3.3.3 Property 1.2.3.2.1.1 System Volume Control 251
8.3.4 Property Subgroup 1.2.5.nnn Security Settings 252
8.3.4.1 Property 1.2.5.2.1.1 Device Lock State 252
8.3.4.2 Property 1.2.5.2.1.2 Device Lock State After Reset 253
8.3.4.3 Property 1.2.5.2.1.3 Device Lock Passcode 254
8.3.5 Property Subgroup 1.2.7.nnn System Settings 255
8.3.5.1 Property 1.2.7.1.1.1 Device Reset Occurred Notification Control 255
8.3.5.2 Property 1.2.7.1.1.2 Device Reset Occurred Notification Acknowledged 256
8.3.5.3 Property 1.2.7.1.1.3 Device Reset Will Occur Soon Notification Control 257
8.3.5.4 Property 1.2.7.1.1.4 Auto Reset Configuration 258
8.3.5.5 Property 1.2.7.1.2.1 User Event Notification Controls Enable 259
8.3.5.6 Property 1.2.7.1.3.1 Maximum Battery Charge Level (Deprecated) 261
8.3.5.7 Property 1.2.7.1.4.1 Device Low Temperature Notification Level 262
8.3.5.8 Property 1.2.7.1.4.2 Device High Temperature Notification Level 263
8.3.5.9 Property 1.2.7.1.4.3 Device Temperature Notification Repeat Interval 264
8.4 Property Group 2.1.2.nnn Device Firmware Identification Information 265
8.4.1 Property Subgroup 2.1.2.1.nn Boot Firmware Information 265
8.4.1.1 Property 2.1.2.1.1.2 Boot1 Firmware Version 265
8.4.1.2 Property 2.1.2.1.1.4 Boot1 Firmware Part Number 265
8.4.1.3 Property 2.1.2.1.2.2 Boot0 Firmware Version 266
8.4.1.4 Property 2.1.2.1.2.4 Boot0 Firmware Part Number 267
8.4.2 Property Subgroup 2.1.2.2.nn Core Firmware Information 268
8.4.2.1 Property 2.1.2.2.2.1 Device Model Name 268
8.4.2.2 Property 2.1.2.2.2.2 Main Firmware Version 268
8.4.2.3 Property 2.1.2.2.2.4 Main Firmware Part Number 269
8.4.2.4 Property 2.1.2.2.2.6 Key Type 270
8.4.3 Property Subgroup 2.1.2.3.nn EMV Firmware Information 270
8.4.3.1 Property 2.1.2.3.2.1 EMV Contact L1 Kernel ID (Contact Only) 270
8.4.3.2 Property 2.1.2.3.2.2 EMV Contact L1 Kernel Firmware Part Number (Contact Only) 271
8.4.3.3 Property 2.1.2.3.3.1 EMV Contact L2 Kernel ID (Contact Only) 272
8.4.3.4 Property 2.1.2.3.3.2 EMV Contact L2 Kernel Firmware Part Number(Contact Only) 273

8.4.3.5 Property 2.1.2.3.3.3 EMV Contact L2 Kernel Checksum (Contact Only) 273
8.4.3.6 Property 2.1.2.3.3.4 EMV Contact L2 Kernel Configuration Checksum (Contact Only) 274
8.4.3.7 Property 2.1.2.3.4.1 EMV Contactless L1 Kernel ID (Contactless Only) 275
8.4.3.8 Property 2.1.2.3.4.2 EMV Contactless L1 Kernel Firmware Part Number (Contactless Only).....275
8.4.3.9 Property 2.1.2.3.4.3 EMV Contactless L1 Kernel Checksum (Contactless Only) 276
8.4.3.10 Property 2.1.2.3.5.1 Mastercard MCL Kernel ID (Contactless Only) 277
8.4.3.11 Property 2.1.2.3.5.2 Mastercard MCL Kernel Firmware Part Number (Contactless Only) 278
8.4.3.12 Property 2.1.2.3.5.3 Mastercard MCL Kernel Checksum (Contactless Only) 278
8.4.3.13 Property 2.1.2.3.6.1 Visa payWave Kernel ID (Contactless Only).....279
8.4.3.14 Property 2.1.2.3.6.2 Visa payWave Kernel Firmware Part Number (Contactless Only) 280
8.4.3.15 Property 2.1.2.3.6.3 Visa payWave Kernel Checksum (Contactless Only).....280
8.4.3.16 Property 2.1.2.3.6.5 Entry Point Checksum (Contactless Only)(Common Kernel Only) 281
8.4.3.17 Property 2.1.2.3.7.1 Discover D-PAS Kernel ID (Contactless Only) 282
8.4.3.18 Property 2.1.2.3.7.2 Discover D-PAS Kernel Firmware Part Number (Contactless Only) 283
8.4.3.19 Property 2.1.2.3.7.3 Discover D-PAS Kernel Checksum (Contactless Only) 283
8.4.3.20 Property 2.1.2.3.8.1 American Express Expresspay Kernel ID (Contactless Only) 284
8.4.3.21 Property 2.1.2.3.8.2 American Express Expresspay Kernel Firmware Part Number (Contactless Only) 285
8.4.3.22 Property 2.1.2.3.8.3 American Express Expresspay Kernel Checksum (Contactless Only).....285
8.4.3.23 Property 2.1.2.3.9.1 Apple VAS Kernel ID 286
8.4.3.24 Property 2.1.2.3.A.1 JCB Kernel ID (Contactless Only) (Common Kernel Only).287
8.4.3.25 Property 2.1.2.3.A.2 JCB Kernel Firmware Part Number (Contactless Only) (Common Kernel Only).....288
8.4.3.26 Property 2.1.2.3.A.3 JCB Kernel Checksum (Contactless Only) (Common Kernel Only) 288
8.4.3.27 Property 2.1.2.3.A.4 Reader Core Checksum (Contactless Only) (Common Kernel Only) 289
8.4.3.28 Property 2.1.2.3.A.5 Entry Point Checksum (Contactless Only) (Common Kernel Only) 290
8.4.3.29 Property 2.1.2.3.B.1 China Union Pay Kernel ID (Contactless Only) (Common Kernel Only).....290
8.4.3.30 Property 2.1.2.3.B.2 China Union Pay Kernel Firmware Part Number (Contactless Only) (Common Kernel Only) 291

8.4.3.31 Property 2.1.2.3.B.3 China Union Pay Kernel Checksum (Contactless Only) (Common Kernel Only) 292
8.4.3.32 Property 2.1.2.3.C.1 Interact Flash Kernel ID (Contactless Only) (Common Kernel Only) 293
8.4.3.33 Property 2.1.2.3.C.2 Interact Flash Kernel Firmware Part Number (Contactless Only) (Common Kernel Only) 293
8.4.3.34 Property 2.1.2.3.C.3 Interact Flash Kernel Checksum (Contactless Only) (Common Kernel Only) 294
8.4.4 Property Subgroup 2.1.2.8.nn Custom UI 295
8.4.4.1 Property 2.1.2.8.1.1 UI Configuration File 295
8.5 Property Group 2.2.1.nnn Device Hardware Information 296
8.5.1 Property Subgroup 2.2.1.1.1.n Device Common Configuration Information 296
8.5.1.1 Property 2.2.1.1.1.1 Serial Number 296
8.5.1.2 Property 2.2.1.1.1.3 PCI Hardware ID 296
8.6 Property Group 2.3.1.nnn System Status Information 297
8.6.1 Property Subgroup 2.3.1.1.1.n System Status Injected Key Information 297
8.6.1.1 Property 2.3.1.1.1.1 Device Key Status 297
8.6.1.2 Property 2.3.1.1.1.2 Transaction Key Status 298
8.6.2 Property Subgroup 2.3.1.1.2.n System Status Security Protection Information 299
8.6.2.1 Property 2.3.1.1.2.1 Real Time Clock Enabled 299
8.6.2.2 Property 2.3.1.1.2.2 Tamper Sensors Activated 300
8.6.2.3 Property 2.3.1.1.2.3 Tamper Sensor Tampered 301
8.6.3 Property Subgroup 2.3.1.2.1.n System Status Device State Information 301
8.6.3.1 Property 2.3.1.2.1.1 Device Operational Status 301
8.6.3.2 Property 2.3.1.2.1.2 Offline Status Detail 302
8.6.3.3 Property 2.3.1.2.1.3 External Power Supplied 303
8.6.3.4 Property 2.3.1.2.1.4 Battery State of Charge 304
8.6.3.5 Property 2.3.1.2.1.5 Battery Charger Status 305
8.6.3.6 Property 2.3.1.2.1.6 Device Temperature 305
Appendix A Object IDs (OIDs) and ITU-T X660 307
A.1 About OIDs 307
A.2 Octet Encoding of OIDs 308
A.3 TLV Encoding of OIDs 309
Appendix B Barcode Reader Symbologies 310
Appendix C Erasing EMV Configurations 311
C.1 Load CAPK with AID: 0000000000 to erase all current CAPK keys with this CAPK key: 311
C.2 Empty Amex DRL: Load AMEX DRL file with “EMPTY” content 312
C.3 Empty Terminal: Load Terminal file with “EMPTY” content 313
C.4 Empty Processing: Load Processing file with “EMPTY” content and 1 delimiter (FF33) ..314

C.5 Empty Entry Point: Load Entry file with “EMPTY” content and 1 delimiter (FF35).....315

1 Introduction

1.1 About This Document

This document describes how to communicate with Secure Card Reader Authenticator (SCRA) devices which implement MagTek Messaging Schema (MMS) DynaProx system architecture.

The document uses **bold face** to:

- Highlight terms / concepts being formally defined in the current sentence / paragraph
- Highlight important distinguishing keywords in sentences
- Indicate hyperlinks to other sections / tables

The document uses a small number of annotation standards that are important to understand:

- Hexadecimal values are prefixed with **0x** unless the context clearly indicates an un-prefixed number is hexadecimal (for example, TLV tags, lengths, and values are always assumed to be hex).
- Binary values are prefixed with **0b** unless the context clearly indicates the value is binary.
- Decimal values are not prefixed unless required for clarity, in which case the prefix is **0d**.

The standard documented by this document makes extensive use of Tag-Length-Value encoding. Section **3.2.1 Tag-Length-Value (TLV) Encoding** describes how to encode and decode TLV, and how to read the tables in this document that describe TLV data objects.

1.2 About This Document

This document describes how to communicate with Secure Card Reader Authenticator (SCRA) devices which implement MagTek Messaging Schema (MMS) and the DynaFlex II Go system architecture.

The document uses **bold face** to:

- Highlight terms / concepts being formally defined in the current sentence / paragraph
- Highlight important distinguishing keywords in sentences
- Indicate hyperlinks to other sections / tables

The document uses a small number of annotation standards that are important to understand:

- Hexadecimal values are prefixed with **0x** unless the context clearly indicates an un-prefixed number is hexadecimal (for example, TLV tags, lengths, and values are always assumed to be hex).
- Binary values are prefixed with **0b** unless the context clearly indicates the value is binary.
- Decimal values are not prefixed unless required for clarity, in which case the prefix is **0d**.

The standard documented by this document makes extensive use of Tag-Length-Value encoding. Section **3.2.1 Tag-Length-Value (TLV) Encoding** describes how to encode and decode TLV, and how to read the tables in this document that describe TLV data objects.

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 instance, the term "host" can signify different things depending on the context—such as USB communication versus networked financial transaction processing. In this document, "device" and "host" are defined specifically as follows:

- **Device** refers to the Secure Card Reader Authenticator (SCRA) or PIN Entry Device (PED) that receives and responds to the command set specified in this document. Refer to **Table 1.6-1 - Device Features**, to determine a device’s specific capabilities, not all devices support PIN entry. Devices include DynaFlex, DynaProx, DynaFlex II, 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 **host software** to refer to software on the host that provides a user interface for the operator.

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

1.4 About SDKs and Sample Code

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:

- ***D998200380 MagTek Universal SDK Programmer’s Manual (Microsoft .NET)***
- ***D998200381 MagTek Universal SDK Programmer’s Manual (Microsoft C++)***
- ***D998200385 MagTek Universal SDK Programmer’s Manual (Java)***
- ***D998200386 MagTek Universal SDK Programmer’s Manual (iOS)***
- ***D998200387 MagTek Universal SDK Programmer’s Manual (Android)***
- ***D998200388 MagTek Universal SDK Programmer’s Manual (macOS)***

The documentation is bundled with the SDKs themselves, which include:

- ***1000007351 MagTek Universal SDK for MMS Devices (Windows)***

- ***1000007352 MagTek Universal SDK for MMS Devices (Android)***
- ***1000007353 MagTek Universal SDK for MMS Devices (iOS)***
- ***1000007354 MagTek Universal SDK for MMS Devices (macOS)***

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 sample software that demonstrates direct communication with the device, which software developers can use to test the device and 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.

1.5 About Connections and Data Formats

MMS 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 vendor-defined HID device ("USB HID"), wireless LAN (WLAN), Bluetooth®, Bluetooth® Low Energy ("Bluetooth® LE"), RS-232, Apple Lightning, and so on. The set of available physical connection types and the data formats available on each connection type is device dependent. **Table 1.5-1** 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**. Section headings in this document include tags that indicate which connection types and/or data formats they apply to.

Table 1.5-1 - Device Connection Types / Data Formats

Product / Connection	Bluetooth®® LE GATT	RS-232 / UART	USB HID	WLAN	iAP2	Ethernet
DynaFlex with USB Only			HID			
DynaFlex Pro with USB Only			HID			
DynaFlex Pro w/WLAN			HID	WLAN		
DynaProx		SLIP	HID		iAP2-USB	
DynaFlex II with USB Only			HID			
DynaFlex II PED with USB Only			HID			
DynaFlex II PED w/WLAN			HID	WLAN		
DynaFlex II Go with USB Only			HID		iAP2-USB	
DynaFlex II Go w/Bluetooth® LE	GATT		HID		iAP2-USB	

1.6 About Device Features

Much of the information in this document is applicable 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 1.6-1** 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 1.6-1 - Device Features

Feature / Product	DynaFlex II GO	DynaFlex with USB Only	DynaFlex w/Bluetooth® LE	DynaFlex Pro with USB Only	DynaFlex Pro with Bluetooth® LE	DynaFlex Pro with WLAN	DynaFlex Pro with Ethernet	DynaProx	DynaFlex II with USB Only	DynaFlex II w/Bluetooth® LE	DynaFlex II PED with USB Only	DynaFlex II PED with Bluetooth® LE	DynaFlex II PED with WLAN	DynaFlex II PED with Ethernet
MSR	Y	Y	Y	Y	Y	Y	Y	N	Y	Y	Y	Y	Y	Y
EMV Contact	Y	Y	Y	Y	Y	Y	Y	N	Y	Y	Y	Y	Y	Y
EMV Contactless	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
MCE (Manual Card Entry)	N	N	N	Y	Y	Y	Y	N	N	N	Y	Y	Y	Y
BCR (Barcode Reader)	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
LED RGBx4	1	Y	Y	Y	Y	Y	Y	N	Y	Y	Y	Y	Y	Y
LED Monox4	Y	N	N	N	N	N	N	Y	N	N	N	N	N	N
Touch	N	N	N	Y	Y	Y	Y	N	N	N	Y	Y	Y	Y
No Touch	Y	Y	Y	N	N	N	N	Y	Y	Y	N	N	N	N
Display	N	N	N	Y	Y	Y	Y	N	N	N	Y	Y	Y	Y
No Display	Y	Y	Y	N	N	N	N	Y	Y	Y	N	N	N	N
Battery Power	Y	N	Y	N	Y	Y	Y	N	N	Y	N	Y	Y	Y
Banking Functions	N	N	N	Y	Y	Y	Y	N	N	N	Y	Y	Y	Y

Feature / Product	DynaFlex II GO	DynaFlex with USB Only	DynaFlex w/Bluetooth® LE	DynaFlex Pro with USB Only	DynaFlex Pro with Bluetooth® LE	DynaFlex Pro with WLAN	DynaFlex Pro with Ethernet	DynaProx	DynaFlex II with USB Only	DynaFlex II w/Bluetooth® LE	DynaFlex II PED with USB Only	DynaFlex II PED with Bluetooth® LE	DynaFlex II PED with WLAN	DynaFlex II PED with Ethernet
Session Management	N	N	N	N	N	Y	N	N	N	N	N	N	Y	N
Apple VAS	Y	N	N	N	N	N	N	Y	Y	Y	Y	Y	Y	Y
Google Wallet Smart Tap	Y	N	N	N	N	N	N	Y	Y	Y	Y	Y	Y	Y
Flexible UI	N	N	N	N	N	N	N	N	N	N	Y	Y	Y	Y
Common Kernel	Y	N	N	N	N	N	N	Y	Y	Y	Y	Y	Y	Y
Card Emulation	Y	N	N	N	N	N	N	Y	Y	Y	Y	Y	Y	Y

2 Connection Types

Table 1.5-1 on page **20** 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 USB Connections (USB Only)

These USB devices conform to the USB specification revision 1.1. They also conform to the Human Interface Device (HID) class specification version 1.1. This document assumes the audience is familiar with USB HID class specifications, which are available at ***www.usb.org***. MagTek strongly recommends becoming familiar with that standard before trying to communicate with the device directly via USB.

These devices are full-speed, high-powered USB devices that draw power from the USB bus they are connected to. They enter and wake up from Suspend mode when directed to do so by the USB host. They do not support remote wakeup.

When connecting via USB, MMS devices connect to the USB host as a **vendor-defined** Human Interface Device. MMS devices identify themselves to the host with MagTek's vendor ID **0x0801**. DynaFlex products report as Product ID (PID) **0x2020**, DynaProx products report as Product ID (PID) **0x2023 and DynaFlex II Go report as Product ID (PID) 0x2024**.

Details for exchanging messages with the device are provided in the sections that follow.

2.1.1 About USB Reports, Usages, Usage Pages, and Usage IDs

All USB HID devices send and receive data using **Reports**. Each report may contain several sections, called **Usages**, each of which has its own unique four-byte (32-bit) identifier. The two most significant bytes of a usage are called the **usage page**, and the two least significant bytes are called the **usage ID**. Vendor-defined HID device usages must have a usage page in the range **0xFF00 - 0xFFFF**, and it is common practice for related usage IDs share the same usage page. For these reasons, all usages for MMS devices use vendor-defined usage page **0xFF00**, **Magnetic Stripe Reader**.

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

- **Output Reports**, which the host uses to send messages (such as commands) to the device.
- **Input Reports**, which the device uses to send messages (such as responses and unsolicited notifications) to the host.

For information about using output reports to send command messages to / receive response messages from the device, see section **2.1.3 How to Send Command Requests Using the USB Connection**. For information about receiving unsolicited data from the device, see section **2.1.4 How to Receive Data Using the USB Connection (HID Only)**.

2.1.2 How to Connect to a Device Using the USB Connection

The steps for a host to connect to the device using the USB connector are the same as connecting to any USB HID device, and are platform specific. The general steps regardless of platform are as follows:

- Enumerate the USB devices connected to the host.
- Locate the desired device by PID.
- Open the USB connection.
- Read the report descriptor.

- Begin listening for incoming reports, and sending data using outgoing reports.

2.1.3 How to Send Command Requests Using the USB Connection

MMS communication between the host and the device consists of **Messages**, which are independent of connection type and are documented in section **2.2**. This section focuses specifically on how to use the device's USB HID connection to transmit a **Command** to the host by composing and sending a **Request Message**, then listening for and interpreting the corresponding **Response Message**.

When the device is connected to the host via USB, the host sends one or more **Output Reports** to the device to send the requests for commands, and listens / waits for the device to send one or more **Input Reports** containing a response (see section **2.1.1**). All reports use Usage Page **0xFF00**, Usage ID **0x20**, and no Report ID (which some platform libraries present to the calling software as Report ID 0x00).

The host may send Output Reports by using either the default **Control** pipe or the Interrupt OUT pipe using a blocking call to the platform's USB libraries. The device ACKs all Output Reports immediately. Upon detecting the ACK for the last Output Report in the message stream, the host software can immediately begin listening for one or more follow-up Input Reports containing the device's response.

Some messages defined in section **2.2** may exceed the maximum packet length allowed by USB HID. For this reason, MMS implements a packetizing scheme the host must use when it composes Output Reports to send commands, and when it decomposes Input Reports to receive responses. The host should follow this general sequence to send a request and receive a response:

1) Choose the command to invoke from section **6 Commands**.
2) Construct the command request message using the **Request** table in the command documentation. For a deeper explanation of the contents of request tables, see section **2.2**.
3) .
4) Determine the complete length of the command request message.
5) Examine the device's **Report Descriptor** to determine what payload length the device expects for an Output Report (operating system libraries may refer to this length as the "Report Length" or "Report Count"). MMS devices using USB HID generally use a payload length of 64 bytes.
   a) If the message length fits the **Single Packet** payload length of 62 bytes, wrap the message in the Single Packet format shown in **Table 2.1-1**.
   b) If the message length does not fit the Single Packet payload length, wrap the message in the format of a single **Multi-Packet Head**, followed by zero or more **Multi-Packet Middles** as necessary, followed by one **Multi-Packet Tail**, as shown in **Table 2.1-2**, **Table 2.1-3**, and **Table 2.1-4**. If necessary, the host can cancel in the middle of a Multi-Packet message using the **Multi-Packet Cancel** packet shown in **Table 2.1-5**.
6) For each packet, send an Output Report to the device, and wait for the device to ACK the output report USB packet. Upon ACK, either send the next packet in the sequence (if any), or begin listening for an Input Report containing the device's response message [see section **2.1.4 How to Receive Data Using the USB Connection (HID Only)**]. If an ACK does not arrive, time out and assume the command needs to be started again.
7) Strip wrappers off the device's response in exactly the same way you constructed the command (using either Single Packet format or Multi-Packet format), and recompose the response message. Be sure to assemble the packets in the correct order using **Packet Number**, and to truncate the padding based on **Message Length** or **Remaining Message Length**.
8) Parse the final response message using the Response table in the documentation for the command.

Table 2.1-1 - Single Packet Format

Byte Offset	Field Name	Field Value
0	Packet Type	0x00 = Single Packet
1	Message Length	Message length, denoted as **N**
2..N+1	Message	Message exactly from section **2.2**
N+2..63	Padding, if needed	0x00

Table 2.1-2 - Multi-Packet Head Format

Byte Offset	Field Name	Field Value
0	Packet Type	0x01 = Multi-packet Head
1..4	Message Length	Total message length before decomposing into packets, denoted as **N**. If 62 or less, use Single Packet Format.
5..63	Message Part	First 59 bytes of message from section **2.2**

Table 2.1-3 - Multi-Packet Middle Format

Byte Offset	Field Name	Field Value
0	Packet Type	0x02 = Multi-Packet Middle
1..2	Packet Number	0x0001 = First Multi-Packet Middle 0x0002 = Second Multi-Packet Middle Etc. up to 0xFFFF
3..63	Message Part	Next 61 bytes of message from section **2.2**

Table 2.1-4 - Multi-Packet Tail Format

Byte Offset	Field Name	Field Value
0	Packet Type	0x03 = Multi-Packet Tail
1	Remaining Message Length	Number of bytes of the message that are being transmitted in this final packet, denoted as **n**.
2..n+1	Message Part	Final n bytes of message exactly as described in section **2.2**
n+2..63	Padding, if needed	0x00

Table 2.1-5 - Multi-Packet Cancel Format

Byte Offset	Field Name	Field Value
0	Packet Type	0x04 = Multi-Packet Cancel
1..2	Reason for Cancel	0x0000 = General
3..63	Padding	0x00

2.1.4 How to Receive Data Using the USB Connection (HID Only)

MMS devices use the same mechanism to send **Response** messages to host commands and to send unsolicited **Notifications** to the host when events occur, such as device state changes or user interactions.

When the device communicates with the host as a vendor-defined HID device, it sends messages to the host via an Input Report, which it sends to the host using the **USB Interrupt IN** pipe. Per the USB HID standard, the host polls the device on a regular Polling Interval to see if the device has input reports available to send. If the device does not, it responds to the host's poll with a USB `NAK`.

The host software should always be listening for **Input Reports** from the device containing notification messages and command response messages. Assembling and parsing the incoming message is the same in both cases, but notification messages, which are generally unpredictable, need to be routed to different handlers than responses, which are generally anticipated after sending a command. All input reports use Usage Page **0xFF00**, Usage ID **0x20**. Because MMS devices only implement one input report, the input report they send to the host does not include an Input Report ID (which some operating system libraries present to the calling software as Input Report ID 0x00), in accordance with the USB HID specification.

Some response and notification messages defined in section **2.2** may exceed the maximum packet length allowed by USB HID. If the message can't fit into one packet, the device sends multiple packets, each containing partial message data using the same message packet structures described in section **2.1.3**, **Table 2.1-1**, **Table 2.1-2**, **Table 2.1-3**, **Table 2.1-4**, **Table 2.1-5**.

Upon receiving an input report, the host can determine the size of message packet Input Reports by looking at the HID report descriptor. The host can locate a specific data element in the report by finding the corresponding Usage and interpreting its contents as binary data. Because MMS devices use a single usage to hold the Input Report contents, unlike devices that divide the Input Report contents into length-delimited Usages, the host software can hard-code its method of finding the packet data; it is not necessary for the host software to examine the Report Descriptor for information about separate data elements' offsets upon receiving a packet.

The host should follow this general sequence to process incoming Input Reports:

1) Register a listener with the operating system USB library to have the relevant Input Reports routed to the host software for processing.
2) When an Input Report arrives, knowing from section **2.1.1** that the device uses **Usage Page 0xFF00**, and knowing the desired data is found in the usage with **Usage ID 0x0020**, call the platform's USB SDK to retrieve the data from **Usage 0xFF000020** in the input report.
3) Interpret the blob of data according to tables in section **2.1.3** to determine whether the incoming message is in single packet format or multi-packet format. If it is using multi-packet format:
   a) Continue buffering the message and processing incoming packets until the last packet of the message arrives.
   b) Assemble the full message. Be sure to assemble the packets in the correct order using the Packet Number, and to truncate the padding based on **Message Length** or **Remaining Message Length**.
4) Determine the message type and message ID and parse it according to the corresponding Response table or Notification table in section **2.2**.
5) Route the fully composed message to the appropriate handler.

2.2 How to Use Apple iAP2 Connections (iAP2 Only)

This section provides information about developing an iOS app that interfaces with the device via the USB connector using iPod Accessory Protocol (iAP2). For sample code and other supporting materials, see ***1000007353 MAGTEK UNIVERSAL SDK FOR MMS DEVICES (iOS)***, 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:

- Apple iAP2 AppBundleID, 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 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.

3 Messages (Requests, Responses, Notifications, and Files)

This section describes the general format of messages exchanged between hosts and devices that are using the common MMS message framework.

Documentation about the specific messages supported by a given device is provided in section **6 Commands**.

3.1 About Messages

The host and the device communicate with each other by exchanging blocks of data called **Messages**, which are standardized wrappers containing a **payload** that is either a command **Request**, a command **Response**, an unsolicited **Notification**, or a **File**. For example, the host may send a command **request** message to the device to change a configuration setting, and the device may send a command **response** message to indicate the command was successful; when a cardholder inserts a card, the device may send a **notification** message to the host that a cardholder has initiated a transaction; the host may send the device a **file** message to load firmware.

Messages can be nested. For example, a top-level secure wrapper request from the host to the device may contain an encrypted or signed command request for the device to unpack, validate, and execute.

Requests and Responses. Requests and responses are two of the message payload types the host and device exchange inside messages. The combination of a message that contains a **request** payload and a message that contains the corresponding **response** payload is referred to generally in this document as a **Command**. The device can only service one command request at a time, and sends each command response within a pre-determined finite amount of time after receiving the request. After sending a command request, the host must wait until the device returns a response before sending another request, or until the request is unanswered after a reasonable host-defined timeout period passes.

Notifications. Notifications are a message payload type the host and device exchange inside messages. The device sends notification messages to the host if the device's state changes or if an external event occurs, such as a cardholder inserting a card. The device can send a notification at any time, and does not expect a response or any specific action from the host. By default, the device sends all notifications to the USB interface. To configure the device to send notifications on additional connections, use **Command 0x1F02 - Set Notification Subscriptions**.

Data Files. Data Files are a message payload type the host and device exchange inside messages. The device handles them as a stream, in that it begins storing the payload of the message before it has received the final packet of the message, allowing for much larger payloads than standard requests. This is made possible by restricting the function of the message to just transferring a file, which means restricting the message payload to primitive data only; it cannot contain composed TLV data objects.

Regardless of connection type, all MMS devices use the same schema for sending and receiving messages, which is documented in section **3.2 Message Format**. For information about transmitting and receiving messages using specific connection types, which involves following connection-specific rules for breaking messages down into transmittable **Message Streams**, see section **2 Connection Types**.

3.2 Message Format

3.2.1 Tag-Length-Value (TLV) Encoding

3.2.1.1 About TLV Encoding

All messages exchanged between the host and the device are formatted using the tag-length-value **Distinguished Encoding Rules** (DER) defined in ***ITU-T X.680|ISO/IEC 8824-1*** and ***ITU-T X.690|ISO/IEC 8825-1***. A subset of these standards is also used in ***EMV Integrated Circuit Card Specifications for Payment Systems 4.3, Part IV, Annex B Rules for BER-TLV Data Objects***, so the latter can serve as a useful point of reference, especially for developers who are familiar with systems that exchange blocks of EMV data.

Summarizing those specifications, each TLV data object follows these basic rules:

- The DER standard designates the least significant bit of a byte as **bit 1**, and the most significant bit of a byte as **bit 8**. This is different from the remainder of the MMS standard, which indexes bit numbers starting at 0 to be consistent with each bit position number representing that bit's power of 2.
- The **Tag** or **Identifier** portion of a TLV data object identifies the TLV data object. This standard assigns the **tag** (identifier) portion of TLV data objects according to the DER format as follows:
    - Bits 8 and 7 specify whether the TLV data object is universal, application-defined, context-specific, or private. Most messages in this standard contain **context-specific** tags (bits 8 and 7 = **10**), meaning different messages reuse the same tags, and the tags represent sequentially numbered parameters passed in any message. The contents of each context-specific tag (parameter) are described for each message in section **6 Commands**.
    - Bit 6 specifies whether the tag is **primitive** (bit 6=**0**), meaning it contains its values directly, or **constructed** (bit 6=**1**), meaning the TLV data object contains more TLV data objects.
    - Bits 5 to 1 specify a unique tag number, with 11111 reserved to mean:
        - The tag is not a single byte long
        - Bits 7 to 1 of subsequent bytes with bit 8 set to 1 are also part of the tag identifier with the most significant of the whole tag number in bit 7
        - Bits 7 to 1 of the final byte with bit 8 set to 0 is also part of the tag identifier
- The **Length** portion is calculated as the total length of the Data portion that follows it: Lengths can be either short form or long form:
    - **Short form** means the length is one byte long in the range `0x00` to `0x7F`.
    - **Long form** means the length is multiple bytes long, starting with one-byte `0x80` or greater, where the lower 7 bits specify how many subsequent bytes are used to indicate the length. For example, in the case of length `8201C3`, `0x82` is greater than `0x7F`, and the lower 7 bits equal `0x02`, so the next 2 bytes `0x01C3` give the total length of the data block, 451 bytes.
    - DER format stipulates all TLV data objects should be encoded using the smallest length required to fit the data.
- The **Value** or **Data** portion is the actual payload of the TLV data object.

This document provides message definitions in hexadecimal format; when the host constructs or interprets a message, if no additional encode / decode filtering or translation is in place at the platform layer, it should expect each hexadecimal value shown in this document to be represented as binary bytes in the message stream, not as string literals. For example, FF is a single byte with all bits set to 1, not the two-byte string literal "FF."

3.2.1.2 TLV Example

Below is a brief example of a TLV-encoded request and response for **Command 0xDF01 - Echo**, wrapped in the standard message format documented in section **3.2.2 Message Structure**.

The host sends the device a binary byte stream `AA0081040101DF018407DF018103010203`, which breaks down as follows:

- AA00 = Standard **Start of Message** / **API Framework Version** (not TLV)
- 81, 04, 0101DF01
  - Tag 81 = Request Message Parameter 1, **Message Information**
  - Length 04
  - Value 0101DF01
    - 01 = Request from host to device
    - 01 = Message reference number
    - DF01 = **Command 0xDF01 - Echo**
- 84, 07, DF018103010203
  - Tag 84 = Request Message Parameter 4, **Request Payload**
  - Length 07
  - Value DF018103010203
    - DF01 = Payload format is for Request **Command 0xDF01 - Echo** (not TLV)
    - Tag 81 = Payload Parameter 1, **Value to Echo**
    - Length 03
    - Value 010203 = Value to Echo 010203

In response, the device sends the host a binary byte stream `AA0081048201DF01820400000000008407DF018103010203`, which breaks down as follows:

- AA00 = Standard **Start of Message** / **API Framework Version** (not TLV)
- 81, 04, 8201DF01
  - Tag 81 = Response Message Parameter 1, **Message Information**
  - Length 04
  - Value 8201DF01
    - 82 = Response from device to host
    - 01 = Message reference number
    - DF01 = **Command 0xDF01 - Echo**
- 82, 04, 00000000
  - Tag 82 = Response Message Parameter 2, **Response Status** (one byte **Operation Status Summary**, three bytes **Operation Status Detail**)
  - Length 04
  - Value 00, 000000 = OK / Done, General / All Good / Requested operation was successful
- 84, 07, DF018103010203
  - Tag 84 = Response Message Parameter 4 for **Response Payload**
  - Length 07
  - Value DF018103010203

- DF01 = Payload format is for Response **Command 0xDF01 - Echo** (not TLV)
- Tag 81 = Payload Parameter 1, **Value to Echo**
- Length 03
- Value 010203 = Value to Echo 010203

3.2.1.3 How to Read TLV Tables

Throughout this document, tables that show TLV data objects use some number of slashes in front of the **Tag** identifier to indicate that object's *relative* level of nesting / containment within other TLV data objects *in the same table*. These levels should not be assumed to be absolute levels, because a given TLV data object may be nested within other TLV data objects at any level. The slash notation saves table space compared to whitespace indenting and is easier to read than a separate "Nesting Level" column, which can be difficult to notice.

For example:

- **Earth** contains
- **/North America**, which contains
- **//United States**, which contains
- **///California**.

Although **Earth** is a *relative* root within the context of the objects above, it is not an *absolute* root, because the nested objects starting with the **Earth** relative root could be wrapped inside another object representing **Sol Solar System**, and that could be wrapped in **Milky Way Galaxy**, and so on.

In tables that show TLV data objects, a Length of **var** means the length is variable, and the device or host must calculate it based on whatever is nested inside the TLV data object.
See **Table 3.2-1 below** for an example.

Table 3.2-1 - TLV Table Example

Tag	Len	Value / Description	Typ	Req	Default
A1	var	TLV data object **A1** contains four directly nested TLV data objects: 81, 82, A3, and 84 for short, or more precisely A1/81, A1/82, A1/A3, and A1/84. A1/82 is optional (Req = **O**), so the length of A1 will vary depending on whether or not A1/82 is included (Len = **var**) or excluded / left implicit.	T	R	
/81	01	TLV data object A1/81 contains one byte and is required. It has no default value because it must be explicitly included.	B	R	
/82	03	TLV data object A1/82 contains three bytes but is optional. If it is not included in this object, the device (or host) assumes the default value 0x4D6F6D.	B	O	0x4D6F6D
/A3	08	TLV data object A1/A3 contains two TLV data objects: 81 and 82, or more precisely, A1/A3/81 and A1/A3/82. Its length is the combined length of its two nested objects: 5 bytes for 81 (tag **81**, length **03**, value **xx xx xx**) and 3 bytes for 82 (tag **82**, length **01**, value **xx**).	T	R	
//81	03	TLV data object A1/A3/81 contains three bytes and is required.	B	R	
//82	01	TLV data object A1/A3/82 contains one byte and is required.	B	R	

Tag	Len	Value / Description	Typ	Req	Default
/84	03	TLV data object A1/84 contains three bytes that represent distinct values, but to save space, they are length-delimited directly inside A1/84, instead of being assigned their own individual nested TLV data objects. The length of A1/84 is simply the combination of three single bytes.	B	R	
//null	(1)	This is just a raw byte inside 84. As a simple byte, it has no tag-length-value structure of its own, so its Tag is shown here as **/null**, and its length is in parentheses to indicate it is not encoded as a TLV data object, it is simply a byte count that can be used to "slice apart" the contents of 84 into separate values.	B	R	
//null	(1)	This is another byte contained by 84.	B	R	
//null	(1)	This is another byte contained by 84.	B	R	

3.2.2 Message Structure

Each message type follow a specific structure, detailed in the subsections below.

3.2.2.1 Request Message

Table 3.2-2 - Request Message Format

Tag	Len	Value / Description	Typ	Req	Default
One byte standard **Start of Message** constant, not in TLV format. • 0xAA = Standard start of message byte					
One-byte standard **API Framework Version** constant, not in TLV format, used for tracking version compatibility. Devices that implement higher (newer) versions of the API framework are designed to be reasonably compatible with current and previous (lower) versions: • 0x00 = Pre-production development (limited breaking changes). At minimum, every device MagTek ever produces using the MMS framework will accept this value. • 0x01 = First Production release (no breaking changes) • 0x02 = Second Production release • Etc.					
81	var	Message Information	B	R	
/null	(1)	Message Type & Direction • 0x01 = Request from host to device • 0x81 = Request from device to host (Reserved)	B	R	
/null	(1)	Message Reference Number The host can use any value in this byte to help match responses with their corresponding requests. The device includes the same number in the corresponding response(s). MagTek recommends using a simple incrementing counter per request the host sends during a session.	B	R	
/null	(2)	Command ID	B	R	

Tag	Len	Value / Description	Typ	Req	Default
		The fully qualified Command number as defined in section **6 Commands**. The first byte is the Command Group number, which groups functionally related requests, and the second byte is the Command number within that group. If the **Request Payload** in the message contains wrappers, the host should specify the command it is invoking at the core of the request after all wrappers have been removed. This value is included primarily for performance and key conservation; the device simply uses it to make sure it is in a mode where it is capable of processing the desired command before it begins processing the **Request Payload**.			
/null	(var)	Reserved		O	
84	var	Request Payload As documented in the message's Request table in section **6 Commands**.	B	R	
9E	var	Reserved	B	O	

3.2.2.2 Response Message

Table 3.2-3 - Response Message Format

Tag	Len	Value / Description	Typ	Req	Default
One byte standard **Start of Message** constant, not in TLV format. • 0xAA = Standard start of message byte					
One-byte standard **API Framework Version** constant, not in TLV format, used for tracking version compatibility. Devices that implement higher (newer) versions of the API framework are designed to be reasonably compatible with current and previous (lower) versions: • 0x00 = Pre-production development (limited breaking changes). At minimum, every device MagTek ever produces using the MMS framework will accept this value. • 0x01 = First Production release (no breaking changes) • 0x02 = Second Production release • Etc.					
81	4	Message Information	B	R	
/null	(1)	Message Type & Direction • 0x02 = Response from host to device (Reserved) • 0x82 = Response from device to host		R	
/null	(1)	Message Reference Number The device echoes back whatever value the host populated in the Message Reference Number when it made the corresponding request. The host can use this value to match responses with their corresponding requests.		R	
/null	(2)	Response ID		R	

Tag	Len	Value / Description	Typ	Req	Default
		Matches the fully qualified Command number from the corresponding Command Request message, as defined in section **6.1 Command Group 0x10nn - Transactions**. • 0x1001 – Start Transaction			
/null	(var)	Reserved		O	
82	04	Response Status	B	R	
/null	(1)	Operation Status Summary • 0x00 = OK, Done • 0x01 = OK, Started / Running • 0x40 = OK, Done with Warnings • 0x41 = OK, Started / Running with Warnings • 0x80 = Failed to start operation (missing parameters, etc.) • 0x81 = Failed during operation	B	R	
/null	(1)	Operation Status Detail (Group) Operation Status Detail **Group**, **Subgroup**, and **Status Code** combine to provide a more detailed report than the Operation Status Summary. See **Table 3.2-4 - Operation Status Detail Codes** on page **35**. • 0x02 = Security / Permission Problems	B	R	
/null	(1)	Operation Status Detail (Subgroup) Operation Status Detail **Group**, **Subgroup**, and **Status Code** combine to provide a more detailed report than the Operation Status Summary. See **Table 3.2-4 - Operation Status Detail Codes** on page **35**. • 0x03 = Device State Issue	B	R	
/null	(1)	Operation Status Detail (Status Code) Operation Status Detail **Group**, **Subgroup**, and **Status Code** combine to provide a more detailed report than the Operation Status Summary. See **Table 3.2-4 - Operation Status Detail Codes** on page **35**. • 0x16 = Low Battery (5% or less)	B	R	
83	var	Additional Details If provided, use this value to help isolate the problem such as the specific parameter number or EMV tag. See the Command definition in section **6 Commands** for command-specific Additional Details.		O	
84	var	Response Payload As documented in the message's Response table in section **6 Commands**.	B	O	
9E	var	Reserved	B	O	

Table 3.2-4 - Operation Status Detail Codes

Source	Grp	Sub	Cde	Meaning
The **General** group 0x00 contains operation status detail codes related to the platform that do not originate from a specific functional module. • Subgroup 0x00 = General				
General	00	00	00	All good / requested operation was successful.
General	00	00	02	Requested Operation Failed
General	00	00	10	Setting up RTC data and time failure
General	00	00	11	Setting up RTC alarm failure
General	00	00	12	Key generation failure
General	00	00	13	Tamper setting is locked, can't be changed
General	00	00	14	Tamper setting requires system reset to continue
General	00	00	15	Tamper status can't be cleared, failure
General	00	00	16	Device has been tampered, need attention
General	00	00	17	Tamper module failed for other cases
General	00	00	18	Setting WLAN SoftAP password failure
The **Message Handler** group 0x01 contains operation status detail codes related to parsing and validating messages. • Subgroup 0x01 = Device issues that prevent Message Processing (e.g., Critical Battery, Pending Reset, System Failure, System Busy).				
Message	01	01	01	Generic Failure.
Message	01	01	02	Bad message parameter. The host has sent a message to the device that is not constructed properly.
Message	01	01	09	Device offline, can not process messages. For example, the device returns this detail code when it does not have keys injected or has registered a tamper.
Message	01	01	10	PIN Key Not Mapped.
Message	01	01	13	Feature Not Available
The **Request Handler** group 0x02 contains operation status detail codes related to starting actual command requests. • Subgroup 0x01 = Data issues (bad, missing, unknown…) • Subgroup 0x02 = Security / permission problems • Subgroup 0x03 = Device state issues (busy, not permitted, tampered, low battery) • Subgroup 0x04 = Device issues (missing hardware or features) • Subgroup 0x05 = TR31 Errors				
Request Handler	02	00	00	Reserved
Request Handler	02	01	00	Reserved
Request Handler	02	01	01	Generic Failure

Source	Grp	Sub	Cde	Meaning
Request Handler	02	01	02	Bad Message Parameter
Request Handler	02	01	03	Response Payload too big
Request Handler	02	01	07	Internal FW Failure
Request Handler	02	01	0A	Image Failure
Request Handler	02	01	19	Key does not exist
Request Handler	02	01	1A	Not Secured
Request Handler	02	01	1B	Passcode validation failed
Request Handler	02	01	1C	Device is locked
Request Handler	02	02	00	Reserved
Request Handler	02	03	04	Failed, device state issue, no transaction.
Request Handler	02	03	05	Failed, device state issue, cannot cancel.
Request Handler	02	03	08	Failed, device state, Transaction in Progress.
Request Handler	02	03	0C	Failed, device state, Signature Not allowed
Request Handler	02	03	0D	Failed, device state, Wrong Transaction State
Request Handler	02	03	0E	Failed, device state, Invalid PIN Entry State
Request Handler	02	03	0F	Failed, device state, PIN Entry in Session.
Request Handler	02	03	11	Failed, device state, Barcode Read in Progress.
Request Handler	02	03	12	Failed, device state, Pass-through command Not Activated.
Request Handler	02	03	14	Failed, device state, UI Settings in Progress.
Request Handler	02	03	15	Failed, device state, Buzzer in Progress
Request Handler	02	03	16	Failed, device state, Low Battery (5% or less)
Request Handler	02	03	17	Attempt to display Flexible UI page while touchscreen UI notifications enabled
Request Handler	02	03	18	Request is invalid while card emulation is in progress
Request Handler	02	04	13	Failed, BCR hardware not found.
Request Handler	02	05	01	Invalid TR31parameter
Request Handler	02	05	02	Invalid AES length
Request Handler	02	05	03	Invalid 16-Byte Boundary
Request Handler	02	05	04	Invalid Length in Message
Request Handler	02	05	05	Invalid number of optional KBH
Request Handler	02	05	06	Error in conversion of data type
Request Handler	02	05	07	Invalid KCV algorithm

Source	Grp	Sub	Cde	Meaning
Request Handler	02	05	08	Invalid KCV length
Request Handler	02	05	09	Invalid Optional KBH ID
Request Handler	02	05	0A	Invalid KBH ID
Request Handler	02	05	0B	Invalid algorithm used in KBH
Request Handler	02	05	0C	Invalid KBH usage
Request Handler	02	05	0D	Invalid KBH length
Request Handler	02	05	0E	Invalid version ID for key derivation
Request Handler	02	05	0F	Invalid KBH mode of use
Request Handler	02	05	10	TR31 engine not installed
Request Handler	02	05	11	Invalid Cryptographic operation
Request Handler	02	05	12	MAC Verification Failed
Request Handler	02	05	13	Error in Decrypting Key data
Request Handler	02	05	14	Error in computing MAC over entire message
Request Handler	02	05	15	Invalid MAC length
Request Handler	02	05	16	KDF Error
Request Handler	02	05	17	Buffer Insufficient
Request Handler	02	05	18	Invalid Storage KPM
Request Handler	02	05	19	Invalid Storage Secure RAM
Request Handler	02	05	1A	Invalid Key ID specified in option block
Request Handler	02	05	1B	Unsupported Key ID specified in option block
Request Handler	02	05	1C	Invalid Key ID Relationship
Request Handler	02	05	1D	Protection Key ID not loaded
Request Handler	02	05	1E	Invalid Data Tag MagTek Custom option block
Request Handler	02	05	1F	Invalid Kcv
Request Handler	02	05	20	Invalid Data
Request Handler	02	05	21	Invalid DUKPT key derivation
Request Handler	02	05	22	Invalid Exportability
Request Handler	02	05	23	Invalid Key Class
Request Handler	02	05	24	Invalid DSN
Request Handler	02	05	25	Invalid Challenge
Request Handler	02	05	26	Key Undeletable
Request Handler	02	05	27	Key not present
Request Handler	02	05	28	Unsupported Keyset ID

Source	Grp	Sub	Cde	Meaning
Request Handler	02	05	29	KPM Error
Request Handler	02	05	2A	Secure RAM Error
Request Handler	02	05	2B	Duplicated Key
Request Handler	02	05	2C	Invalid Key Usage Rule
Request Handler	02	05	2D	Selftest Key Corrupted
Request Handler	02	05	2E	Selftest System Key Bitmap Corrupted
Request Handler	02	05	2F	Selftest System Key Missing
Request Handler	02	05	30	Selftest System Key Not Loaded
Request Handler	02	05	31	Invalid Key Storage Limit
Request Handler	02	05	32	Duplicated Key set
Request Handler	02	05	33	Key Restriction
Request Handler	02	05	34	Key Transported by Weaker key
Request Handler	02	05	35	Repeat Key Agreement
Request Handler	02	05	36	Security not activated
Request Handler	02	05	37	Selftest key relocated
Request Handler	02	05	38	Invalid Selftest Scanned Versus Saved Bitmap

3.2.2.3 Notification Message

Table 3.2-5 - Notification Message Format

Tag	Len	Value / Description	Typ	Req	Default
One-byte standard **Start of Message** constant, not in TLV format. • 0xAA = Standard start of message byte					
One-byte standard **API Framework Version** constant, not in TLV format, used for tracking version compatibility. Devices that implement higher (newer) versions of the API framework are designed to be reasonably compatible with current and previous (lower) versions: • 0x00 = Pre-production development (limited breaking changes). At minimum, every device MagTek ever produces using the MMS framework will accept this value. • 0x01 = First Production release (no breaking changes) • 0x02 = Second Production release • Etc.					
81	4	Message Information	B	R	
/null	(1)	Message Type & Direction • 0x03 = Notification from host to device (Reserved) • 0x83 = Notification from device to host		R	
/null	(1)	Reserved, set to 0x00		R	
/null	(1)	Notification Source		R	

Tag	Len	Value / Description	Typ	Req	Default
		This byte and the **Notification Type** form the first two bytes of a unique six-byte Notification ID. Use this byte to look up the **Notification Group** subsection in **7 Notifications**. • 0x01 = Transaction • 0x09 = Firmware Update • 0x10 = Device • 0x18 = User Interface			
/null	(1)	Notification Type Append this byte to **Notification Source** to look up the **Notification 0xNNNN** subsection in the Notification Group subsection to find the documentation for the notification. • 0x01 = Information Update • 0x02 = Warning (e.g. low battery, reset soon) • 0x03 = Action Request (device is asking the host to take action) • 0x04 = Callback (request for action from host, device waits before continuing the related operation) • 0x05 = Operation Complete (either success or error)		R	
/null	(var)	Reserved		O	
82	(4)	Notification Detail Code These four bytes, when combined with **Notification Source** and **Notification Type**, form a six-byte unique ID for the notification. Each six-byte combination has a specific meaning and represents a different call to action for the host. See section **7 Notifications** for **Notification Detail Codes** tables containing information about the notification.	B	R	
/null	1	Category • 0x00 = Power/Reset	B	R	
/null	1	Reason • 0x02 = Battery	B	R	
/null	1	Reason Detail (Subgroup) • 0x01 = Power Down Imminent	B	R	
/null	1	Reserved, set to 0x00	B	R	
83	var	Additional Detail See the notification definition in section **7 Notifications** for notification-specific Additional Details.		O	
84	var	Notification Payload As documented in the message's Notification table in section **7 Notifications**.	B	O	
9E	var	Reserved	B	O	

3.2.2.4 Data File Message

This message type is used exclusively for transferring larger blocks of data that can overwhelm the standard message queue. The payload data for each message is always be treated as an individual file.

This message is only valid after a successful invocation of **Command 0xD811 - Start Send File to Device (Secured)**, **Command 0xD812 - Start Send File to Device (Unsecured)**, or **Command 0xD821 - Start Get File from Device**.

Table 3.2-6 - Data File Message Format

Tag	Len	Value / Description	Typ	Req	Default
One byte standard **Start of Message** constant, not in TLV format. • 0xAA = Standard start of message byte					
One-byte standard **API Framework Version** constant, not in TLV format, used for tracking version compatibility. Devices that implement higher (newer) versions of the API framework are designed to be reasonably compatible with current and previous (lower) versions: • 0x00 = Pre-production development (limited breaking changes). At minimum, every device MagTek ever produces using the MMS framework will accept this value. • 0x01 = First Production release (no breaking changes) • 0x02 = Second Production release • Etc.					
81	08	Message Information	B	R	
/null	(1)	Message Type & Direction • 0x04 = Data file from host to device • 0x84 = Data file from device to host	B	R	
/null	(1)	Message Reference Number The host can use any value in this byte to help match responses with their corresponding requests. The device includes the same number in the corresponding response(s). MagTek recommends using a simple incrementing counter per request the host sends during a session.	B	R	
/null	(2)	Command Number that prompted this message to be sent, as documented in **Command Group 0xD8nn - File Operations**.	B	R	
/null	(4)	File Type The file type as defined in **Command Group 0xD8nn - File Operations**	B	R	
84	var	File Payload As documented in section **6.6.1 About Files**.	B	R	

Table 3.2-7 - Data File Message Example

Example (Hex)
AA 00 81 08 84 08 D8 21 00 00 00 01 84 40 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F 30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F

4 Data Types and Shared TLV Data Objects

This section describes the primitive and composed data types referred to throughout this document.

4.1 Primitive Data Types

TLV data objects use the following primitive data types:

- A = Alphabetic (string, no numbers).
- AN = Alphanumeric (string).
- B = Binary value, which includes bit combinations ("OR" types).
- CN = Compressed numeric, as defined by ***EMV 4.3 Book 3***, section ***Data Element Format Conventions***.
- N = Numeric, as defined by ***EMV 4.3 Book 3***, section ***Data Element Format Conventions***.
- T = TLV **Constructed** data object (TLV Value contains additional layers of TLV-encoded data the parser should continue to process).

4.2 About Track Data

After the host receives and decrypts **EMV ARQC Type** data or **Merchant Data Container** data from the device, 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 4.2-1 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. Manually entered data does not include Track 3.

Table 4.2-1 - Example Generic ISO/ABA Track Data Format

Generic ISO/ABA Track Data Format	
Track 1 Data	%7555555555555555^CARDHOLDER NAME/^33338800044440000006?
Track 2 Data	;5555555555555555=33338800044440006?
Track 3 Data	;5555555555555555=333388000444400000006?

The example track data in **Table 4.2-1** can be interpreted as follows:

- The **%**, **?**, and **;** are Sentinels / delimiters, and are taken directly from the data on the card.
- 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**. Manually entered data uses **M**.
- The string of **5**s 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. Manually entered data uses string literal **MANUAL ENTRY/**.
- The string of **3**s is the Expiration Date (YYMM).
- The string of **8**s is the Service Code. For swiped credit / debit cards, this comes from the card. Manually entered data uses **000**.

- 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. Manually entered track data uses a MagTek standard for Discretionary Data as follows:
    - The string of 4s is the CVV2 a cardholder or operator entered on the keypad. This may be 3 or 4 characters long and is not padded, so the host software must find it by using the fixed-length padding and sentinels that surround it.
    - The strings of 0s are literals of fixed length: Track 1 has three zeroes after the Service Code, and five zeroes after the CVV2; Track 2 has three zeroes after the Service Code, and two zeroes after CVV2.
    - The 6 contains either a `0` or a `1`. This Field Option tells what data is included in the track data, where **0 = Acct, Date, CVV** and **1 = Name on Card, Acct, Date, CVV**.

4.3 Display Strings

The Display Strings type provides a pre-defined set of messages by string ID that the host and device use for various user interface features.

Table 4.3-1 - Display String IDs and Strings

Display String ID	Display String (en-US)
0x00	Reserved, do not use.
0x01	“AMOUNT”
0x02	“AMOUNT OK?”
0x03	“APPROVED”
0x04	“CALL YOUR BANK”
0x05	“CANCEL OR ENTER”
0x06	“CARD ERROR”
0x07	“DECLINED”
0x08	“ENTER AMOUNT”
0x09	Reserved, do not use.
0x0A	Reserved, do not use.
0x0B	“INSERT CARD”
0x0C	“NOT ACCEPTED”
0x0D	Reserved, do not use.
0x0E	“PLEASE WAIT”
0x0F	“PROCESSING ERROR”
0x10	“REMOVE CARD”
0x11	“USE CHIP READER”
0x12	“USE MAGSTRIPE”
0x13	“TRY AGAIN”

Display String ID	Display String (en-US)
0x14	“WELCOME”
0x15	“PRESENT CARD”
0x16	“PROCESSING”
0x17	“CARD READ OK - REMOVE CARD”
0x18	“INSERT OR SWIPE CARD”
0x19	“PRESENT ONE CARD ONLY”
0x1A	“APPROVED PLEASE SIGN”
0x1B	“AUTHORIZING PLEASE WAIT”
0x1C	“INSERT, SWIPE, OR TRY ANOTHER CARD”
0x1D	“PLEASE INSERT CARD”
0x1E	Null prompt (empty screen)
0x1F	Reserved, do not use.
0x20	“SEE PHONE”
0x21	“PRESENT CARD AGAIN”
0x22	“INSERT/SWIPE/TRY OTHER CARD”
0x23	“TAP or SWIPE CARD”
0x24	“TAP or INSERT CARD”
0x25	“TAP, INSERT or SWIPE CARD”
0x26	“TAP CARD”
0x27	“TIMEOUT”
0x28	“TRANSACTION TERMINATED”
0x29	“USE CHIP READER or MAGSTRIPE”
0x2A	“SCAN BARCODE”
0x2B	“BARCODE READ SUCCESSFULLY”
0x2C	“CANCELED”
0x2D	“SWIPE CARD or SCAN BARCODE”
0x2E	“INSERT CARD or SCAN BARCODE”
0x2F	“INSERT, SWIPE or SCAN BARCODE”
0x30	“TAP CARD or SCAN BARCODE”
0x31	“TAP, SWIPE or SCAN BARCODE”
0x32	“TAP, INSERT or SCAN BARCODE”
0x33	“TAP, INSERT, SWIPE or SCAN BARCODE”
0x34	“TRY ANOTHER INTERFACE”

Display String ID	Display String (en-US)
0x35	“NFC TAG DETECTED”
0x36	“ERROR REMOVE CARD”
0x37	“MIFARE CLASSIC 1K DETECTED”
0x38	“MIFARE CLASSIC 4K DETECTED”
0x39	“MIFARE DESFIRE DETECTED”

4.4 Encryption Type

The Encryption Type provides the key type, variant, and other information the host can use to decrypt encrypted data included in various payloads. The possible values are an ORed bitmask using the following elements:

- 0xxx xxxx = Fixed Key (Not used)
- 1xxx xxxx = DUKPT Key
- xx00 xxxx = TDES
- xx01 xxxx = AES128
- xx10 xxxx = AES256
- xxxx 0000 = Data Encrypt/Decrypt Variant
- xxxx 0001 = PIN Variant
- xxxx 0010 = MAC Variant
- xxxx 0011 = Data, Encrypt Variant
- xxxx 0100 = MAC Verify Variant
- xxxx 0101 = RESERVED
- xxxx 0110 = RESERVED
- xxxx 0111 = AES PIN Encrypt
- xxxx 1000 = AES MAC Generate
- xxxx 1001 = AES MAC Verify
- xxxx 1010 = AES MAC Generate/Verify
- xxxx 1011 = AES Data Encrypt
- xxxx 1100 = AES Data Decrypt
- xxxx 1101 = AES Data Encrypt/Decrypt
- xxxx 1110 = RESERVED
- xxxx 1111 = RESERVED

4.5 EMV ARQC Type

The device formats ARQC messages as shown in **Table 4.5-1**. The default is an EMV standard list of ARQC message tags. The host may also customize the contents of ARQC messages by setting **Property 1.1.1.1.1.2 EMV ARQC Message Tag List**.

4.5.1 EMV ARQC (DynaPro Format) Type

Table 4.5-1 - EMV ARQC (DynaPro Format) Type

Tag	Len	Value / Description	Typ	Req	Default
2-byte MSB message length excluding padding and CBC-MAC					
F9	var	Container for MAC structure and generic data	T	R	
/DFDF54	var	MAC KSN	B	R	
/DFDF55	01	MAC Encryption Type See section **4.4 Encryption Type** for a list of valid values.	B	R	
/DFDF25	var	Device Serial Number (IFD Serial Number)	B	R	
/FA	var	Container for generic data	T	R	
//70	var	Container for ARQC	T	R	
///82	02	Application Interchange Profile Available on: DynaProx FW Ver A8 or newer DynaFlex II FW Ver A6 or newer	B	O	
///9F6E	var	Third Party Data Available on: DynaProx FW Ver A8 or newer DynaFlex II FW Ver A6 or newer	B	O	
///DFDF53	01	Fallback Indicator • 0x00 = No Fallback • 0x01 = Technical Fallback • 0x81 = MSR Fallback	B	R	
///DFDF33	var	Masked Track 2 MSR Data If the payment method presented by the cardholder provides it	AN	O	
///DFDF4D	var	Masked Track 2 ICC Data If the payment method presented by the cardholder provides it	AN	O	
///DFDF52	01	Card Type • 0x00 = Othe • 0x04 = Unknown • 0x05 = Contact Chip Card (ICC) • 0x06 = Contactless Chip Card (PICC), EMV • 0x07 = Contact Chip Card (ICC) • 0x08 = Contactless PICC	B	R	
///FF42	var	Container for Selectable Encrypted Card Data Set up OID 1.1.2.6.1.1 to enable this container.	T	O	

Tag	Len	Value / Description	Typ	Req	Default
////DFDFDF37	var	Selectable Encrypted Data Primitive Decrypt the value of this TLV data object using the algorithm and variant specified in the **Selectable Encrypted Data KSN** parameter and the **Selectable Encrypted Data Encryption Type** parameter. See **Table 4.5-3 - EMV ARQC (DynaPro Format) DFDFDF37 Decrypted Contents** for the data structure as it should appear after decryption. (This item will be present if 0xFF42 is enabled)	B	O	
////DFDFDF38	0C	Selectable Encrypted Data KSN (This item will be present if 0xFF42 is enabled)	B	O	
////DFDFDF39	01	Selectable Encrypted Data Encryption Type (This item will be present if 0xFF42 is enabled)	B	O	
///DF2A	06	Tip Mode Sale Amount Entered	B	O	
///DF2B	06	Tip Mode Total Amount	B	O	
///DF5D	06	Tip Amount	B	O	
///DF5E	06	Tax Amount	B	O	
///F8	var	Container for Encrypted Data	T	R	
////DFDF59	var	Encrypted Data Primitive Decrypt the value of this TLV data object using the algorithm and variant specified in the **Encrypted Transaction Data KSN** parameter and the **Encrypted Transaction Data Encryption Type** parameter to read its contents. See **Table 4.5-2** on page **47** for the data structure as it should appear after decryption.	B	R	
////DFDF56	var	Encrypted Transaction Data KSN	B	R	
////DFDF57	01	Encrypted Transaction Data Encryption Type See section **4.4 Encryption Type** for a list of valid values.	B	R	
////DFDF58	01	Number of Padding Bytes Number of bytes added to DFDF59 value to force its length to a multiple of 8 bytes for TDES, or 16 bytes for AES.	B	R	
/FE	Var	VAS Data Container See **Table 4.5-7 – VAS Data Container Payload**	T	O	
/FF40	Var	Fleet Data Container (Common Kernel Only) See **Table 4.5 9 – Fleet Data Container Payload**	T	O	
Padding to ensure the length of data, starting with the message length at the very beginning, and ending with any additional padding, is a multiple of 8 bytes for TDES, or 16 bytes for AES. This is a requirement of using the CBC-MAC algorithm.					
Four-byte CBC-MAC. The host should calculate the CBC-MAC and verify that it matches. For details about calculating a CBC-MAC, see **About Message Authentication Codes (MAC)**.					

The device encrypts the value inside the **Encrypted Data Primitive** container using the **Encrypted Transaction Data Encryption Type** parameter and working key associated with the keyset number currently active in the device's configuration. As a requirement for using DUKPT encryption algorithms, the device pads it so the length of its value is a multiple of 8 bytes for TDES, or 16 bytes for AES. The device uses container DFDF58 to report how many bytes of data object DFDF59 are padding. Data object DFDF59 itself is formatted like **Table 4.5-2** after the host decrypts it.

Table 4.5-2 - EMV ARQC (DynaPro Format) DFDF59 Decrypted Contents

Tag	Len	Value / Description	Typ	Req	Default
FC	var	Decrypted Data Container Inside this container, the device inserts all EMV TLV data objects specified by the setting in **Property 1.1.1.1.1.2 EMV ARQC Message Tag List**. The remainder of this table shows the basic structure and content of MagTek custom tags. For definitions of all other standard EMV tags that can be included directly under container FC, see ***EMV 4.3 Book 3***.	T	R	
/DF29	08	Only if tag DF29 is included in **Property 1.1.1.1.1.2 EMV ARQC Message Tag List**. Outcome Parameter Set Byte 1 - Outcome 0x10 = Approved 0x20 = Declined 0x30 = Online Request 0x40 = End Application 0x50 = Select Next Application 0x60 = Try Another Interface 0x70 = Try Again 0xF0 = N/A Byte 2 – Entry Point Start 0x00 = Start A 0x10 = Start B 0x20 = Start C 0x30 = Start D 0xF0 = N/A Byte 3 – Entry Point Online Response 0x00 = EMV Data 0x10 = Any 0xF0 = N/A Byte 4 – CVM 0x00 = No CVM 0x10 = Obtain Signature 0x20 = Online PIN 0x30 = Confirmation Code Verified	B	O	

Tag	Len	Value / Description	Typ	Req	Default
		0xF0 = N/A Byte 5 – UI/Data/Receipt 0x80 = UI Request on Outcome Present 0x40 = UI Request on Restart Present 0x20 = Data Record Present 0x10 = Discretionary Data Present 0x08 = Provide Receipt Byte 6 – Alternate Interface Preference 0x10 = Contact 0x20 = MSR 0xF0 = N/A Byte 7 – Field Off Request FF = N/A Byte 8 – Removal Timeout			
Padding to force DFDF59 plus padding to be a multiple of 8 bytes					

Table 4.5-3 - EMV ARQC (DynaPro Format) DFDFDF37 Decrypted Contents

Tag	Len	Value / Description	Typ	Req	Default
FC	var	Decrypted Data Container Inside this container, the device inserts EMV TLV data objects specified by the setting in Property 1.1.1.1.1.2 EMV ARQC Message Tag List. If the data is not available for a given selected card data, the tag will still get transmitted with a length of '1' and value = '*'.	T	R	
/5F20	var	Only if Byte 0 – Bit 0 is set in Property 1.1.2.6.1.1 Selectable Card Data Encryption Enable. Cardholder Name	an	O	
/5F24	02/ 03	Only if Byte 0 – Bit 2 is set in Property 1.1.2.6.1.1 Selectable Card Data Encryption Enable. Expiration Date, YYMM or YYMMDD	n4/ n6	O	
/5F30	02	Only if Byte 0 – Bit 3 is set in Property 1.1.2.6.1.1 Selectable Card Data Encryption Enable. Service Code	n3	O	
/9F1F	var	Only if Byte 0 – Bit 4 is set in Property 1.1.2.6.1.1 Selectable Card Data Encryption Enable. T1 Discretionary Data	an	O	

Tag	Len	Value / Description	Typ	Req	Default
/9F20	var	Only if Byte 0 – Bit 5 is set in Property 1.1.2.6.1.1 Selectable Card Data Encryption Enable. T2 Discretionary Data	cn	O	
Padding to force DFDFDF37 plus padding to be a multiple of 16 bytes for AES encryption.					

Table 4.5-4 – EMV ARQC Enhanced DFDF59 Decrypted Contents for EMV Data

Tag	Len	Value / Description	Typ	Req	Default
FC	var	Decrypted Data Container This contains all EMV TLV data objects specified in **Property 1.1.1.1.1.2 EMV ARQC Message Tag List**.	T	R	
/FE	Var	VAS Data Container See **Table 4.5-7 – VAS Data Container Payload**	T	O	
Padding to ensure the length of data, starting with the message length at the very beginning, and ending with any additional padding, is a multiple of 8 bytes for TDES, or 16 bytes for AES. This is a requirement of using the CBC-MAC algorithm.					

Table 4.5-5 – EMV ARQC Enhanced DFDF59 Decrypted Contents for MSR and MagnePrint Data

Tag	Len	Value / Description	Typ	Req	Default
FC	var	Decrypted Data Container Inside this container, the device inserts all EMV TLV data objects specified by the setting in **Property 1.1.1.1.1.2 EMV ARQC Message Tag List**. The remainder of this table shows the basic structure and content of MagTek custom tags. For definitions of all other standard EMV tags that can be included directly under container FC, see ***EMV 4.3 Book 3***.	T	R	
/9F41	04	Transaction Counter Starts at 00000000 each time the device powers up or resets, increments for each transaction.	B	R	
/DFDF36	01	MSR Track 1 Status • 0x00 = OK • 0x01 = Empty • 0x02 = Error • 0x03 = Disabled	B	O	
/DF41	var	MSR Track 1 Clear Text	AN	O	
/DFDF38	01	MSR Track 2 Status • 0x00 = OK • 0x01 = Empty • 0x02 = Error • 0x03 = Disabled	B	O	
/DF42	var	MSR Track 2 Clear Text	AN	O	

Tag	Len	Value / Description	Typ	Req	Default
/DFDF3A	01	MSR Track 3 Status • 0x00 = OK • 0x01 = Empty • 0x02 = Error • 0x03 = Disabled	B	O	
/DF43	var	MSR Track 3 Clear Text	AN	O	
/DFDF43	04	MagnePrint Status The device only includes this if MSR and MagnePrint data are both included in the transaction and the device is configured to encrypt them using the same key, to avoid consuming two DUKPT keys encrypting separate containers. If the device is configured to encrypt MSR and MagnePrint data using different keys, it provides MagnePrint data in the **Container for Encrypted MagnePrint Data** instead. • Bit 0 = MagnePrint Capable Flag o 0 = Device is not MagnePrint capable o 1 = Device is MagnePrint capable • Bits 1 through 3 = Mode o 0 = Standard MagnePrint o 1 = Extended MagnePrint • Bits 4 through 15 = ASIC Revision • Bit 16 = Reserved • Bit 17 = Reserved • Bit 18 = Swipe too slow • Bit 19 = Swipe too fast • Bit 20 = Reserved • Bit 21 = Card swipe direction o 0 = Forward o 1 = Reverse • Bits 22..31 = Reserved	B	O	
/DF44	var	MagnePrint Data The device only includes this if MSR and MagnePrint data are both included in the transaction and the device is configured to encrypt them using the same key. The host can use this data in conjunction with Magensa services to determine whether the swiped card is authentic.	B	O	
/FE	Var	VAS Data Container See **Table 4.5-7 – VAS Data Container Payload**	T	O	
Padding to ensure the length of data, starting with the message length at the very beginning, and ending with any additional padding, is a multiple of 8 bytes for TDES, or 16 bytes for AES. This is a requirement of using the CBC-MAC algorithm.					

Table 4.5-6 – EMV ARQC Enhanced DFDF59 Decrypted Contents for MagnePrint Data

Tag	Len	Value / Description	Typ	Req	Default
FC	var	Decrypted Data Container	T	R	
/DFDF43	var	MagnePrint Status The device only includes this when MSR and MagnePrint data are included in the transaction, but the device is configured to encrypt them using a different key. • Bit 0 = MagnePrint Capable Flag o 0 = Device is not MagnePrint capable o 1 = Device is MagnePrint capable • Bits 1..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 = Card swipe direction o 0 = Forward o 1 = Reverse • Bits 22..31 = Reserved	B	R	
/DF44	var	MagnePrint Data The host can use this data in conjunction with Magensa services to determine whether the swiped card is authentic.	B	R	
/DF4B	var	MSR PAN	B	R	
/FE	Var	VAS Data Container See **Table 4.5-7 – VAS Data Container Payload**	T	O	
Padding to ensure the length of data, starting with the message length at the very beginning, and ending with any additional padding, is a multiple of 8 bytes for TDES, or 16 bytes for AES. This is a requirement of using the CBC-MAC algorithm.					

Table 4.5-7 – VAS Data Container Payload

Tag	Len	Value / Description	Typ	Req	Default
/FE	var	VAS Data Container	T	O	
//FF01	var	Apple VAS Container Slot 1 Container	B	O	
///9F27	var	VAS Data Up to 128 bytes.	B	O	
///9F2A	var	Mobile Token Up to 36 bytes.	B	O	
//FF02	var	Apple VAS Container Slot 2 Container	B	O	
///9F27	var	VAS Data Up to 128 bytes.	B	O	
///9F2A	var	Mobile Token	B	O	

Tag	Len	Value / Description	Typ	Req	Default
		Up to 36 bytes.			
//FF03	var	Apple VAS Container Slot 3 Container	B	O	
///9F27	var	VAS Data Up to 128 bytes.	B	O	
///9F2A	var	Mobile Token Up to 36 bytes.	B	O	
//FF04	var	Apple VAS Container Slot 4 Container	B	O	
///9F27	var	VAS Data Up to 128 bytes.	B	O	
///9F2A	var	Mobile Token Up to 36 bytes.	B	O	
//FF05	var	Apple VAS Container Slot 5 Container	B	O	
///9F27	var	VAS Data Up to 128 bytes.	B	O	
///9F2A	var	Mobile Token Up to 36 bytes.	B	O	
//FF06	var	Apple VAS Container Slot 6 Container	B	O	
///9F27	var	VAS Data Up to 128 bytes.	B	O	
///9F2A	var	Mobile Token Up to 36 bytes.	B	O	
//FF41	var	Google Smart Tap Container	B	O	
///FF01	var	Collector ID Slot 1 Container	B	O	
////DF7B	var	Service Response NDEF Record	B	O	
///FF02	var	Collector ID Slot 2 Container	B	O	
////DF7B	var	Service Response NDEF Record	B	O	
///FF03	var	Collector ID Slot 3 Container	B	O	
////DF7B	var	Service Response NDEF Record	B	O	
///FF04	var	Collector ID Slot 4 Container	B	O	
////DF7B	var	Service Response NDEF Record	B	O	
///FF05	var	Collector ID Slot 5 Container	B	O	
////DF7B	var	Service Response NDEF Record	B	O	
///FF06	var	Collector ID Slot 6 Container	B	O	
////DF7B	var	Service Response NDEF Record	B	O	

Table 4.5-9 – Fleet Data Container Payload (Common Kernel Only)

Tag		Len	Value / Description	Typ	Req	Default
/FF40		var	Fleet Data Container	T	O	
//DF30		var	Prompting	B	O	
//DF32		var	Purchase Restrictions	B	O	
//DF33		var		B	O	
//DF34		var	Chip Offline purchase Restrictions for Fuel	B	O	
//DF35		var	Chip Offline purchase Restrictions for Non-fuel	B	O	
//DF36		var	Relationship Codes	B	O	
//DF37		var	3rd Party Reference Data Generation 2	B	O	
//DF38		var	Loyalty ID	B	O	
//DF39		var	Purchase Device Sequence Number	B	O	
//DF40		var	Generic Tag	B	O	
//DF41		var	Vehicle/Trailer Number	B	O	
//DF42		var	Vehicle Tag	B	O	
//DF43		var	Driver ID	B	O	
//DF44		var	Driver's License Number	B	O	
//DF45		var	Driver's License State/Province Abbreviation	B	O	
//DF46		var	Driver's License Name Abbreviation	B	O	
//DF47		var	Date of Birth	B	O	
//DF48		var	Zip/Postal Code	B	O	
//DF49 – //DF51		var	IFSR Reserved for Future Use	B	O	
//DF52		var	Trailer Number	B	O	
//DF53		var	Employee Number	B	O	
//DF54		var	Work Order / Purchase Order Number	B	O	
//DF55		var	Additional Prompted Data 1	B	O	
//DF56		var	Additional Prompted Data 2	B	O	
//DF57		var	Proprietary Data	B	O	
//9F5A		var		B	O	
//9F0A		var	ASRPD	B	O	
//9F6E		var	M/C Fleet	B	O	
//9FD4		var		B	O	
//9F50		var		B	O	

4.6 EMV ARPC Type

Table 4.6-1 – EMV ARPC Data

Tag	Len	Value / Description	Typ	Req	Default
FF74	var	Container for non-MAC ARPC	T	R	
/DFDF25	var	Device Serial Number (IFD Serial Number)	B	R	
/FA	var	Container for generic data	T	R	
//70	var	Container for ARPC	T	R	
///8A	02	Authorization Response Code • '00' = Approved • '01' = Issuer Referral • '05' = Declined • '12' = Switch Interface • '13' = Request Online PIN	AN	R	
///91	var	Issuer Authentication Data As defined in ***EMV Integrated Circuit Card Specifications for Payment Systems 4.3***	B	O	
///71	var	Issuer Script Template 1 As defined in ***EMV Integrated Circuit Card Specifications for Payment Systems 4.3. The host may include as many instances of this parameter as needed, up to a maximum length of 128 bytes including Tags and Lengths.***	B	O	
///72	var	Issuer Script Template 2 As defined in ***EMV Integrated Circuit Card Specifications for Payment Systems 4.3. The host may include as many instances of this parameter as needed, up to a maximum length of 128 bytes including Tags and Lengths.***	B	O	

4.7 EMV Batch Data Type

The device formats EMV batch data, such as merchant data and pre-defined EMV batch data tags, using the format shown in **Table 4.7-1**. The default is an EMV standard list of batch data message tags. The host may also customize the contents of batch data messages by setting **Property 1.1.1.1.1.3 EMV Batch Data Tag List**.

(EMV Contact Only) For unsuccessful transactions, this data object can contain additional pre-defined reversal data. It is normally used by the host for data capture. The default is an EMV standard list of reversal data message tags. The host may also customize the contents of reversal data messages by setting **Property 1.1.1.1.1.4 EMV Reversal Data Tag List**.

As part of successful completion of **Command 0x1001 - Start Transaction**, this data structure contains the results of the transaction. The set of tags used during a given EMV transaction is a combination of the tags defined in the EMV specification and the tags that are specific to the kernel being used for the transaction.

4.7.1 EMV Batch Data (DynaPro Format) Type

Table 4.7-1 – EMV Batch Data (DynaPro Format) Type

Tag	Len	Value / Description	Typ	Req	Default
2-byte MSB message length excluding padding and CBC-MAC					
F9	var	Container for MAC structure and generic data	T	R	
/DFDF54	var	MAC KSN	B	R	
/DFDF55	01	MAC Encryption Type See section **4.4 Encryption Type** for a list of valid values.	B	R	
/DFDF25	var	Device Serial Number (IFD Serial Number)	B	R	
/FA	var	Container for Generic Data	T	R	
//F0	var	Transaction Results	T	R	
///F1	var	Container for Status Data	T	R	
////DFDF1A	01	Transaction Status • 0x00 = Accept • 0x01 = Decline • 0x02 = Error	B	R	
////DFDF1B	01	Additional Transaction Information 0x00	B	R	
///F8	var	Container for Encrypted Data	T	R	
////DFDF59	var	Encrypted Data Primitive Decrypt the value of this TLV data object according to the **Encrypted Transaction Data KSN** parameter and the **Encrypted Transaction Data Encryption Type** parameter to read its contents. See **Table 4.7-2** on page **56** for the data structure as it should appear after decryption. Use the data variant of the current MSR DUKPT working key used in the relevant transaction.	B	R	
////DFDF56	var	Encrypted Transaction Data KSN	B	R	
////DFDF57	01	Encrypted Transaction Data Encryption Type See section **4.4 Encryption Type** for a list of valid values.	B	R	
////DFDF58	01	Number of padding bytes added to DFDF59 value to force length to a multiple of 8 bytes	B	R	
///F7	var	Merchant Data This contains an instance of **Merchant Data Container**.	T	R	

Tag	Len	Value / Description	Typ	Req	Default
/FE	Var	VAS Data Container See **Table 4.5-7 – VAS Data Container Payload**	T	O	
Padding to ensure the length of data, starting with the message length at the very beginning, and ending with any additional padding, is a multiple of 8 bytes for TDES, or 16 bytes for AES. This is a requirement of using the CBC-MAC algorithm.					
Four-byte CBC-MAC. The host should calculate the CBC-MAC and verify that it matches. For details about calculating a CBC-MAC, see **About Message Authentication Codes (MAC)**.					

Table 4.7-2 – EMV Batch Data (DynaPro Format) DFDF59 Decrypted Contents

Tag	Len	Value / Description	Typ	Req	Default
FC	var	Decrypted Data Container	T		
/F2	var	Container for Batch Data This data object contains the set of EMV TLV data objects specified in **Property 1.1.1.1.1.3 EMV Batch Data Tag List**.	T		
//DF29	08	Only if tag DF29 is included in **Property 1.1.1.1.1.3 EMV Batch Data Tag List** Outcome Parameter Set Byte 1 - Outcome 0x10 = Approved 0x20 = Declined 0x30 = Online Request 0x40 = End Application 0x50 = Select Next Application 0x60 = Try Another Interface 0x70 = Try Again 0xF0 = N/A Byte 2 – Entry Point Start 0x00 = Start A 0x10 = Start B 0x20 = Start C 0x30 = Start D 0xF0 = N/A Byte 3 – Entry Point Online Response 0x00 = EMV Data 0x10 = Any 0xF0 = N/A Byte 4 – CVM 0x00 = No CVM 0x10 = Obtain Signature	B	O	

Tag	Len	Value / Description	Typ	Req	Default
		0x20 = Online PIN 0x30 = Confirmation Code Verified 0xF0 = N/A Byte 5 – UI/Data/Receipt 0x80 = UI Request on Outcome Present 0x40 = UI Request on Restart Present 0x20 = Data Record Present 0x10 = Discretionary Data Present 0x08 = Provide Receipt Byte 6 – Alternate Interface Preference 0x10 = Contact 0x20 = MSR 0xF0 = N/A Byte 7 – Field Off Request FF = N/A Byte 8 – Removal Timeout			
/F3	var	(EMV Contact Only) Container for Reversal Data, if any This data object contains the set of EMV TLV data objects specified in **Property 1.1.1.1.1.4 EMV Reversal Data Tag List**.	T	O	null
null	(var)	Padding to force DFDF59 plus padding to be a multiple of 8 bytes	B		

Table 4.7-3 - EMV Batch Data (DynaPro Format) DFDF59 Decrypted Contents

Tag	Len	Value / Description	Typ	Req	Default
FC	var	Decrypted Data Container	T		
/F2	var	Container for Batch Data This data object contains the set of EMV TLV data objects specified in **Property 1.1.1.1.1.3 EMV Batch Data Tag List**.	T		
/F3	var	Container for Reversal Data, if any This data object contains the set of EMV TLV data objects specified in **Property 1.1.1.1.1.4 EMV Reversal Data Tag List**.	T	O	null
null	(var)	Padding to force DFDF59 plus padding to be a multiple of 8 bytes or 16 bytes depending on the cipher block size of the algorithm being used.	B		
/FE	Var	VAS Data Container See **Table 4.5-7 – VAS Data Container Payload**	T	O	/FE

4.7.2 Merchant Data Container

Merchant Data is normally used by the host for receipt printing. The contents of this container are not customizable.

Table 4.7-4 - Merchant Data Container

Tag	Description	Source	Format	Length
DFDF40	Signature Status • 0x00 = Signature Not Required • 0x01 = Signature Required • 0x02 = Signature Available • 0x03 = Signature Succeeded • 0x04 = Signature Failed	Device	b	1
5F25	EMV Application Effective Date	Card	n6	3
5F24	EMV Application Expiration Date	Card	n6	3
89	Authorization Code	Device	b	6
5F2A	Transaction Currency Code	Card	n3	2
9F02	Amount, authorized	Device	n12	6
9F03	Amount, other	Device	n12	6
9F06	AID - terminal	Device	b	5..16
9F12	EMV Application Preferred Name	Card	ans	1..16
9F1C	Terminal ID	Device	an 8	8
9F39	POS Entry Mode	Device	n2	1
9C	Transaction Type	Device	n2	1
9F34	Indicates the results of the last CVM performed	Device	b	3
5F57	Account Type	Device	n2	1
5F34	PAN Sequence Number	Card	n2	1
DFDF4D	Masked ICC Track 2 Data	Card	ans	30..38

4.8 EMV Terminal Configuration File Type

The host uses **Command 0xD812 - Start Send File to Device (Unsecured)** to load this file type to control the behavior of the device's EMV contact kernel. The configuration loaded using this file type must be designed to work together with all instances of **EMV Processing Configuration File Type** and **EMV Entry Point Configuration File Type** the host loads into the device.

MagTek provides tools that allow these settings to be loaded using a Microsoft Excel spreadsheet for more convenient authoring, review, and change tracking. For a reference sample spreadsheet that contains EMVCo approved configurations, contact MagTek Support Services.

This document shows one example of the available Contact Level 2 certified configurations (***DynaFlex C01, Merchant, Attended, ODA***). To see which configurations are supported on the devices you are using, see the list of **Vendor Config IDs** in the device's ***Letter of Approval for Contact Level 2*** posted

in the list of ***Approved / Evaluated*** products on the EMVCo web site. For detailed descriptions of the tags included in this file type, including possible valid values and their effects on device behavior, see ***EMV Integrated Circuit Card Specifications for Payment Systems v4.3***.

Table 4.8-1 - EMV Configuration Terminal File Type

Tag	Len	Value / Description	Typ	Req	Example
File Type Version One byte indicating the version of the file type format being used. • 0xAA					
SHA-1 Hash 20-byte hash of all values that follow.					
9F1A	02	Terminal Country Code	B	R	08 40
DF79	01	Cardholder Confirmation	B	R	01
9F35	01	Terminal Type	B	R	21
DF0A	01	EMV Contact Supported	B	R	01
9F33	03	Terminal Capabilities	B	R	E0 28 C8
9F40	05	Additional Terminal Capabilities	B	R	EF 80 F0 A0 01
DF55	01	EMV Contactless Supported	B	R	01
DF0B	01	Magnetic Stripe Supported	B	R	01
DF27	01	Time allocated to enter a PIN	B	R	0A
DF06	01	Batch / Online Data Capture managed	B	R	01
DF08	00	Advice Managed	B	R	00
DF7A	01	PSE Supported	B	R	01
DF0D	00	AutoRun Mode	B	R	00
DF10	03	Predefined amount for AutoRun mode	B	R	00 00 00
DF7B	01	PIN Bypass Supported	B	R	00
DF07	01	Referral Managed	B	R	01
DF09	01	Default TAC supported when regular TACs are not present	B	R	01
DF73	05	Default TAC default	B	R	00 00 00 00 00
DF74	05	Default TAC denial	B	R	00 00 00 00 00
DF75	05	Default TAC online	B	R	00 00 00 00 00
DF53	01	Random Transaction Selection not supported	B	R	00
DF54	01	Velocity Checking not supported	B	R	00
DF7C	01	CDA Mode	B	R	01

4.9 EMV Processing Configuration File Type

The host uses **Command 0xD812 - Start Send File to Device (Unsecured)** to load this file type to control the behavior of the device's EMV kernels. The host must compile a single instance of this file type containing multiple instances of the **AID Delimiter Container**, one for each contact or contactless AID the device should support. For each instance of the **AID Delimiter Container** where tag 9F01 is set to a contactless AID, the host must load a corresponding instance of an Entry Point Table when it loads the **EMV Entry Point Configuration File Type** .

Table 4.9-1 - EMV Configuration Processing File Type

File Type Version One byte indicating the version of the file type format being used. • 0xAA					
SHA-1 Hash 20-byte hash of all values that follow.					
FF33	var	AID Delimiter Container There can be multiple instances of this in sequence. The contents of the first instance are loaded into Processing Table Slot 1, the contents of the second are loaded into Processing Table Slot 2, etc.	T	R	
/9F01	06	Payment Brand Identifier This serves as supporting information to clarify whether this instance of the AID Delimiter Container is for Contact or Contactless. Byte 1 upper nibble must be set to a value flagging that it corresponds to a Contactless AID, generally by using 0xC0 or 0xF0. For Contact AID, the value of Byte 1 is 00. Byte 1 lower nibble: • 0 = Contact • 1 = Interac (Common Kernel Only) • 2 = Mastercard Contactless • 3 = Visa payWave • 4 = Expresspay • 5 = JCB (Common Kernel Only) • 6 = Discover D-PAS • 7 = China UnionPay (Common Kernel Only) Bytes 2..5 Reserved for future use	B	R	F2 00 00 00 00 00
/4F	0..16	Application Identifier (AID)	B	R	A0 00 00 00 04 10 10
/DF7E	01	ASI	B	R	01

/9F09	02	Application Version Only applies when Payment Brand Identifier indicates Contact.	B	R	00 00
/DF11	01	Skip TAC/IAC default supported Only applies when Payment Brand Identifier indicates Contact.	B	R	00
/DF12	01	Random transaction selection supported Only applies when Payment Brand Identifier indicates Contact.	B	R	00
/DF13	01	Velocity checking supported Only applies when Payment Brand Identifier indicates Contact.	B	R	00
/DF14	01	Floor limit checking supported Only applies when Payment Brand Identifier indicates Contact.	B	R	00
/DF15	01	TAC supported Only applies when Payment Brand Identifier indicates Contact.	B	R	00
/DF20	05	TAC default Only applies when Payment Brand Identifier indicates Contact.	B	R	00 00 00 00 00
/DF21	05	TAC denial Only applies when Payment Brand Identifier indicates Contact.	B	R	00 00 00 00 00
/DF22	05	TAC online Only applies when Payment Brand Identifier indicates Contact.	B	R	00 00 00 00 00
/9F1B	04	Floor limit Only applies when Payment Brand Identifier indicates Contact.	B	R	00 00 00 00
/DF70	01	Target percentage Only applies when Payment Brand Identifier indicates Contact.	B	R	00
/DF6E	03	Threshold value Only applies when Payment Brand Identifier indicates Contact.	B	R	00 00 00
/DF6F	01	Maximum target percentage Only applies when Payment Brand Identifier indicates Contact.	B	R	00
/DF01	01	Default DDOL supported Only applies when Payment Brand Identifier indicates Contact.	B	R	00

/DF71	0..FC	DDOL Only applies when Payment Brand Identifier indicates Contact.	B	R	
/DF02	01	Default TDOL supported Only applies when Payment Brand Identifier indicates Contact.	B	R	00
/DF72	0..252	TDOL Only applies when Payment Brand Identifier indicates Contact.	B	R	
/5F2A	02	Currency Code Only applies when Payment Brand Identifier indicates Contact.	B	R	00 00
/5F36	01	Transaction currency exponent Only applies when Payment Brand Identifier indicates Contact.	B	R	00
Additional instances of the **AID Delimiter Container** parameter, one per Application Identifier (AID) the device should support.					

4.10 EMV Entry Point Configuration File Type

The host uses **Command 0xD812 - Start Send File to Device (Unsecured)** to load this file type to control the behavior of the device's EMV kernels.

Table 4.10-1 - EMV Entry Point Configuration File Type Header

Tag	Value (hex)	Description
File Type Version One byte indicating the version of the file type format being used. • 0xAA		
SHA-1 Hash 20-byte hash of all values that follow		
One or more instances of the following: • **4.10.1 Mastercard MCL Entry Point** Table • **4.10.2 Visa payWave Entry Point** Table • • **4.10.3 American Express Expresspay Entry** Point Table • **4.10.4 Discover D-PAS Entry Point** Table • **4.10.5 China Unionpay Entry Point Table (Common Kernel** Only) • **4.10.6 JCB Entry Point Table (Common Kernel** Only) • **4.10.7 Interac Flash Entry Point Table (Common Kernel** Only)		

Tag	Value (hex)	Description
The host should include an Entry Point Table for each transaction type to be supported by each contactless payment brand Application Identifier (AID) listed in the loaded **EMV Processing Configuration File Type** . For example, if the device should support three transaction types across each of four payment brands, it would combine 12 total Entry Point Tables into a single file instance before loading.		

4.10.1 Mastercard MCL Entry Point Table

Table 4.10-2 - Mastercard MCL Entry Point Table

Tag	Len	Value / Description	Typ	Req	Example
FF35	var	AID Delimiter Container There can be multiple instances of this in sequence. The contents of the first instance are loaded into Entry Point Table Slot 1, the contents of the second are loaded into Entry Point Table Slot 2, etc.	T	R	
/DF0E	03	Kernel ID, Processing Slot, Transaction Type Byte 1 Kernel ID • 0x02 = MasterCard Contactless (MCL) Byte 2 Processing Slot to Use See the **AID Delimiter Container** parameter in **EMV Processing Configuration File Type** for information about how to identify slots. Byte 3 Transaction Type • 0x00 = Purchase • 0x01 = Cash • 0x02 = Purchase with cashback • 0x03 = Refund	B	R	02 01 00
/DF0F	var	Payload Delimiter Container Include only one of these containers inside each **AID Delimiter Container**.	T	R	
//9F1A	02	Terminal Country Code	B	R	08 40
//9F35	01	Terminal Type	B	R	21
//9F40	05	Additional Terminal Capabilities	B	R	00 00 00 00 00
//9F7E	01	Mobile Support Indicator	B	R	01
//DF0C	01	Kernel ID	B	R	02

Tag	Len	Value / Description	Typ	Req	Example
//DF1B	01	Kernel Configuration • Bit 8 = MSD Mode Not Supported (Not Used, kernel does not support MSD, always set to 0) • Bit 7 = EMV Mode contactless transaction not supported (Always set to 0) • Bit 6 = On-Device-CVM Supported • Bit 5 = Relay Resistance Protocol Supported • Bit 4..1 = Reserved for future use	B	R	20
//DF2D	03	Message Hold Time (100 of ms)	B	R	00 00 0D
//9F6D	02	Magnetic Stripe Application Version Number This value only applies when the Kernel Configuration parameter is set to support MSD. The device ignores this value.	B	R	00 01
//DF1A	03	Magnetic Stripe Default UDOL This value only applies when the Kernel Configuration parameter is set to support MSD. The device ignores this value.	B	R	9F 6A 04
//DF1E	01	CVM Capability - CVM Required This value only applies when the Kernel Configuration parameter is set to support MSD. The device ignores this value. • Bits 8..5: o 0000 = No CVM o 0001 = Obtain Signature o 0010 = Online PIN o Others = Reserved for future use • Bits 4..1 = Reserved for future use	B	R	00
//DF2C	01	CVM Capability - No CVM Required This value only applies when the Kernel Configuration parameter is set to support MSD. The device ignores this value. • Bits 8..5: o 0000 = No CVM o 0001 = Obtain Signature o 0010 = Online PIN o Others = Reserved for future use • Bits 4..1 = Reserved for future use	B	R	00
//9F09	02	EMV Application Version Number	B	R	00 02

Tag	Len	Value / Description	Typ	Req	Example
//DF03	01	Security Capabilities • Bit 8 = SDA • Bit 7 = DDA • Bit 6 = Card Capture • Bit 5 = Reserved for future use • Bit 4 = CDA • Bits 3..1 = Reserved for future use	B	R	08
//DF17	01	Card Data Input Capabilities • Bit 8 = Manual Key Entry • Bit 7 = MSR • Bit 6 = ICC • Bits 5..1 = Reserved for future use	B	R	60
//DF18	01	CVM Capability - CVM Required • Bit 8 = Offline Plaintext PIN • Bit 7 = Enciphered Online PIN • Bit 6 = Signature • Bit 5 = Enciphered Offline PIN • Bit 4 = No CVM • Bits 3..1 = Reserved for future use	B	R	28
//DF19	01	CVM Capability - No CVM Required • Bit 8 = Offline Plaintext PIN • Bit 7 = Enciphered Online PIN • Bit 6 = Signature • Bit 5 = Enciphered Offline PIN • Bit 4 = No CVM • Bits 3..1 = Reserved for future use	B	R	08
//DF1C	02	Max Lifetime Torn Transaction(s)	B	R	01 2C
//DF1D	01	Max Number Torn Transaction	B	R	00
//DF20	05	Terminal Action Code - Default	B	R	00 00 00 00 00
//DF21	05	Terminal Action Code - Denial	B	R	00 00 00 00 00
//DF22	05	Terminal Action Code - Online	B	R	00 00 00 00 00
//DF04	0	Balance Read Before GenAC	B	R	
//DF05	0	Balance Read After GenAC	B	R	
//DF23	06	Reader Contactless Floor Limit	B	R	00 00 00 01 00 00
//DF24	06	Reader Contactless Transaction Limit (No On-Device CVM)	B	R	00 00 00 03 00 00

Tag	Len	Value / Description	Typ	Req	Example
//DF25	06	Reader Contactless Transaction Limit (On-Device CVM)	B	R	00 00 00 05 00 00
//DF26	06	Reader CVM Required Limit	B	R	00 00 00 00 10 00
//DF27	02	Timeout Value (ms)	B	R	13 88
//DF30	01	Hold time value before field off (100 of ms)	B	R	0D
//DF32	02	Minimum Relay Resistance Grace Period (100 of micro sec)	B	R	00 14
//DF33	02	Maximum Relay Resistance Grace Period (100 of micro seconds)	B	R	00 32
//DF34	02	Terminal Expected Transmission Time for Relay Resistance C-APDU (100 of micro seconds)	B	R	00 12
//DF35	02	Terminal Expected Transmission Time for Relay Resistance R-APDU (100 of micro seconds)	B	R	00 18
//DF36	02	Relay Resistance Accuracy Threshold (100 of micro seconds)	B	R	01 2C
//DF37	01	Relay Resistance Transmission Time Mismatch Threshold (%)	B	R	32

4.10.2 Visa payWave Entry Point Table

Table 4.10-3 - Visa payWave Entry Point Table

Tag	Len	Value / Description	Typ	Req	Example
FF35	var	AID Delimiter Container There can be multiple instances of this in sequence. The contents of the first instance are loaded into Entry Point Table Slot 1, the contents of the second are loaded into Entry Point Table Slot 2, etc.	T	R	
/DF0E	03	Kernel ID, Processing Slot, Transaction Type Byte 1 Kernel ID • 0x03 = Visa payWave Byte 2 Processing Slot to Use See the **AID Delimiter Container** parameter in **EMV Processing Configuration File Type** for information about how to identify slots. Byte 3 Transaction Type • 0x00 = Purchase • 0x01 = Cash • 0x02 = Purchase with cashback • 0x03 = Refund	B	R	03 05 00

Tag	Len	Value / Description	Typ	Req	Example
/DF0F	var	Payload Delimiter Container Include only one of these containers inside each **AID Delimiter Container**.	T	R	
//9F35	01	Terminal Type	B	R	21
//9F1A	02	Terminal Country Code	B	R	08 40
//9F33	03	Terminal Capabilities	B	R	00 00 00
//9F40	05	Additional Terminal Capabilities	B	R	00 00 00 00 00
//9F66	04	Terminal Transaction Qualifier	B	R	22 00 40 00
//DF1B	03	Kernel Configuration Byte 1 • Bit 8 = CVN17 fallback to MSD legacy (Not supported) • Bit 7 = Enable MSD and CVN17 (Not supported) • Bit 6 = MSD Formatting Track 2 Data (Not supported) • Bit 5 = MSD Formatting Track 1 Data (Not supported) • Bits 4..3 = Reserved for future use • Bit 2 = Force Online if Online Only terminal and with cash back transaction • Bit 1 = UL2 #23 Errata S.79/5.80 Performed Byte 2 • Bit 8 = DRL Set • Bit 7 = TVR not reset • Bit 6 = TSI not reset • Bits 5..1 = Reserved for future use Byte 3 Reserved for future use	B	R	00 00 06
//DF2D	03	Message Hold Time (100 of ms)	B	R	00 00 0F
//9F09	02	EMV Application Version Number	B	R	00 01
//DF30	01	Bitmap Entry Point • Bit 8 = Status Check Support Flag • Bit 7= Zero Amount Allowed Flag • Bit 6= Reader Contactless Transaction Limit • Bit 5 = Reader Contactless Floor Limit • Bit 4 = Reader CVM Required Limit • Bits 3..1 = Reserved for future use	B	R	F8
//DF32	01	Status Zero Amount Allowed Flag • 0x01 = Option 1, Online Cryptogram Request • 0x02 = Option 2, Not Allowed	B	R	02

Tag	Len	Value / Description	Typ	Req	Example
//9F1B	04	Terminal Floor Limit	B	R	00 00 00 00
//DF23	06	Reader Contactless Floor Limit	B	R	00 00 00 00 20 00
//DF24	06	Reader Contactless Transaction Limit	B	R	00 00 00 00 50 00
//DF26	06	Reader CVM Required Limit	B	R	00 00 00 00 10 00

4.10.3 American Express Expresspay Entry Point Table

Table 4.10-4 - American Express Expresspay Entry Point Table

Tag	Len	Value / Description	Typ	Req	Example
FF35	var	AID Delimiter Container There can be multiple instances of this in sequence. The contents of the first instance are loaded into Entry Point Table Slot 1, the contents of the second are loaded into Entry Point Table Slot 2, etc.	T	R	
/DF0E	03	Kernel ID, Processing Slot, Transaction Type Byte 1 Kernel ID • 0x04 = Expresspay Byte 2 Processing Slot to Use See the **AID Delimiter Container** parameter in **EMV Processing Configuration File Type** for information about how to identify slots. Byte 3 Transaction Type • 0x00 = Purchase • 0x01 = Cash • 0x02 = Purchase with cashback • 0x03 = Refund	B	R	04 04 00
/DF0F	var	Payload Delimiter Container Include only one of these containers inside each **AID Delimiter Container**.	T	R	
//9F09	02	EMV Application Version Number	B	R	00 01
//9F1A	02	Terminal Country Code	B	R	08 40
//9F33	03	Terminal Capabilities	B	R	60 28 00
//9F35	01	Terminal Type	B	R	21
//9F40	05	Additional Terminal Capabilities	B	R	00 00 00 00 00
//9F6D	01	Contactless Reader Capability Bits 8..7 • 00 = Expresspay 1.0 • 01 = Expresspay 2.0 and Expresspay >= 3.x (MSD) • 11 = Expresspay >= 3.x(MSD) Bits 6..1 = 0 (Reserved, not to be configured)	B	R	C0

Tag	Len	Value / Description	Typ	Req	Example
//DF1B	06	Kernel Configuration Byte 1 Seed for UN for MSD mode Byte 2 • Bit 8 = DRL Set • Bit 7 = EMVCo Entry Point • Bit 6 = Use TAC and IAC • Bit 5 = Not EMV Consistency Check • Bit 4 = Patch ODA Applied • Bit 3 = Consistency Check as per EMV • Bit 2 = Delay authorization • Bit 1 = Unable to go online support Byte 3 Hold Time for 'See Phone' cases Byte 4 No holding time on Approve and Decline Byte 5 • Bit 8 = XP 3.0 activated • Bit 7 = Start D managed in EMV Mode • Bit 6 = Disable Discretionary Data • Bit 5 = Display delayed message • Bit 4 = Check CDOL2 Presence • Bit 3 = CVM Mobile Supported • Bit 2 = Terminal exempt of No CVM checks • Bit 1 = Transit Terminal Byte 6 • Bit 8 = Display PROCESSING Message • Bit 7 = Only EMV Mode • Bits 6..4 = Reserved for future use • Bits 3..1 = C-4 Kernel Version	B	R	31 01 00 00 00 00
//DF27	01	Timeout, Field off request (100 of ms)	B	R	20
//DF2D	03	Message Hold Time (100 of ms)	B	R	00 00 0F
//DF30	01	Bitmap Entry Point • Bit 8 = Status Check Support Flag • Bit 7= Zero Amount Allowed Flag • Bit 6= Reader Contactless Transaction Limit • Bit 5 = Reader Contactless Floor Limit • Bit 4 = Reader CVM Required Limit • Bits 3..1 = Reserved for future use	B	R	F8

Tag	Len	Value / Description	Typ	Req	Example
//DF32	01	Status Zero Amount Allowed • 0x01 = Option 1, Online Cryptogram Request • 0x02 = Option 2, Not Allowed	B	R	01
//DF20	05	Terminal Action Code - Default	B	R	00 00 00 00 00
//DF21	05	Terminal Action Code - Denial	B	R	00 00 00 00 00
//DF22	05	Terminal Action Code - Online	B	R	00 00 00 00 00
//DF23	06	Reader Contactless Floor Limit	B	R	00 00 00 00 20 00
//DF24	06	Reader Contactless Transaction Limit	B	R	00 00 00 01 00 00
//DF26	06	Reader CVM Required Limit	B	R	00 00 00 01 00 00

4.10.4 Discover D-PAS Entry Point Table

Table 4.10-5 - Discover D-PAS Entry Point Table

Tag	Len	Value / Description	Typ	Req	Example
FF35	var	AID Delimiter Container There can be multiple instances of this in sequence. The contents of the first instance are loaded into Entry Point Table Slot 1, the contents of the second are loaded into Entry Point Table Slot 2, etc.	T	R	
/DF0E	03	Kernel ID, Processing Slot, Transaction Type Byte 1 Kernel ID • 0x06 = Discover D-PAS Byte 2 Processing Slot to Use See the **AID Delimiter Container** parameter in **EMV Processing Configuration File Type** for information about how to identify slots. Byte 3 Transaction Type • 0x00 = Purchase • 0x01 = Cash • 0x02 = Purchase with cashback • 0x03 = Refund	B	R	06 06 00
/DF0F	var	Payload Delimiter Container Include only one of these containers inside each **AID Delimiter Container**.	T	R	
//9F09	02	EMV Application Version Number	B	R	00 01

Tag	Len	Value / Description	Typ	Req	Example
//9F1A	02	Terminal Country Code	B	R	08 40
//9F33	03	Terminal Capabilities	B	R	00 00 00
//9F35	01	Terminal Type	B	R	21
//9F66	04	Terminal Transaction Qualifier	B	R	B6 00 C0 00
//DF1B	01	Kernel Configuration • Bit 8 = Exception List Supported • Bit 7 = Check PID Limit • Bit 6 = Extended Selection Support • Bit 5 = RFU • Bit 4 = Kernel Supports Do Not Fix Track • Bit 3 = Online PIN allowed after going online • Bits 2..1 = Reserved for future use	B	R	60
//DF1B	02	Kernel Configuration (Common Kernel Only) Byte 1 • Bit 8 = Exception List Supported • Bit 7 = Check PID Limit • Bit 6 = Extended Selection Support • Bit 5 = RFU • Bit 4 = Kernel Supports Do Not Fix Track • Bit 3 = Online PIN allowed after going online • Bits 2..1 = Reserved for future use Byte 2 • Bit 8 = RFU • Bit 7 = RFU • Bit 6 = RFU • Bit 5 = Deferred Authorization	B	R	60 00
//DF30	02	Bitmap Entry Point • Bit 8 = Status Check Support Flag • Bit 7= Zero Amount Allowed Flag • Bit 6= Reader Contactless Transaction Limit • Bit 5 = Reader Contactless Floor Limit • Bit 4 = Reader CVM Required Limit • Bits 3..1 = Reserved for future use	B	R	F8
//DF32	01	Status Zero Amount Allowed Flag • 0x01 = Option 1, Online Cryptogram Request • 0x02 = Option 2, Not Allowed	B	R	01
//9F1B	06	Terminal Floor Limit	B	R	00 00 00 00
//DF23	06	Reader Contactless Floor Limit	B	R	00 00 00 01 50 00

Tag	Len	Value / Description	Typ	Req	Example
//DF24	06	Reader Contactless Transaction Limit	B	R	00 00 00 03 00 00
//DF26	06	Reader CVM Required Limit	B	R	00 00 00 00 20 00

4.10.5 China Unionpay Entry Point Table (Common Kernel Only)

Table 4.10-6 – China Unionpay Entry Point Table

Tag	Len	Value / Description	Typ	Req	Example
FF35	var	AID Delimiter Container There can be multiple instances of this in sequence. The contents of the first instance are loaded into Entry Point Table Slot 1, the contents of the second are loaded into Entry Point Table Slot 2, etc.	T	R	
/DF0E	03	Kernel ID, Processing Slot, Transaction Type Byte 1 Kernel ID • 0x07 = China Unionpay Byte 2 Processing Slot to Use See the **AID Delimiter Container** parameter in **EMV Processing Configuration File Type** for information about how to identify slots. Byte 3 Transaction Type • 0x00 = Purchase • 0x01 = Cash • 0x02 = Purchase with cashback • 0x03 = Refund	B	R	07 05 00
/DF0F	var	Payload Delimiter Container Include only one of these containers inside each **AID Delimiter Container**.	T	R	
//9F09	02	EMV Application Version Number	B	R	00 30
//9F1A	02	Terminal Country Code	B	R	01 56
//9F33	03	Terminal Capabilities	B	R	60 08 00
//9F35	01	Terminal Type	B	R	21
//9F66	04	Terminal Transaction Qualifier	B	R	36 00 00 80

Tag	Len	Value / Description	Typ	Req	Example
//DF1B	02	Kernel Configuration Byte1: • → b8..b2: RFU • → b1: Display PROCESSING Message Byte2: • → b8: C7 Outcome • → b7: TVR Not Reset • → b6: 9F27 Always Raised • → b5: Not Forced Online • → b4: Standard UICS DT/CT Application Flow Supported • → b3: Always DD • → b2: Not Decline On Transit Debit AID ONLINE PIN Case • → b1: Transit Terminal	B	R	00 00
//DF20	05	Terminal Action Code - Default	B	R	00 00 00 00 00
//DF21	05	Terminal Action Code - Denial	B	R	00 00 00 00 00
//DF22	05	Terminal Action Code - Online	B	R	00 00 00 00 00
//DF23	06	Reader Contactless Floor Limit	B	R	00 00 00 01 50 00
//DF24	06	Reader Contactless Transaction Limit	B	R	00 00 00 03 00 00
//DF26	06	Reader CVM Required Limit	B	R	00 00 00 01 20 00
//DF30	01	Bitmap Entry Point • Bit 8 = Status Check Support Flag • Bit 7= Zero Amount Allowed Flag • Bit 6= Reader Contactless Transaction Limit • Bit 5 = Reader Contactless Floor Limit • Bit 4 = Reader CVM Required Limit Bits 3..1 = Reserved for future use	B	R	78
9F1B	04	Terminal Floor Limit	B	R	00 00 3a 98

4.10.6 JCB Entry Point Table (Common Kernel Only)

Table 4.10-7 - JCB Entry Point Table

Tag	Len	Value / Description	Typ	Req	Example
FF35	var	AID Delimiter Container There can be multiple instances of this in sequence. The contents of the first instance are loaded into Entry Point Table Slot 1, the contents of the second are loaded into Entry Point Table Slot 2, etc.	T	R	
/DF0E	03	Kernel ID, Processing Slot, Transaction Type Byte 1 Kernel ID • 0x05 = JCB Byte 2 Processing Slot to Use See the **AID Delimiter Container** parameter in **EMV Processing Configuration File Type** for information about how to identify slots. Byte 3 Transaction Type • 0x00 = Purchase • 0x01 = Cash • 0x02 = Purchase with cashback • 0x03 = Refund	B	R	05 06 00
/DF0F	var	Payload Delimiter Container Include only one of these containers inside each **AID Delimiter Container**.	T	R	
//9F01	06	Acquirer Identifier	B	R	00 00 00 00 00 01
//9F15	02	Merchant Category Code	B	R	70 32
//9F09	02	EMV Application Version Number	B	R	00 01
//9F1A	02	Terminal Country Code	B	R	03 92
//9F33	03	Terminal Capability	B	R	60 68 08
//9F35	01	Terminal Type	B	R	21
//9F4E	var	Merchant Name and Location	B	R	5858204D 455243484 14E542059 59204C4F 434154494 F4E

Tag	Len	Value / Description	Typ	Req	Example
//DF1B	03	Kernel Configuration Byte1 (reflects Combination Options): • → b8: RFU • → b7: Status Check Supported • → b6: ODA Supported • → b5: Exception File Supported • → b4: Random Transaction Selection Supported • → b3: MagStripe Mode Supported (see here) • → b2: EMV Mode Supported (see here) • → b1: Legacy Mode Supported (see here) Byte2 (reflects Combination Options): • → b8..b1: RFU • Byte3: • → b8: On Device CVM • → b7: Transit Reader Supported • → b6: Issuer Update Supported • → b5..b3: RFU • → b2: Do Not Display PROCESSING message • → b1: TVR with Online Pin for Legacy	B	R	7B 00 80
//DF20	05	Terminal Action Code - Default	B	R	90 40 20 80 20
//DF21	05	Terminal Action Code - Denial	B	R	04 10 20 20 20
//DF22	05	Terminal Action Code - Online	B	R	90 60 20 90 20
//DF23	06	Reader Contactless Floor Limit	B	R	00 00 00 01 50 00
//DF24	06	Reader Contactless Transaction Limit	B	R	00 00 00 03 00 00
//DF25	06	On Device CVM Contactless Transaction Limit	B	R	00 00 00 02 50 00
//DF26	06	Reader CVM Required Limit	B	R	00 00 00 01 20 00
//9F1B	04	Terminal Floor Limit	B	R	00 00 3a 98

4.10.7 Interac Flash Entry Point Table (Common Kernel Only)

Table 4.10-8 – Interac Flash Entry Point Table

Tag	Len	Value / Description	Typ	Req	Example
FF35	var	AID Delimiter Container There can be multiple instances of this in sequence. The contents of the first instance are loaded into Entry Point Table Slot 1, the contents of the second are loaded into Entry Point Table Slot 2, etc.	T	R	
/DF0E	03	Kernel ID, Processing Slot, Transaction Type Byte 1 Kernel ID • 0x41 = Interac Flash Byte 2 Processing Slot to Use See the **AID Delimiter Container** parameter in **EMV Processing Configuration File Type** for information about how to identify slots. Byte 3 Transaction Type • 0x00 = Purchase • 0x01 = Cash • 0x02 = Purchase with cashback • 0x03 = Refund	B	R	41 05 00
/DF0F	var	Payload Delimiter Container Include only one of these containers inside each **AID Delimiter Container**.	T	R	
//9F09	02	EMV Application Version Number	B	R	00 02
//9F1A	02	Terminal Country Code	B	R	01 24
//9F33	03	Terminal Capabilities	B	R	60 68 08
//9F35	01	Terminal Type	B	R	21
//9F40	05	Additional Terminal Capabilities	B	R	E0 00 E0 F0 01
//9F58	01	Merchant Type Indicator	B	R	03
//9F5D	06	Receipt Limit	B	R	00 00 00 00 50 00

Tag	Len	Value / Description	Typ	Req	Example
//9F5E	02	Terminal Option Status Byte1: • → b8: Use Interface If Different Currency • → b7: Use Interface If Different Country Code • → b6: Use Interface If Domestic Transaction With Different Currency • → b5..b1: RFU Byte2: • → b8..b1: RFU	B	R	E0 00
//9F5F	06	Reader Contactless Floor Limit	B	R	00 00 00 01 00 00
//DF1B	02	Kernel Configuration Byte1: • Max Retry Byte2: • → b8: Interac Contact • → b7: Interac on Other Terminal • → b6: Mobile NFC • → b5: Contactless Card • → b4: Legacy Floor Limit • → b3: Always DD • → b2: Terminal v1.4 • → b1: Processing Message Disabled	B	R	02 34
//DF20	05	Terminal Action Code - Default	B	R	00 00 00 00 00
//DF21	05	Terminal Action Code - Denial	B	R	00 00 00 00 00
//DF22	05	Terminal Action Code - Online	B	R	00 00 00 00 00
//9F1B	04	Terminal Floor Limit	B	R	00 00 1F 40

4.11 EMV Configuration CA Public Keys File Type

The host can load this file type to control the behavior of the device's EMV contact and contactless kernels when the device should support Offline Data Authentication (ODA). Populate all values from information provided by each payment brand that should be supported by the device. The host can load this file using **Command 0xD812 - Start Send File to Device (Unsecured)**.

MagTek provides tools that allow these settings to be loaded using a Microsoft Excel spreadsheet in xlsx format for more convenient authoring, review, and change tracking. For a reference sample spreadsheet,

contact MagTek Support Services. The MagTek tools expect the spreadsheet to be formatted format as shown in **Table 4.11-1**. Each CA Key to be supported is defined in a tab of the Excel file.

Table 4.11-1 - EMV Configuration CA Keys File Type

Tag	Len	Value / Description	Typ	Req	Example
DFDF79	05	Registered Application ID (RID)	B	R	A0 00 00 00 04
DFDF7A	01	CA Public Key Index	B	R	05
DFDF7B	var	CA Public key Modulus	B	R	B8 04 8A … D5 97
DFDF7C	01 or 03	CA Public Key Exponent	B	R	03
DFDF7D	14	CA Public Key Checksum	B	R	EB FA 0D 5D 06 D8 CE 70 2D A3 EA E8 90 70 1D 45 E2 74 C8 45

The MagTek tool converts the spreadsheet data into the format shown in **Table 4.11-2**.

Table 4.11-2 - CA Keys Raw Format

CA Keys Raw Format
RID (5 Bytes) As defined by the payment brand.
Index (1 Byte) As defined by the payment brand.
Exponent Length (1 Byte) • 0x01 • 0x03
Key Length (1 Byte), Max of 248 bytes per EMVCo specifications
Exponent (1 or 3 Bytes) • 0x03 • 0x010001
Modulus As defined by the payment brand.
Additional CA Keys, repeating from RID through Modulus, as needed.
SHA-1 hash of all data in the file

4.12 EMV American Express DRL Configuration File Type (Not Supported on Expresspay 4.x)

The host can load this file type to control the behavior of the device's American Express contactless kernel when the card sends tag 9F70 and one or more DRL is defined.

The host can load it using **Command 0xD812 - Start Send File to Device (Unsecured)**. See the ***Expresspay 4.0.2*** specification for functional details.

MagTek provides tools that allow these settings to be loaded using a Microsoft Excel spreadsheet in xlsx format for more convenient authoring, review, and change tracking. For a reference sample spreadsheet, contact MagTek Support Services. The MagTek tools expect the spreadsheet to be formatted as shown in **Table 4.12-1**. Each DRL to be supported is defined in a tab of the Excel file.

Table 4.12-1 - EMV Configuration American Express DRL Set File Type

Tag	Len	Value / Description	Typ	Req	Example
DF23	06	Reader Contactless Floor Limit	B	R	00 00 00 00 15 00
DF24	06	Reader Contactless Transaction Limit	B	R	00 00 00 00 05 00
DF26	06	Reader CVM Required Limit	B	R	00 00 00 00 10 00

The MagTek tool converts the spreadsheet data into the raw format shown in **Table 4.12-2**:

Table 4.12-2 -Raw EMV Configuration American Express DRL Set File Type

Tag	Len	Value / Description	Typ	Req	Example
File Type Version One byte indicating the version of the file type format being used. • 0xAA					
SHA-1 Hash 20 byte hash of all values that follow					
FF37	var	DRL Configuration Container	T	R	
/FF36	var	DRL Set Container	T	R	
//DF23	06	Reader Contactless Floor Limit	B	R	00 00 00 00 15 00
//DF24	06	Reader Contactless Transaction Limit	B	R	00 00 00 00 05 00
//DF26	06	Reader CVM Required Limit	B	R	00 00 00 00 10 00
Additional instances of DRL Set Container as needed					

4.13 Security Operation Type

This non-TLV data structure consists of four or five bytes that describes a security operation, including the algorithms and methods to be used in that operation.

Table 4.13-1 - Security Operation Type

Offset	Description	Typ	Req	Default
0	Operation Type • 0x01 = Key Agreement • 0x02 = Command Authorization Using Signature • 0x03 = Command Authorization Using MAC • 0x05 = Data Authentication Using MAC • 0x07 = Data Encryption • 0x10 = Data Signature	B	R	
1	Operation Algorithm If **Operation Type** is Key Agreement type: • 0x01 = ECDHE If **Operation Type** is a Signature type: • 0x01 = ECDSA (indeterministic) If **Operation Type** is a MAC type: • 0x01 = HMAC • 0x02 = CBC-MAC • 0x03 = CMAC If **Operation Type** is an Encryption type: • 0x01 = DEA • 0x02 = 2TDEA • 0x03 = 3TDEA • 0x04 = AES-128 • 0x05 = AES-192 • 0x06 = AES-256	B	R	

Offset	Description	Typ	Req	Default
2	Operation Curve/Mode/Hash/Cipher If **Operation Type** is a Key Agreement type, this specifies the Curve: • 0x01 = P192 • 0x02 = P224 • 0x03 = P256 • 0x04 = P384 • 0x05 = P521 If **Operation Type** is a Signature type, this specifies the Hash: • 0x01 = MD5 • 0x02 = SHA-1 • 0x03 = SHA-224 • 0x04 = SHA-256 • 0x05 = SHA-384 • 0x06 = SHA-512 • 0x07 = SHA-512/224 • 0x08 = SHA-512/256 • 0x09 = SHA3-224 • 0x0A = SHA3-256 • 0x0B = SHA3-384 • 0x0C = SHA3-512 If **Operation Type** is a MAC type, this specifies the Encryption Algorithm: • 0x01 = DEA • 0x02 = 2TDEA • 0x03 = 3TDEA • 0x04 = AES-128 • 0x05 = AES-192 • 0x06 = AES-256 If **Operation Type** is an Encryption type, this specifies the Mode: • 0x01 = ECB (Block) • 0x02 = CBC (Block) • 0x03 = CFB (Stream) • 0x04 = OFB (Stream) • 0x05 = CTR (Stream)	B	R	

Offset	Description	Typ	Req	Default
3	KDF/Curve/Padding If **Operation Type** is a Key Agreement type, this specifies the KDF: • 0x01 = SP800-56A / X9.63 If **Operation Type** is a Signature type, this specifies the Curve: • 0x01 = P192 • 0x02 = P224 • 0x03 = P256 • 0x04 = P384 • 0x05 = P521 If **Operation Type** is a MAC type, this specifies the Padding: • 0x00 = None (for streaming modes) • 0x01 = Zeros (ISO 9797 Padding Method 1) • 0x02 = One and zeros (ISO 9797 Method 2) • 0x03 = Length + zeros (ISO 9797 Method 3) • 0x10 = PKCS7 (pad # = pad length) • 0x11 = X9.23 (random + pad length) • 0x20 = Random (when length is known) If **Operation Type** is an Encryption type, this specifies the Padding: • 0x00 = None (for streaming modes) • 0x01 = Zeros (ISO 9797 Padding Method 1) • 0x02 = One and zeros (ISO 9797 Method 2) • 0x03 = Length + zeros (ISO 9797 Method 3) • 0x10 = PKCS7 (pad # = pad length) • 0x11 = X9.23 (random + pad length) • 0x20 = Random (when length is known)	B	R	
4	MAC Block Size If **Operation Type** is a MAC type, this specifies the data to be MACed must be padded to a multiple of this many bytes. For all other Operation Types, do not include this byte.	B	O	

4.14 Security Parameters Type

Table 4.14-1 - Security Parameters Type

Tag	Len	Value / Description	Typ	Req	Default
81	var	Operation This contains an instance of a **Security Operation Type** structure specifying the operation to be performed.	B	R	

Tag	Len	Value / Description	Typ	Req	Default
84	var	Data Reserved for future use. Do not include this parameter. It is reserved for Initialization Vector or nonce, if needed.	B	O	
85	var	Extra Data Item Reserved for future use. Do not include.	B	O	
A8	var	Key Information This specifies the key used in the operation. Populate with a **Key Information Type** TLV data object. For ECDSA operations, do not include this parameter.	T	O	
A9	var	Second Key Information (Reserved, do not include) This specifies a second key used in the operation. If needed, populate with a **Key Information Type** TLV data object.	T	O	

4.15 Key Information Type

Table 4.15-1 - Key Information Type

Tag	Len	Value / Description	Typ	Req	Default
81	02	Key Slot ID Identifies the key being used for operation. See **Table 4.18-6 - Key Slot IDs**.	B	R	
82	var	Key Label The label that indicates the key type. For example, **DEVTK**.	AN	O	
86	var	Key Derivation Details Use Key Serial Number (KSN) in requests, Key Derivation Information in responses.	B	O	
88	var	Additional Information Reserved for future use. Do not include.	B	O	

4.16 NFC UID Type (EMV Contactless Only)

Table 4.16-1 – NFC UID Type

Tag	Len	Value / Description	Typ	Req	Default
DF79	var	NFC UID	B	R	

4.17 GPO Response Type (EMV Contactless Only)

Table 4.17-1 – NFC UID Type

Tag	Len	Value / Description	Typ	Req	Default
81	var	GPO Response	B	R	

4.18 TR-31 Key Block Type

A ***TR-31*** (***X9.143***) key block consists of three parts:

The **Key Block Header** (KBH) which contains attribute information about the key and the key block and is not encrypted. It is always treated as ASCII.

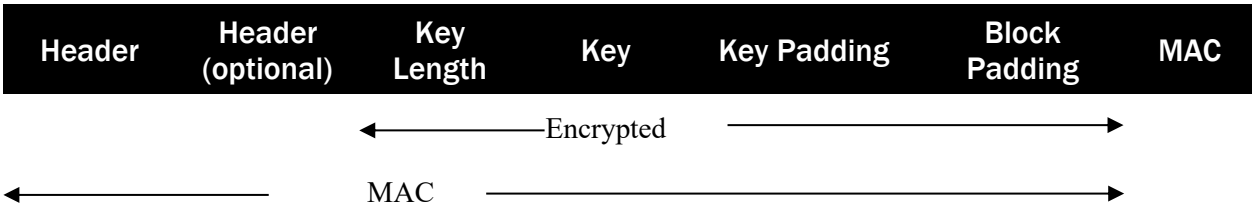
- The first section is 16 bytes with a fixed format defined below.
- The second section is optional within the standard, but required for current products.

The **Confidential Data**, which is encrypted and always binary.

- Two bytes indicating the key length (in bits, AES-128 is 128 bits, so length will be 0080).
- The secret key and/or sensitive data.
- Padding as required (random bytes 0x00 to 0xFF).

The **MAC**, which is of varying length as follows:

- 64 bits if the TDEA key derivation method is used (typically not used for this device).
- 128 bits if the AES key derivation method is used.



Symmetric keys are padded with **Block Padding** to the maximum length for the algorithm, 192 bits for TDEA or 256 bits for AES, to hide the true length of short keys.

The data to be encrypted and the MAC are always binary for calculation purposes. The encrypted data and the MAC are converted to ASCII hex as the last step.

Date and time strings specified within the TR-31 block are represented according to the rules described in ***ISO 8601*** and ***TR-31***. Year is 4 digits. Time uses UTC 24 hour clock. Some functions like 'toISOString()' will produce a string of format **yyyy-mm-ddThh:mm:ss.fffZ** where fff is a decimal fraction of a second, Z is UTC time zone. The device ignores 'Z' and '.fff' if they are present. Seconds ':ss' are optional. Date, hours, and minutes are required. For example, March 23, 2020 4:19PM is encoded as **2020-03-23T16:19** at minimum, but could also be **2020-03-23T16:19:00.000Z**.

Table 4.18-1 - TR-31 Block Fixed Header

Offset	Name	Fixed Value	Variable
0	Key Block V ID	'D'	
1..4	Key Block Length		Calculated (in decimal, e.g. 138 bytes shown as '0138'
5..6	Usage		Look up the desired **Key Type** in **Table 4.18-2 below** and select this value from the **Usage** column.
7	Algorithm		Look up the desired **Key Type** in **Table 4.18-2 below** and select this value from the **Algorithm** column.
8	Mode of Use		Look up the desired **Key Type** in **Table 4.18-2 below** and select this value from the **Mode of Use** column.

Offset	Name	Fixed Value	Variable
9..10	Key Version #	'00'	Always '00'
11	Exportability	'N'	Always no export allowed
12..13	# option blocks		Calculated
14..15	Reserved	'00'	

Table 4.18-2 - TR-31 Key Type Table - Usage/Algorithm/Mode

Key Type	Usage	Algorithm AES/TDEA	Mode of Use (Both, To, From)
Transport (KBPK)	'K1'	'A' / 'T'	'D'
Initial DUKPT Key	'B1'	'A' / 'T'	'X'
Fixed MAC (CMAC)	'M6'	'A' / 'T'	('C', 'G', 'V')
Fixed Encrypt	'D0'	'A' / 'T'	('B', 'E', 'D')

Table 4.18-3 - TR-31 Optional Blocks

ID	Purpose
'IK'	DUKPT KSID
'KS'	Key Set Identifier (e.g. data used by host to find and/or derive this key).
'KC'	Key Check Value (KCV) (Legacy or CMAC)
'PB'	Padding Field
'TS'	Current Time Stamp (optional) see description in previous section.
'KP'	KCV of KBPK that created this Key Block (optional-preferred)
'21'	MagTek Additional Key Info From **Table 4.18-4 - MagTek Custom TR-31 Small Optional Block**

Table 4.18-4 - MagTek Custom TR-31 Small Optional Block

Offset	Name	Value	Variable
0..1	Block ID	'21'	MagTek Added Key Info Block
2..3	Block Length	var	ASCII Hex (Length 01-FF from offset 0)
4..7	Owner Tag	'MGTK'	Avoid collision with others using Block ID '21'
8..9	Data Tag	'10'	Field ID
10..11	Data Len	'01'	Field Length (ASCII Hex 00-FF)
12	Data	'T','P', or '0'	Field Data for Key Environment • T = Test • P = Production • 0 = Erase Key
13…	Added elements		More Fields (Tags, Lengths, and Data)

Table 4.18-5 - MagTek Custom Key Data Fields

Field ID	Length	Purpose
'10'	'01'	Key Environment • T = Test • P = Production • 0 = Erase Key
'11'	'04'	Key Slot ID See **Table 4.18-6 - Key Slot ID**.
'12'	'04'	Key Slot ID of Transport Key
'20'	--	Reserved
'21'	'04'	DUKPT Data Type Restriction Bitmask This is for Transport Keys and DUKPT keys. Default to 0.
'31'	'07'	Device Serial Number
'32'	'10'	Challenge Token 10h = 16 characters
'33'	'10' ..'18'	Expiration Date/Time This is in UTC format, use short form if possible. Reserved.

Table 4.18-6 - Key Slot IDs

ID	Label	Description	Load Transport Key	TR31-F
10xx		Transport Keys (KBPK)		
1000	TMPTK	Temporary KBPK	Key agreement process from **Command 0xF017 - Establish Ephemeral KBPK**	N/A
1001	MTK	Master Transport Key	TMPTK	K1AD
1002	DEVTK	Device Master Transport Key	MTK	K1AD
1003	FINTK	Financial Master Transport Key	MTK	K1AD
1081	MKIFTK	MagTek KIF Financial Transport Keys	FINTK	K1AD
1101	FREQMK	Factory Request MAC Key	PRODTK	M6AV
1102	MREQMK	Manufacturer Device Request MAC Key	MFGTK	M6AV
1111	MFRQMK	Manufacturer Financial Request MAC (Configuration) Key	MKIFTK	M6AV
0x2000 to 0x201F	DKPTM0 to DKPTM1F	DUKPT Initial Keys,	MKIFTK	B1TX

4.18.1 DUKPT Key Mapping

4.18.1.1 Terms and Definitions

DUKPT – Derived Unique Key Per Transaction

OID – Object Identifier
SRED - Secure Reading and Exchange of Data

There are 7 new OIDs defined for these 7 SRED Data IDs.
Each OID value contains a two-byte DUKPT slot ID and a one-byte transformation ID.

Table 4.18-7 - SRED Data IDs and OIDs

SRED Data ID	OID	OID Size
0: Not assigned	N/A	N/A
1: PIN-TDES (supported on PED devices Only)	0x010102040101	3
2: Account Data	0x010102040102	3
3: MAC	0x010102040103	3
4: Magneprint (supported on devices with MSR Only)	0x010102040104	3
5: MagTek Token	0x010102040105	3
6: User Data 1	0x010102040106	3
7: PIN-AES (supported on PED devices Only)	0x010102040107	3

DUKPT Slot IDs

The existing TR31 Module supports 32 MagTek DUKPT Slot IDs, from 0x2000 to 0x201F.
The Key Injection Software Tool shall inject DUKPT keys through these DUKPT Slot IDs.

Transformation IDs

This is the list of DUKPT transformations defined in both the Legacy and AES specifications.

4.18.2 Restrictions of a DUKPT Slot ID

Table 4.18-8 - Transformation IDs for DUKPT Legacy and AES

Transformation ID #	Usage Name	Type	Data for calculation
0	Reserved		
1	PIN Encryption	Legacy	00 00 00 00 00 00 00 FF
2	MAC Generate/Verify	Legacy	00 00 00 00 00 00 FF 00
3	MAC Verify	Legacy	00 00 00 00 FF 00 00 00
4	Data Enc/Decryption	Legacy	00 00 00 00 00 FF 00 00
5	Data Encryption	Legacy	00 00 00 FF 00 00 00 00
6	Reserved		
7	PIN Encryption	AES	0x1000
8	MAC Generate	AES	0x2000
9	MAC Verify	AES	0x2001
A	MAC Generate/Verify	AES	0x2002
B	Data Encryption	AES	0x3000
C	Data Decryption	AES	0x3001
D	Data Enc/Decryption	AES	0x3002

Table 4.18-9 - The definition of Restriction bit map

Bit #	5	4	3	2	1	0
Data Type	User Data (RFU)	Token (RFU)	Magneprint	MAC	Account Data	PIN

During TR31 Key Injection, each DUKPT Slot ID contains a parameter indicates the purpose of a Key Set.

Example 1: The restriction value is 0x3F
This Key Set can be used for all purposes.

Example 2: The restriction value is 0x3E
This Key Set can be used for all purposes, except PIN Encryption.

Example 3: The restriction value is 0x01
This Key Set can be used for PIN Encryption only.

4.18.3 The Rules of Key Mapping

SRED Data ID map configuration values (Slot ID and Transformation ID) must be checked and rejected if they don't meet the following conditions.

a). The DUKPT Slot ID must be loaded. (**Table 4.18-10**)
b). The loaded DUKPT Slot ID must allows this type of SRED Data ID. (Table 4.18-9)
c). The transformation must be allowed by **Table 4.18-11**.

4.18.4 The settings of DUKPT Slot IDs injected through TR31

Here is the list of parameters of 4 DUKPT Slot IDs based on the existing Key Injection Tool.

Table 4.18-10 - Settings of Injected DUKPT Slot IDs

DUKPT Slot ID	Key Type	Restrictions
DKPTM0-2000	TDES	0x3E
DKPTM2-2002	AES-128	0x3F
DKPTM3-2003	AES-256	0x3F
DKPTM7-2007	TDES	0x3F

4.18.5 The Allowed Key Mapping Table

Table 4.18-11 - Allowed Key Mapping Table

SRED Data ID	Data Type (Working Key Purpose)	Allowed Legacy DUKPT Transforms	Allowed AES DUKPT Transforms
0	Not assigned	-	-
1	PIN-TDES (supported on PED devices Only)	01	Not allowed
2	Account Data	01, 04, 05	0B, 0D
3	Transaction MAC	02	08, 0A
4	MagnePrint (supported on devices with MSR Only)	01, 04, 05	0B, 0D
5	MagTek Token (RFU)	RFU	RFU
6	User Data #1 (RFU)	RFU	RFU
7	PIN-AES (supported on PED devices Only)	Not allowed	07
…	RFU	-	-

Note: If SRED Data ID 2 and 4 are mapped to the same Key Set, then they must have the same Transformation ID. If the Transformation ID of the latest key mapping request is different, then the original OID setting of the other SRED Data ID will be forced to match the latest OID setting. For example, SRED Data ID 2 has been mapped to 0x2007 0x04, user wants to map SRED Data ID 4 to 0x2007 0x05, then the OID setting of SRED Data ID 2 will be forced to 0x2007 0x05.

4.18.5.1 Examples of Key Mapping

The following OID Values indicate that:

a) 200701: Map PIN-TDES to DKPTM7-2007 PIN Encryption Variant.
b) 20020B: Map Account Data to DKPTM2-2002 Data Encryption Usage.
c) 200702: Map MAC to DKPTM7-2007 MAC Generate/Verify Variant.
d) 20030B: Map MangePrint to DKPTM3-2003 Data Encryption Usage.
e) 000004: MagTek Token is RFU, 0000 ID does not exist (this is default value).
f) 000004: User Data is RFU, 0000 ID does not exist (this is default value).
g) 200207: Map PIN-AES to DKPTM2-2002 PIN Encryption Usage.

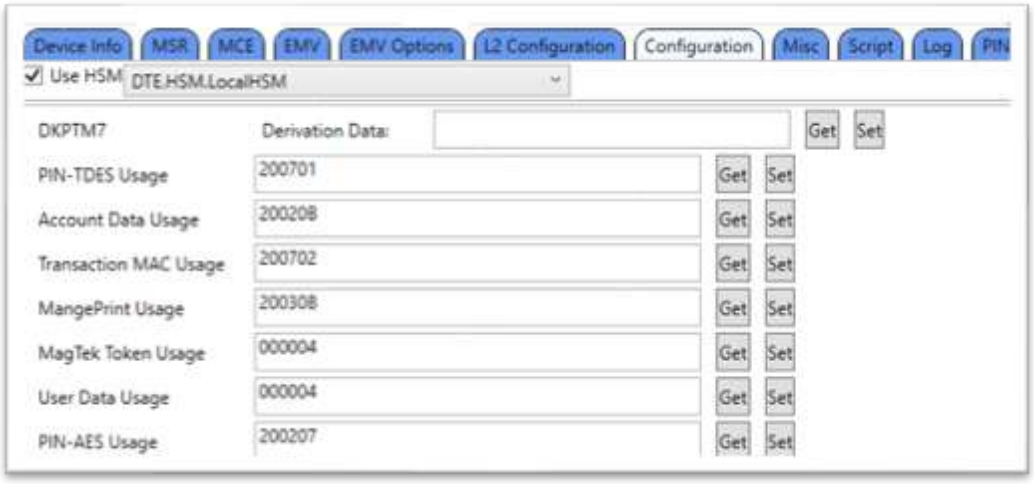


Figure 4.18-1 - Configuration Usage Values

4.19 Common File Structure

Some files types conform to the common file structure format that follows.

The following is a list of file types that conform to this format.

1) EMV file types
2) **Certificate File Types**
3) **Certificate Signing Request (CSR) File Types**

Table 4.19-1 - Common File Structure

Tag	Len	Value / Description	Typ	Req	Default
MGTKAP10 = Start Of File Marker					
C1	4	File Type See Table 6.6-1	B	R	N/A
CE	var	File Payload See the "File Type" subsections of section **4 Data Types and Shared TLV Data Objects**.	B	R	N/A

4.20 Certificate File Types

These file types are listed in Table 6.6-1.

These file types conform to the **Common File Structure** format.

The File Payload of these files contain a certificate in PEM format.

Certificates must be loaded into the device in the order of trust starting with the root CA, next intermediate CA(s) and ending with the leaf (server or client for example) so that the device can verify the signature of each certificate. If the certificates are not loaded in order, they will be rejected.

When loading a server cert, the associated key pair must already exist in the device as the CSR keys or as the existing server or client cert and will be used to verify that the public key contained in the certificate is

correct. If the keys don't match the certificate will be rejected. If the certificate is associated with the CSR keys, the CSR keys will be associated with the certificate and will no longer be available for other certificates until re-generated.

Initial server or client certificates can not be loaded until the respective CSR keys and CSR is generated.

4.21 Certificate Signing Request (CSR) File Types

These file types are listed in Table 6.6-1.

These file types conform to the **Common File Structure** format.

The File Payload of these files contain a certificate signing request in PEM format.

The device will erase the CSR file from volatile memory after the host fetches it with **Command 0xD821 - Start Get File from Device**. After fetching the CSR, the keys used to generate the CSR will still exist in non-volatile memory associated with a CSR and can be used to generate a new CSR if needed.

4.22 Card Emulation

4.22.1 Card Emulation Overview

Card emulation enables a DynaFlex/DynaProx device to simulate a Type 4 smart card. The emulated card has the following characteristics:

- Compliance: Card emulation conforms to ISO/IEC 14443 Type-A and NFC Forum Type 4 standards.
- Passive Operation: The emulated card functions in passive mode. The phone is responsible for generating the magnetic field required to activate the simulated card.
- Read-Only: The emulated card is read-only and does not support writing.
- Supported Data Type: Supports the URI (URL) data type.

4.22.2 Device Compatibility

- iPhone Support: NFC card reading was introduced on Apple iPhones starting with the iPhone 7.
- Android Support: Android added NFC support in 2012; however, compatibility depends on the specific phone model and hardware capabilities. Most Android phones released since 2016 support NFC. Verify with the phone's manufacturer to confirm compatibility.
- Google Pay Indicator: If a phone supports Google Pay, it is likely capable of reading NFC tags.

5 Security

5.1 About Message Authentication Codes (MAC)

"MAC" is an abbreviation of Message Authentication Code, which is a string of bytes included in a message that can be used to provide reasonable assurance that the message originated from a trusted source and has not been modified.

5.1.1 MACs for EMV Data

This section describes how to calculate MACs for EMV ARQC and EMV Batch Data.

The key and variant used to calculate the MAC is determined by how the **DUKPT Key Mapping** is mapped, this can be set-up for TDES or AES mode.

The key used to calculate the MAC is normally the same key used to encrypt the encrypted data included as part of the same data structure. For TDES MAC, the key variant is always ***Message Authentication, Request or Both Ways***. For AES MAC, this can be AES-128 or AES-256, the valid usages are: 0x08=MAC Generate, 0x0A=MAC Generate/Verify.

For TDES MAC:
ANSI X9.24-3-2017
The MAC operations follow the CBC procedure described in ISO 16609 Section C.4 using padding method 1 defined in ISO 9797 section 6.1.1.

For AES CMAC:
NIST Special Publication 800-38B
Section 6.2 MAC Generation

The data structure for both EMV ARQC and EMV Batch Data has the following format related to MACing:

```
AAAA /* 2-byte MSB message length excluding padding and CBC-MAC */

F9<len> /* container for MAC structure and generic data */
     DFDF54(MAC KSN)<len><val>
     DFDF55(MAC Encryption Type)<len><val>
     DFDF25(IFD Serial Number)<len><val>
     <Nested TLV data objects specific to the message>

<Padding to force the 2-byte MSB message length plus F9 plus padding
to be a multiple of 8 bytes>

<Four byte CBC-MAC of all data starting with the 2-byte MSB message
length and ending with the last byte of padding (if any)>
```

5.2 About Key Serial Numbers (KSNs)

The device can use two different types of keys: Triple Data Encryption Standard (TDES) DUKPT Keys and AES DUKPT Keys. The Key Serial Number (KSN) format is slightly different depending on which

type of key is injected into the key slot that is being used by the operation being performed or the data being passed.

When the device and host are using TDES keys, the Key Serial Number (KSN) is an 80-bit value. The rightmost 21 bits are the current value of the encryption counter associated with that key. The leftmost 59 bits are the **Initial KSN** for that key, which is specified during key injection and 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 a 3 bit **Initial Key Load Counter** the injecting host increments each time the same key is re-loaded into the device. 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."

5.3 About Encryption and Decryption

Some data exchanged between the device and the host is encrypted. This includes parts of the **EMV ARQC Type** and the **EMV Batch Data Type**. To decrypt this data, the host must first determine what key to use, then decrypt the data. The following sections explain each of those steps in detail.

5.4 How to Determine the Key

When the device and the host are using TDES DUKPT key and the device is encrypting data, 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:
   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. The encrypted portions of EMV ARQC and EMV Batch Data both use the *Data Encryption, Request or Both Ways* variant.
4) Use the variant algorithm with the current key to calculate that variant.
5) Decrypt the data according to the steps in section **5.5 How to Decrypt Data**.

5.5 How to Decrypt Data

For EMV ARQC and EMV Batch Data, the device begins by TDES encrypting the first 8 bytes of clear text data. The 8-byte result of this encryption is placed in an encrypted data buffer. The process continues using the TDES 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. For EMV ARQC and EMV Batch Data this information is included as part of the unencrypted data structure.
7) Truncate the end of the decrypted data block to the expected data length, which discards the padding at the end.

5.6 24 Hour Automatic Reset PCI Requirement

Due to Payment Card Industry (PCI) certification requirements, the device must automatically reset at least once every 24 hours. This is required for security purposes, so that the device can reinitialize memory and perform a self-test. Host software developers and potentially end users should be aware of this.

By default, the device will automatically reset 23 hours after it boots up and it will attempt to send a warning notification message to the host 3 minutes before the reset occurs.

The default behavior is adjustable. See the following for more information.

Property 1.2.7.1.1.4 Auto Reset Configuration
Property 1.2.7.1.1.3 Device Reset Will Occur Soon Notification Control
Device Reset Will Occur Soon notification in **Notification 0x1001 - Device Information Update**

One way the host software could handle receiving a device reset will occur notification would be as follows.

1) Cancel any operation that is in process like a transaction if it is unlikely to end before the reset.
2) Optionally send **Command 0x1F01 - Reset Device** instead of waiting for the automatic reset to occur.
3) Re-connect with the device and re-start any operation that was in process.

5.7 Device Lock Feature

The purpose of locking a device is to disable most of the device's functionality for use cases that need this extra layer of security. When a device is locked, it will reject all commands except for a few. This locks the majority of the device's functionality. Use of the device lock feature is optional. It is not required by PCI.

The device lock state can be either unlocked or locked. The device lock state can be retrieved by getting a property.

The device lock state can be changed by setting a secure property or it can be changed with a command that requires knowledge of the device lock passcode.

The device can be configured to always have the device lock state set to locked after a reset or power cycle by setting a property.

If the device has a display and the device is locked, the welcome screen will display “WELCOME” “Device is Locked”. This screen can be customized.

The device lock passcode can be changed by setting a secure property or it can be changed with a command that requires knowledge of the current device lock passcode.

The following commands and properties can be used to manage the device lock feature:

Command 0xEF06 – Change Device Lock State
Command 0xEF07 – Change Device Lock Passcode
Property 1.2.5.2.1.1 Device Lock State
Property 1.2.5.2.1.2 Device Lock State After Reset
Property 1.2.5.2.1.3 Device Lock Passcode

Only the following commands are allowed when the device lock state is set to locked:

Command 0x1F01 - Reset Device
Command 0xD101 - Get Property
Command 0xD112 - Set Property (Secured)
Command 0xDF01 - Echo
Command 0xE001 - Get Challenge
Command 0xEF06 – Change Device Lock State
Command 0xEF11 - Get Key Info
Command 0xEEEE - Send Secured Command to Device

6 Commands

6.1 Command Group 0x10nn - Transactions

6.1.1 Command 0x1001 - Start Transaction

The host uses this command to start a payment transaction.

The sequence of events for transactions with card readers enabled is roughly as follows.
(MCE Only) The sequence for Manual Entry Mode is provided further below.

1) If the device is configured to enable user action event notifications using **Property 1.2.7.1.2.1 User Event Notification Controls Enable**, the cardholder may present a card or payment device before the host calls this command, and the device sends **Notification 0x1001 - Device Information Update** to the host to indicate it should call this command to start a transaction.
   a) **(EMV Contact Only | EMV Contactless Only)** In the case where the cardholder has inserted or tapped before the transaction started, the host should call this command as quickly as possible while the card is still in the slot, or while the card or contactless payment device is still within tap range. The device does not begin contact or contactless reads until after the host invokes this command and does not store any data from the cardholder action event before the transaction is in process.
2) The host composes a command request in the format below, and sends it to the device. It may cancel the transaction in process by calling **Command 0x1008 - Cancel Transaction**.
3) The device sends a response in the format below, and waits for the cardholder to present payment using one of the enabled payment technologies.
4) **(BCR Only)** If the cardholder scans a barcode, the device sends **Notification 0x0101 - Transaction Information Update** to report Barcode / Barcode Event / Type / Data Attached with the barcode data attached and terminates the transaction.
5) After the cardholder presents payment, the device sends **Notification 0x0101 - Transaction Information Update** to report the payment technology being used / **Card Event**.
6) **(EMV Contact Only)** If the cardholder has inserted a chip card and there is more than one application the device and card mutually support:
   a) **(No Display Only)** The device sends **Notification 0x1803 - User Interface Host Action Request** to report **Cardholder Selection Request**/ **Notification Payload** to prompt the cardholder to select the application to use. The host should respond by showing the prompt, receiving input from the cardholder, and calling **Command 0x1802 - Report Cardholder Selection** to report the selection result to the device.
7) The device sends **Notification 0x0101 - Transaction Information Update** to report the payment technology being used / **Data Update / ARQC Update / Data Attached**.
8) If the host specified **Quick Chip Transaction Flow** in the **Transaction Flow** parameter:
   a) The device immediately constructs its own internal ARPC Response, with tag 8A set to 'Z3' to coordinate the transaction with the card or other payment method, and sends the host **Notification 0x0105 - Transaction Operation Complete** to report the payment technology being used / **Kernel Outcome** / **Quick Chip Deferred** /outcome detail. A Transaction Option parameter can be set to display on amount or not.
   b) **(No Display only)** The device sends the host **Notification 0x1803 - User Interface Host Action Request** to report **Display**/**Display Message**/**Data Attached** with message **REMOVE CARD** to notify the cardholder the card can be removed.

c) The host should then process the ARQC Message data, including replacing the default amount with the final transaction amount, and should coordinate with the transaction processor to retrieve a final transaction result. Because in this case the device is not involved in determining the final transaction result, it does not send a notification to the host to show **APPROVED** or **DECLINED**.

i) **(No Display Only)** Instead, the host should use its local display to show an appropriate message to the cardholder, such as **APPROVED** or **DECLINED**, based on the final transaction result.

9) If the host specified **EMV Transaction Flow** in the **Transaction Flow** parameter:

a) The host processes the ARQC Message data and uses it to coordinate with the transaction processor to receive an ARPC Response, which it processes and sends to the device using **Command 0x1004 - Resume Transaction**.

b) The device waits for the period of time specified in **Property 1.1.1.1.1.5 ARPC Receive Timeout**. If an ARPC timeout occurs, the device sends the ARQC again based on the setting in **Property 1.1.1.1.1.6 ARPC Retry Attempts**.

c) **(EMV Contact Only)** If the cardholder inserted a contact chip card, the device communicates with the card to determine whether to approve or decline the transaction.

d) The device sends the host **Notification 0x0105 - Transaction Operation Complete** to report the payment technology being used / **Kernel Outcome** / **Approved** or **Declined** /outcome.

e) **(No Display Only)** The device sends the host **Notification 0x1803 - User Interface Host Action Request** to report **Display**/**Display Message**/**Data Attached** with message **APPROVED** or **DECLINED** to notify the cardholder of the transaction result.

10) The device sends **Notification 0x0105 - Transaction Operation Complete** to report the payment technology used / **Outcome** / the final result of the transaction.

11) **(EMV Contactless Only)** For NFC Tag, the flow is as follows:

(1) Use Start Transaction command with the NFC enabled in Contactless Reader Mode

(2) If an NFC Tag is Detected

a. the terminal will send a notification that identifies the NFC card type, see **Notification 0x0101 - Transaction Information Update**

b. the terminal will send another notification with the UID as a payload, see **Table 7.1-2 - Notification Payload for Data Update, ARQC Update (Quick Chip), Data Attached**. If a card is configured with a random ID, its value will be changed every time the card gets detected. The host is responsible to retrieve the real UID.

(3) No ARQC or BATCH data will be sent

(4) The host application can continue interfacing with the NFC tag by sending pass-through commands

(5) When an NFC Tag gets out of the field, the terminal sends **Notification 0x0105 – 20 05 00 00 (PICC, NFC Tag, Tag Removed, Reserved)** indicating that the Tag has been removed.

The host uses **Command 0x1001** to start a payment transaction. If the battery charge is five percent or less, a response is returned indicating that the command has not been executed. See **Table 6.1-3 - Response Example for Command 0x1001 – Start Transaction Command not executed due to Battery Charge State**.

Table 6.1-1 - Request Data for Command 0x1001 - Start Transaction

Tag	Len	Value / Description	Typ	Req	Default
Beginning of any wrappers, at minimum including **Request Message** found on page **32**					
1001 = **Command 0x1001 - Start Transaction**					

Tag	Len	Value / Description	Typ	Req	Default
81	01	Reserved		O	
82	01	Transaction Timeout, in seconds This parameter defines how long the device waits for the cardholder to take action on any cardholder input, for example, when waiting for the cardholder to present payment after the host starts the transaction. • 0x00 = No timeout • 0x01 to 0xFF = 1 to 255 seconds	B	R	
A3	var	Reader Options The parameters inside this TLV data object allow the host to enable and disable the various payment method interfaces.	T	O	
/82	01	Contact Reader Mode (EMV Contact Only) • 0x00 = Disabled • 0x01 = EMV	B	O	0x01
/83	01	Contactless Reader Mode (EMV Contactless Only) • 0x00 = Disabled • 0x01 = EMV • 0x02 = NFC • 0x03 = EMV and NFC	B	O	0x01
/85	02	Barcode Reader Mode (BCR Only) Populate this parameter to enable the device's barcode reader. This feature can be enabled alongside all other reader modes except Manual Entry Mode. Byte 1 Barcode Reader Enable • 0x00 = Disabled • 0x01 = Enabled Byte 2 Encrypt Non-EMV Barcode Data • 0x00 = Disabled • 0x01 = Enabled	B	O	0x0000
A4	var	Tip and Tax Options	B	O	
/81	1F	Byte 1 Tip Mode 0x00 – Disable Tip Mode 0x01 – Show Tip GUI immediately using % value 0x02 – Show Tip GUI immediately using $ amount 0x11 - Enable Read Channel(s), with +Tip Button, %value 0x12 - Enable Read Channel(s), with +Tip Button, $ Amount For Tip Modes 0x01 and 0x02, the Tip GUI is shown immediately with selected read channels disabled. They get enabled after the Tip information are entered.	B	O	

Tag	Len	Value / Description	Typ	Req	Default
		For Tip Modes 0x11 and 0x12, the read channels are enabled and "+Tip" button is shown. If no tip is desired, just present the card. If "+Tip" button is selected, read channels are first disabled, then show the Tip GUI. Byte 2, Display Mode for Button1 0 - % or Amount 1 – Display Custom 2 – Display NO TIP 3 – Disabled (An OID controls whether the button is blank, grayed out or not showing. Bytes 3 to 6 Value in % or Amount for Button 1, if applicable Byte 7, Display Mode for Button 2 (See Byte 2 for Details) Bytes 8 to 11 Value in % or Amount for Button 2, if applicable Byte 12, Display Mode for Button 3 (See Byte 2 for Details) Bytes 13 to 16 Value in % or Amount for Button 3, if applicable Byte 17, Display Mode for Button 4 (See Byte 2 for Details) Bytes 18 to 21 Value in % or Amount for Button 4, if applicable Byte 22, Display Mode for Button 5 (See Byte 2 for Details) Bytes 23 to 26 Value in % or Amount for Button 5, if applicable Byte 27, Display Mode for Button 6 (See Byte 2 for Details) Bytes 28 to 31 Value in % or Amount for Button 6, if applicable.			
/82	06	Tax or Surcharge Amount to Display	B	O	
84	02	Transaction Options This parameter is a bitmask (ORed bits) that sets various device behaviors that change the transaction flow or the way the device reports transaction results, as follows: Byte 1 Apple VAS Mode (**Apple / Google VAS Only, set to 0 if not supported**) Bits 0, 1 • 0x00 = **Apple/Google VAS Support Disabled** • 0x01 = **VAS App OR Payment Mode (Single Mode)**. The device reads only Apple/Google VAS data from a tapped smartphone, or reads EMV payment data from a tapped card. When the device sends ARQC to conclude the	B	O	0x0003

Tag	Len	Value / Description	Typ	Req	Default
		transaction, it only includes either EMV payment data in container FC for cards, or includes VAS data in container FE for smartphones • 0x02 = **VAS App and Payment Mode (Dual Mode)**. The device reads both Apple/Google VAS data and EMV payment data from a tapped smartphone, or reads EMV payment data from a tapped card. When device sends ARQC to the host to conclude the transaction, it includes EMV payment data in container FC and includes VAS data, if available, in container FE • 0x03 = **VAS App Only Mode (VAS Mode)**. The device reads only Apple/Google VAS data from a tapped smartphone, and does not read data from a tapped card. If the tapped smartphone does not support VAS, the device does not detect or read from the smartphone. When the device send ARQC to conclude the transaction, it includes VAS data in container FE and does not include EMV payment data in container FC • 0x04 = **Payment Only Mode (Payment Mode)**. The device operates the same as EMV mode (01). It reads only EMV payment data from a tapped smartphone or a tapped card. When the device sends ARQC to conclude the transaction, it includes EMV payment data in container FC and does not include VAS data in container FE. Bits 4, 5, 6 Wallet Mode 4 -Apple 5 - Google 6 - Reserved • 0x000 = Wallet Support Disabled • 0x001 = Apple VAS Enable • 0x002 = Google VAS Enabled • 0x003 = –Apple and Google VAS Enabled Bit 7 Apple VAS Protocol Mode ○ Value 0 – URL VAS Protocol ○ Value 1 – FULL VAS Protocol Byte 2 Transaction Flow Control • Bit 0 Transaction Flow ○ Value 1 = Quick Chip Transaction Flow ○ Value 0 = EMV Transaction Flow • Bit 1 Response Format			

Tag	Len	Value / Description	Typ	Req	Default
		○ Value 1 = DynaPro Response Format. For sending ARQC data and batch data, the device uses **EMV ARQC (DynaPro Format) Type** and **EMV Batch Data (DynaPro Format) Type**. ○ Value 0 = Reserved. • Bit 3 Display Amount for Quick Chip Transaction Flow ○ Value 1 = Display Amount ○ Value 0 = Do not Display Amount			
86	var	Transaction TLV This is a list of self-contained TLV data objects that defines the basic parameters for the transaction. It may contain any of the following tags in the formats defined in the ***EMV 4.3*** specification or payment brand specifications, but at minimum it must contain 9C and 9F02, plus 9F03 if the transaction includes cash back. • 9C Transaction Type • 9F02 Amount Authorized. If the **Transaction Flow** parameter specifies **Quick Chip Transaction Flow**, the host must specify a non-zero amount. • 9F03 Amount Other • 9F7C Merchant Custom Data • 5F2A Transaction Currency Code • 5F36 Transaction Currency Exponent • 9F53 Transaction Category Code • 9F15 Merchant Category Code • 9F16 Merchant ID	B	R/O	
AC	var	User Interface Options	T	O	null
/81	00	Suppress Thank You Message By default, devices with a display signal the end of a transaction by briefly showing "THANK YOU," then "WELCOME." The host can include this parameter to direct the device to suppress the "THANK YOU" message during this transaction.	T	O	null
/82	01	Override Final Transaction Message By default, devices with a display signal the end of a transaction by returning to the idle page and showing "WELCOME." The host can include this parameter to direct the device to show a different message, chosen from the list of available Display String IDs in section **4.3 Display Strings**. This option completely overrides the device's idle page behavior until the next transaction, power cycle, or other similar state change.	B	O	null
/83	02	Functional button Right option.	B	O	null

Tag	Len	Value / Description	Typ	Req	Default
		String ID = Enable the present card page with a green functional button Right – label with a String ID associates with a configured String message. See **Table 7.4-7 – Default User Interface String IDs and Strings.** This button can fit about 15 characters. When user presses this button, device sends notification to the host to indicate the present card functional button Right is pressed. See **Notification 0x1803 - User Interface Host Action Request** If host wants to disable this button, do not include this tag.			
End of any wrappers, at minimum including **Request Message** found on page **32**					

Table 6.1-2 - Response Data for Command 0x1001 - Start Transaction

Tag	Len	Value / Description	Typ	Req	Default
Beginning of any wrappers, at minimum including **Response Message** found on page **33**					
1001 = **Command 0x1001 - Start Transaction**					
No parameters.					
End of any wrappers, at minimum including **Response Message** found on page **33**					

Table 6.1-3 - Response Example for Command 0x1001 – Start Transaction Command not executed due to Battery Charge State

Example (Hex)
AA 00 81 04 82 01 10 01 82 04 80 02 03 16

If the request started successfully, the Request Status in the message wrapper is **OK, Started / Running, All good / requested operation was successful**.

Table 6.1-4 - Request Example (Contactless Only)

Example (Hex)
AA 00 81 04 01 00 10 01 84 3D 10 01 82 01 3C A3 09 81 01 00 82 01 00 83 01 01 84 02 00 03 86 27 9C 01 00 9F 02 06 00 00 00 00 01 00 9F 03 06 00 00 00 00 00 00 5F 2A 02 08 40 5F 36 01 02 9F 15 02 00 00 9F 53 01 00

Table 6.1-5 - Response Example

Example (Hex)
AA 00 81 04 82 01 10 01 82 04 01 00 00 00

6.1.2 Command 0x1004 - Resume Transaction

The host uses this command to provide the device with additional / modified data to resume a transaction that is currently paused.

Table 6.1-6 - Request Data for Command 0x1004 - Resume Transaction

Tag	Len	Value / Description	Typ	Req	Default
Beginning of any wrappers, at minimum including Request Message found on page 32					
1004 = **Command 0x1004 - Resume Transaction**					
81	01	Resume Code Indicates the pause state the transaction will resume from: • 0x00 = Waiting for ARPC	B	R	
83	var	Reserved	B	O	
84	var	ARPC Data This contains an **EMV ARPC Type**.	B	R	
86	var	Transaction TLV Update Not applicable when Resume Code = **Waiting for ARPC**	B	O	
End of any wrappers, at minimum including **Request Message** found on page **32**					

Table 6.1-7 - Response Data for Command 0x1004 - Resume Transaction

Tag	Len	Value / Description	Typ	Req	Default
Beginning of any wrappers, at minimum including **Response Message** found on page **33**					
1004 = **Command 0x1004 - Resume Transaction**					
No parameters.					
End of any wrappers, at minimum including **Response Message** found on page **33**					

If the request started successfully, the Request Status in the message wrapper is **OK, Started / Running, All good / requested operation was successful**.

Table 6.1-8 - Request Example

Example (Hex)
AA 00 81 04 01 00 10 04 84 21 10 04 81 01 00 82 01 78 84 17 FF 74 14 DF DF 25 08 99 26 90 E1 16 12 07 10 FA 06 70 04 8A 02 30 30

Table 6.1-9 - Response Example

Example (Hex)
AA 00 81 04 82 06 10 04 82 04 00 00 00 00 84 02 10 04

6.1.3 Command 0x1008 - Cancel Transaction

The host can use this command to cancel a transaction in progress that it initiated using **Command 0x1001 - Start Transaction**.

The sequence of events is as follows:

1) The host has already called **Command 0x1001 - Start Transaction** and the transaction is still in process.
2) The host constructs the command request in the format below.

3) The host sends the command request to the device.
4) The device sends a response in the format below to the host.
   a) If the transaction is in a state where it can not be canceled, the device's response returns operation status detail **Failed, Device State Issue, Cannot Cancel**.
   b) If there is no transaction in progress, the device's response returns operation status detail codes **Failed, Device State Issue, No Transaction**.
5) If the device successfully cancels the transaction, the device's response returns operation status detail **All Good, Requested Operation Was Successful**, shows **CANCELED** on the display (if any), and returns to the idle state.

Table 6.1-10 - Request Data for Command 0x1008 - Cancel Transaction

Tag	Len	Value / Description	Typ	Req	Default
Beginning of any wrappers, at minimum including **Request Message** found on page **32**					
1008 = **Command 0x1008 - Cancel Transaction**					
No parameters.					
End of any wrappers, at minimum including **Request Message** found on page **32**					

Table 6.1-11 - Response Data for Command 0x1008 - Cancel Transaction

Tag	Len	Value / Description	Typ	Req	Default
Beginning of any wrappers, at minimum including **Response Message** found on page **33**					
1008 = **Command 0x1008 - Cancel Transaction**					
No parameters.					
End of any wrappers, at minimum including **Response Message** found on page **33**					

Table 6.1-12 - Request Example

Example (Hex)
AA 00 81 04 01 13 10 08 84 02 10 08

Table 6.1-13 - Response Example

Example (Hex)
AA 00 81 04 82 13 10 08 82 04 00 00 00 00

6.2 Command Group 0x11nn - NFC/MIFARE Pass Through Commands (Contactless Only)

6.2.1 Command 0x1100 – Pass Through Command For NTag/MIFARE Ultralight, Type 2

After an NTag/MIFARE Ultralight is activated, the host uses this command to send commands and receive responses to and from a Ntag/MIFARE Ultralight. Do not change the address 0x00 for read protection of Ultralight C/AES card because the device will fail to access the card if the address 0x00 is read protected.

Table 6.2-1 - Request Data for Command 0x1100 – Pass Through Command For NTag/MIFARE Ultralight, Type 2.

Tag	Len	Value / Description	Typ	Req	Default
Beginning of any wrappers, at minimum including **Request Message** found on page 32					
1100 = **Command 0x1100 – Pass Through Command For NTag/MIFARE Ultralight, Type** 2					
81	var	Command to Send. See **Table 6.2-2 – NTag Commands** See **Table 6.2-3 – MIFARE Ultralight EV1 Commands** See **Table 6.2-4 – MIFARE Ultralight C Commands** See **Table 6.2-5 – MIFARE Ultralight AES Commands**	B	R	
82	01	00 – No Encrypt 01 - Encrypt	B	R	
83	01	00 – Expect More Commands 01 – FF (Last Command) If the pass-through command is the last successful command, the device will end the transaction with a single beep, indicating success. If an error arises, the device will end the transaction but will sound two beeps to indicate the error. The user should then remove the card.	B	R	
End of any wrappers, at minimum including Request Message found on page **32**					

Table 6.2-2 – NTag Commands

Command	Length	Field Value
Get Version	1	The GET_VERSION command is used to retrieve information on the NTAG family, the product version, storage size and other product data required to identify the specific NTAG21x. Byte 0 = 0x60
Read	2-3	The READ command requires a start page address, and returns the 16 bytes of four NTAG21x pages. For example, if address is 03h then pages 03h, 04h, 05h, 06h are returned. Special conditions apply if the READ command address is near the end of the accessible memory area. The special conditions also apply if at least part of the addressed pages is within a password protected area. The READ command with an option of end page address returns the all n*4 bytes of the addressed pages. For example if the start address is 03h and the end address is 07h then pages 03h, 04h, 05h, 06h and 07h are returned. Byte 0 = 0x30 Byte 1 = Start Page Address Byte 2 = (optional) End Page Address

Command	Length	Field Value
Fast Read	3	The FAST_READ command requires a start page address and an end page address and returns the all n*4 bytes of the addressed pages. For example, if the start address is 03h and the end address is 07h then pages 03h, 04h, 05h, 06h and 07h are returned. Byte 0 = 0x3A Byte 2 = Start Page Address Byte 3 = End Page Address
Write	6	The WRITE command requires a block address, and writes 4 bytes of data into the addressed NTAG21x page. Byte 0 = 0xA2 Byte 1 = Address to Write Byte 2 to 5 = 4 Bytes of Data to Write
Compatibility Write	18	The COMPATIBILITY_WRITE command is implemented to guarantee interoperability with the established MIFARE Classic PCD infrastructure, in case of coexistence of ticketing and NFC applications. Even though 16 bytes are transferred to NTAG21x, only the least significant 4 bytes (bytes 0 to 3) are written to the specified address. Set all the remaining bytes, 04h to 0Fh, to logic 00h. Byte 0 = 0xA0 Byte 1 = Address to Write Byte 2 to 17 = 16 Bytes of Data to Write (only least significant 4 bytes are written) Note: This command is sent in 2 steps, which the Firmware will handle (1) <CMD><Address to Write><CRCH><CRCL> (2) <16 Bytes of Data to Write><CRCH><CRCL>
READ_CNT	2	The READ_CNT command is used to read out the current value of the NFC one-way counter of the NTAG213, NTAG215 and NTAG216. The command has a single argument specifying the counter number and returns the 24-bit counter value of the corresponding counter. If the NFC_CNT_PWD_PROT bit is set to 1b the counter is password protected and can only be read with the READ_CNT command after a previous valid password authentication Byte 0 = 0x39 Byte 1 = 0x02 (NFC Counter Address)

Command	Length	Field Value
PWD_AUTH	5	A protected memory area can be accessed only after a successful password verification using the PWD_AUTH command. The AUTH0 configuration byte defines the protected area. It specifies the first page that the password mechanism protects. The level of protection can be configured using the PROT bit either for write protection or read/write protection. The PWD_AUTH command takes the password as parameter and, if successful, returns the password authentication acknowledge, PACK. By setting the AUTHLIM configuration bits to a value larger than 000b, the number of unsuccessful password verifications can be limited. Each unsuccessful authentication is then counted in a counter featuring anti-tearing support. After reaching the limit of unsuccessful attempts, the memory access specified in PROT, is no longer possible. Byte 0 = 0x1B Byte 1..4 = password (4 bytes)
READ_SIG	2	The READ_SIG command returns an IC specific, 32-byte ECC signature, to verify NXP Semiconductors as the silicon vendor. The signature is programmed at chip production and cannot be changed afterwards. Byte 0 = 0x3C Byte 1 = 0x00, RFU

Table 6.2-3 – MIFARE Ultralight EV1 Commands

Command	Length	Field Value
Get Version	1	The GET_VERSION command is used to retrieve information on the MIFARE family, product version, storage size and other product data required to identify the MF0ULx1. Byte 0 = 0x60
Read	2-3	The READ command requires a start page address, and returns the 16 bytes of four MIFARE Ultralight pages. For example if address (Addr) is 03h then pages 03h, 04h, 05h, 06h are returned. A rollover mechanism is implemented if the READ command address is near the end of the accessible memory area. This rollover mechanism is also used when at least part of the addressed pages is within a password protected area. The READ command with an option of end page address returns the all n*4 bytes of the addressed pages. For example if the start address is 03h and the end address is 07h then pages 03h, 04h, 05h, 06h and 07h are returned. Byte 0 = 0x30 Byte 1 = Start Page Address Byte 2 = (optional) End Page Address

Command	Length	Field Value
Fast Read	3	The FAST_READ command requires a start page address and an end page address and returns the all n*4 bytes of the addressed pages. For example if the start address is 03h and the end address is 07h then pages 03h, 04h, 05h, 06h and 07h are returned. Byte 0 = 0x3A Byte 2 = Start Page Address Byte 3 = End Page Address
Write	6	The WRITE command requires a block address, and writes 4 bytes of data into the addressed MIFARE Ultralight EV1 page. Byte 0 = 0xA2 Byte 1 = Address to Write Byte 2 to 5 = 4 Bytes of Data to Write
Compatibility Write	18	The COMPATIBILITY_WRITE command is implemented to accommodate the established MIFARE Classic PCD infrastructure. Even though 16 bytes are transferred to the MF0ULx1, only the least significant 4 bytes (bytes 0 to 3) are written to the specified address. Set all the remaining bytes, 04h to 0Fh, to logic 00h Byte 0 = 0xA0 Byte 1 = Address to Write Byte 2 to 17 = 16 Bytes of Data to Write (only least significant 4 bytes are written) Note: This command is sent in 2 steps, which the Firmware will handle (1) <CMD><Address to Write><CRCH><CRCL> (2) <16 Bytes of Data to Write><CRCH><CRCL>
READ_CNT	2	The READ_CNT command is used to read out the current value of one of the 3 one-way counters of the MF0ULx1. The command has a single argument specifying the counter number and returns the 24-bit counter value of the corresponding counter. The counters are always readable, independent on the password protection settings. Byte 0 = 0x39 Byte 1 = 0x00..0x02 (counter number from 0x00 to 0x02)
INCR_CNT	6	The INCR_CNT command is used to increment one of the 3 one-way counters of the MF0ULx1. The two arguments are the counter number and the increment value. Byte 0 = 0xA5 Byte 1 = 0x00..0x02 (counter number from 0x00 to 0x02) Byte 2 to 5 = 4 bytes increment value (only the 3 least significant bytes are relevant)

Command	Length	Field Value
PWD_AUTH	5	A protected memory area can be accessed only after a successful password verification using the PWD_AUTH command. The AUTH0 configuration byte defines the protected area. It specifies the first page that the password mechanism protects. The level of protection can be configured using the PROT bit either for write protection or read/ write protection. The PWD_AUTH command takes the password as parameter and, if successful, returns the password authentication acknowledge, PACK. By setting the AUTHLIM configuration bits to a value larger than 000b, the number of unsuccessful password verifications can be limited. Each unsuccessful authentication is then counted in a counter featuring anti-tearing support. After reaching the limit of unsuccessful attempts, the memory access specified in PROT, is no longer possible. Byte 0 = 0x1B Byte 1..4 = password (4 bytes)
READ_SIG	2	The READ_SIG command returns an IC specific, 32-byte ECC signature, to verify NXP Semiconductors as the silicon vendor. The signature is programmed at chip production and cannot be changed afterwards. Byte 0 = 0x3C Byte 1 = 0x00, RFU
CHECK TEARING_EVENT	2	The CHECK_TEARING_EVENT command enables the application to identify if a tearing event happened on a specified counter element. It takes the counter number as single argument and returns a specified valid flag for this counter. If the returned valid flag is not equal to the predefined value, a tearing event happened. Note, although a tearing event might have happened on the counter, a valid value corresponding to the last valid counter status is still available using the READ_CNT command. Byte 0 = 0x3E Byte 1 = 0x00..0x02 (counter number from 0x00 to 0x02)

Command	Length	Field Value
VCSL	21	The VCSL command is used to enable a unique identification and selection process across different MIFARE product-based cards and card implementations on mobile devices. The command requires a 16-byte installation identifier IID and a 4-byte PCD capability value as parameters. The parameters are present to support compatibility to other MIFARE product-based devices but are not used or checked inside the MF0ULx1. Nevertheless, the number of bytes is checked for correctness. The answer to the VCSL command is the virtual card type identifier VCTID. This identifier indicates the type of card or ticket. Using this information, the reader can decide whether the ticket belongs to the installation or not. Byte 0 = 0x4B Byte 1 to 16 = 16-byte IID (installation identifier, can be any number) Byte 17 to 20 = 4-byte PCDCAPS (PCD capabilities, can be any number)

Table 6.2-4 – MIFARE Ultralight C Commands

Command	Length	Field Value
Read	2-3	The READ command takes the page address as a parameter. Only addresses 00h to 2Bh are decoded. For higher addresses the MF0ICU2 returns a NAK. The MF0ICU2 responds to the READ command by sending 16 bytes starting from the page address defined in the command (e.g. if ADR is 03h, pages 03h, 04h, 05h, 06h are returned) A roll-over mechanism is implemented to continue reading from page 00h once the end of the accessible memory is reached. For example, reading from address 29h on a MF0ICU2 results in pages 29h, 2Ah, 2Bh and 00h being returned. The following conditions apply if part of the memory is protected by the 3DES authentication for read access: • if the MF0ICU2 is in the ACTIVE state – addressing a page which is equal or higher than AUTH0 results in a NAK response – addressing a page lower than AUTH0 results in data being returned with the roll-over mechanism occurring just before the AUTH0 defined page • if the MF0ICU2 is in the AUTHENTICATED state – the READ command behaves like on a MF0ICU2 without access protection. The READ command with an option of end page address returns the all n*4 bytes of the addressed pages. For example if the start address is 03h and the end address is 07h then pages 03h, 04h, 05h, 06h and 07h are returned. Byte 0 = 0x30 Byte 1 = Start Page Address Byte 2 = (optional) End Page Address

Command	Length	Field Value
Write	6	The WRITE command is used to program the lock bytes in page 02h, the OTP bytes in page 03h, data bytes in pages 04h to 27h, configuration data from page 28h to 2B and keys from page 2Ch to 2Fh. A WRITE command is performed page-wise, programming 4 bytes in a page. Byte 0 = 0xA2 Byte 1 = Address to Write Byte 2 to 5 = 4 Bytes of Data to Write
Compatibility Write	18	The COMPATIBILITY WRITE command was implemented to accommodate the established MIFARE PCD infrastructure. Even though 16 bytes are transferred to the MF0ICU2, only the least significant 4 bytes (bytes 0 to 3) will be written to the specified address. It is recommended to set the remaining bytes 4 to 15 to all 0. Byte 0 = 0xA0 Byte 1 = Address to Write Byte 2 to 17 = 16 Bytes of Data to Write (only least significant 4 bytes are written) Note: This command is sent in 2 steps, which the Firmware will handle (1) <CMD><Address to Write><CRCH><CRCL> (2) <16 Bytes of Data to Write><CRCH><CRCL>
AUTHENTICATE	2	The AUTHENTICATE command is used to authenticate the MF0ICU2 using 2 keys 3DES encryption in Cipher-Block Chaining (CBC) mode as described in ISO/IEC 10116. • The 16-byte of the 2keys 3DES are programmed to card memory pages from 2Ch to 2Fh. The key itself can be written during personalization or at any later stage using the WRITE or COMPATIBILITY WRITE with Byte 0 is always sent first. On example of Key1 = 0001020304050607h and Key2 = 08090A0B0C0D0E0Fh, the command sequence needed for key programming with WRITE command is: • A2 2C 07 06 05 04 • A2 2D 03 02 01 00 • A2 2E 0F 0E 0D 0C • A2 2F 0B 0A 09 08 • The 16-byte of the same 2keys 3DES are programed to the Device using **Property 1.2.1.1.4.1 MIFARE Ultralight C 2keys3DES** Byte 0 = 0x1A Byte 1 = 0x00

Table 6.2-5 – MIFARE Ultralight AES Commands

Command	Length	Field Value
Get Version	1	The GET_VERSION command is used to retrieve information on the MIFARE family, product version, storage size and other product data required to identify the MIFARE Ultralight AES. Byte 0 = 0x60
Read	2-3	The READ command requires a start page address, and returns the 16 bytes of four pages. For example, if address (Addr) is 03h then pages 03h, 04h, 05h, 06h are returned. So called roll-over mechanism (described later) applies if the READ command address is near the end of the accessible memory area. Same mechanism applies if at least part of the addressed pages is within an authentication protected area In the default state of MIFARE Ultralight AES, all memory pages in the range from 00h to 3Bh are allowed as Addr parameter to the READ command. Addressing a memory page above the limit results in a NAK response. A roll-over mechanism is implemented to continue reading from page 00h once the end of the accessible memory is reached if at least first addressed page is within allowed limit. **Remark**: AES key values can never be directly read out of the memory. When reading from the pages holding key values, all 00h bytes are returned. The READ command with an option of end page address returns the all n*4 bytes of the addressed pages. For example if the start address is 03h and the end address is 07h then pages 03h, 04h, 05h, 06h and 07h are returned. Byte 0 = 0x30 Byte 1 = Start Page Address Byte 2 = (optional) End Page Address
Fast Read	3	The FAST_READ command requires a start page address and an end page address and returns bytes of addressed pages. For example if the start address is 03h and the end address is 07h then pages 03h, 04h, 05h, 06h, and 07h are returned. If either start or end address is outside accessible area, then MIFARE Ultralight AES replies with a NAK. Byte 0 = 0x3A Byte 2 = Start Page Address Byte 3 = End Page Address
Write	6	The WRITE command requires a block address, and writes 4 bytes of data into the addressed MIFARE Ultralight AES page. Byte 0 = 0xA2 Byte 1 = Address to Write Byte 2 to 5 = 4 Bytes of Data to Write

Command	Length	Field Value
READ_CNT	2	The READ_CNT command is used to read out the current value of one of the 3 one-way counters of MIFARE Ultralight AES. The command has a single argument specifying the counter number and returns the 24-bit counter value of the corresponding counter. Counters are always readable, except in case of the counter "0x02" with the optional AES authentication protection enabled. In that case, the counter 0x02 is readable only in the AUTHENTICATE state. Byte 0 = 0x39 Byte 1 = 0x00..0x02 (counter number from 0x00 to 0x02)
INCR_CNT	6	The INCR_CNT command is used to increment one of the 3x one-way counters of the MIFARE Ultralight AES. Two arguments are the counter number and the increment value. Counters are always incrementable, except in case of the counter "0x02" with the optional AES authentication protection enabled. In that case, the counter 0x02 can be incremented only in the AUTHENTICATE state. Byte 0 = 0xA5 Byte 1 = 0x00..0x02 (counter number from 0x00 to 0x02) Byte 2 to 5 = 4 bytes increment value (only the 3 least significant bytes are relevant)
READ_SIG	2	The READ_SIG command returns an IC-specific, 48-byte ECC signature. The originality signature can be changed if it has been unlocked with the LOCK_SIG command. Byte 0 = 0x3C Byte 1 = 0x00, RFU
WRITE_SIG	6	The WRITE_SIG command allows the writing of a customized originality signature into the dedicated originality signature memory. The WRITE_SIG command requires an originality signature block address, and writes 4 bytes of data into the addressed originality signature block. In the initial state of MIFARE Ultralight AES, the following originality signature blocks 00h to 0Bh are valid Addr parameters to the WRITE_SIG command. Addressing a memory block beyond the limits above results in a NAK response from MIFARE Ultralight AES. If the originality signature is locked or permanently locked, a WRITE_SIG command results in a NAK response from the MIFARE Ultralight AES. Byte 0 = 0xA9 Byte 1 = signature block address Byte 2 to 5 = signature bytes to be written

Command	Length	Field Value
LOCK_SIG	2	The LOCK_SIG command allows the user to unlock, lock or permanently lock the dedicated originality signature memory. The originality signature can only be unlocked, if the originality signature is not permanently locked. There is no command to unlock the originality signature, if the originality signature is permanently locked. Byte 0 = 0xAC Byte 1 = lock option • 0x00 = unlock • 0x01 = lock • 0x02 = permanently lock
VCSL	21	The VCSL command is used to enable a unique identification and selection process across different physical MIFARE product-based cards and virtual MIFARE implementations. The command requires a 16-byte installation identifier IID and a 4-byte PCD capability value as parameters. The parameters are present to support compatibility to other MIFARE product-based devices, but are not used or checked inside the MIFARE Ultralight AES. Nevertheless, the number of bytes is checked for correctness. The answer to the VCSL command is the VCTID value stored in the user configuration segment. This identifier indicates the type of card or ticket. Using this information, the contactless reader can decide whether the ticket belongs to the installation or not. Byte 0 = 0x4B Byte 1 to 16 = 16-byte IID (installation identifier, can be any number) Byte 17 to 20 = 4-byte PCDCAPS (PCD capabilities, can be any number)

AUTHENTICATE	2	The AUTHENTICATE command is used to authenticate with a 3-pass mutual authentication the MIFARE Ultralight AES and PCD. The cryptographic method is based on AES in Cipher-Block chaining (CBC) mode according to NIST Special Publication 800-38A. The used key is a 128-bit AES Key. Remark: To reduce the risk on card-only side channel attack to the AES keys, a failed authentication limit (AUTH_LIM) can be set. • The 16 bytes of the AES [DataProtKey] are programmed to memory pages from 30h to 33h. Keys themselves can be written during personalization or at any later stage in a secure environment, as long as the key is not locked for update in the user configuration segment. AES [UIDRetrKey] is stored in memory addresses from 34h until 37h. In case keys are not locked, MIFARE Ultralight AES allows to change AES-keys without authentication as long as AUTH0 is not set to a page address before or at page address where keys bytes are stored. Otherwise MIFARE Ultralight AES requires to be in the AUTHENTICATED state to allow to write AES keys. The key itself can be written using the WRITE with Byte 0 is always sent first. On example of AES [DataProtKey] = 000102030405060708090A0B0C0D0E0Fh, the command sequence needed for key programming with WRITE command is: • A2 30 0F 0E 0D 0C • A2 31 0B 0A 09 08 • A2 32 07 06 05 04 • A2 33 03 02 01 00 On example of AES [UIDRetrKey] = 000102030405060708090A0B0C0D0E0Fh, the command sequence needed for key programming with WRITE command is: • A2 34 0F 0E 0D 0C • A2 35 0B 0A 09 08 • A2 36 07 06 05 04 • A2 37 03 02 01 00 • The 16-byte of the same AES [DataProtKey] are programed to the Device using **Property 1.2.1.1.4.2 MIFARE Ultralight AES DataProtKey**. • The 16-byte of the same AES [UIDRetrKey] are programed to the Device using **Property 1.2.1.1.4.3 MIFARE Ultralight AES UIDRetrKey**. • The 16-byte of the AES [OriginalityKey] are programed to the Device using **Property 1.2.1.1.4.4 MIFARE Ultralight AES OriginalityKey.** This key value is only known by NXP. Byte 0 = 0x1A Byte 1 = Key option 0x00 = DataProtKey 0x01 = UIDRetrKey 0x02 = OriginalityKey

Table 6.2-6 - Response Data for Command 0x1100 – Pass Through Command For NTag/MIFARE Ultralight, Type 2

Tag	Len	Value / Description	Typ	Req	Default
Beginning of any wrappers, at minimum including Response Message found on page **33**					
1100 = Command 0x1100 – Pass Through Command For NTag/MIFARE Ultralight, Type 2 Command For NFC Tag					
81	01	Tag Response Code 0x00 = Success 0x01 = Failed	B	R	N/A
82	var	Encryption Control If encrypted, see Table 6.2-9 - Payload for Encrypted NFC/MIFARE Data If unencrypted see Table 6.2-10 – Unencrypted NFC/MIFARE Data	B	O	N/A
End of any wrappers, at minimum including Response Message found on page **33**					

If the request started successfully, the Request Status in the message wrapper is **OK, Started / Running, All good / requested operation was successful**.

Table 6.2-7 - Request Example (Get Version)

Example (Hex)
AA 00 81 04 01 39 11 00 84 0B 11 00 81 01 60 82 01 00 83 01 00

Table 6.2-8 - Response Example (Get Version)

Example (Hex)
AA 00 81 04 82 39 11 00 82 04 01 00 00 00 84 14 11 00 81 01 00 82 0D FC 0B DF 7A 08 01 02 03 04 05 06 07 08

6.2.1.1 Encrypted Data Format

Table 6.2-9 - Payload for Encrypted NFC/MIFARE Data

Tag	Len	Value / Description	Typ	Req	Default
/DFDF59	var	Encrypted Data Primitive Decrypt the value of this TLV data object using the algorithm and variant specified in the **Encrypted Data KSN** parameter and the **Encrypted Data Encryption Type** parameter to read its contents. The format of the decrypted data is shown in **Table 6.2-10 – Unencrypted NFC/MIFARE** Data.	B	R	
/DFDF50	var	Encrypted Data KSN	B	R	
/DFDF51	01	Encrypted Data Encryption Type See section 4.4 Encryption Type for a list of valid values.	B	R	

Tag	Len	Value / Description	Typ	Req	Default
End of Notification Message found on page **38**					

Table 6.2-10 – Unencrypted NFC/MIFARE Data

Tag	Len	Value / Description	Typ	Req	Default
FC	var	NFC Data Container	T	R	
/DF7A	var	NFC Data	B	O	

6.2.2 Command 0x1101 – Pass Through Command for MIFARE Classic/MINI®/Plus SL1 (Security Level 1), Type 2

After a MIFARE Tag is activated, the host uses this command to send commands and receive responses to and from a MIFARE tag.

For MIFARE Plus EV1/EV2/SE/X at SL1 (Security Level 1), the tag is discovered as MIFARE Classic, and can use the same functionality as MIFARE Classic 1K/4K commands in **Table 6.2-12 – MIFARE Classic/MINI®** Commands**.** Furthermore, an additional optional AES authentication is available in this level without affecting the MIFARE Classic 1K/4K functionality. The authenticity of the card can be proven using strong cryptographic means with this additional functionality. In addition to the backwards compatibility mode, MIFARE Plus card can be switched to higher security levels. After MIFARE Plus is authenticated with AES Security Level 1 Key, the Device doesn't auto detect an error from the MIFARE Tag has been removed to end the pass-through session. To end the pass-through session, the Host application can send the last command, CANCEL command (0xFF), or receive error response from the MIFARE Tag.

Table 6.2-11 - Request Data for Command 0x1101 - Pass Through Command for MIFARE Classic/MINI®/Plus SL1 (Security Level 1)

Tag	Len	Value / Description	Typ	Req	Default
Beginning of any wrappers, at minimum including **Request Message** found on page 32					
1101 = **Command 0x1101 – Pass Through Command for MIFARE Classic/MINI®/Plus SL1 (Security Level 1), Type 2**					
81	var	Command to Send. See **Table 6.2-12 – MIFARE Classic/MINI®** Commands See **Table 6.2-13 – MIFARE Plus EV1/EV2/SE/X SL1 (Security Level 1)** Commands	B	R	
82	01	00 – No Encrypt 01 - Encrypt			
83	01	00 – Expect More Commands 01 – FF (Last Command) If last command, Device will provide a single beep after receiving a successful response from tag, otherwise, device will provide a double beep	B	R	
End of any wrappers, at minimum including Request Message found on page **32**					

Table 6.2-12 – MIFARE Classic/MINI® Commands

Command	Length	Field Value
MIFARE Read		Byte 0 – 0x30 – Read Command Byte 1- Sector Number to Read Byte 2 – Start Block Number Byte 3 – End Block Number Byte 4 – Key Type, 0 = A, 1 = B Byte 5 to 10 = 6 Byte Key
MIFARE Write		Byte 0 – 0xA0 – Write Command Byte 1- Sector Number to Write Byte 2 – Start Block Number Byte 3 – End Block Number Byte 4 – Key Type 0 = A, 1 = B Byte 5 to 10 = 6 Byte Key Byte 11 to x = Variable length Byte Data (16 bytes per block)
MIFARE Increment		Byte 0 – 0xC1 – Increment Command Byte 1 – Source Sector Number Byte 2- Source Block number Byte 3 – Key Type 0 = A, 1 = B Byte 4 to 9 = 6 Byte Key Byte 10 to 13 = 4 Byte Operand
MIFARE Decrement		Byte 0 – 0xC0 – Decrement Command Byte 1 – Source Sector Number Byte 2- Source Block number Byte 3 – Key Type 0 = A, 1 = B Byte 4 to 9 = 6 Byte Key Byte 10 to 13 = 4 Byte Operand
MIFARE Restore		Byte 0 – 0xC2 – Restore Command Byte 1 – Source Sector Number Byte 2 - Source Block number Byte 3 – Key Type 0 = A, 1 = B Byte 4 to 9 = 6 Byte Key
MIFARE Transfer		Byte 0 – 0xB0 – Write the value from the Transfer Buffer into destination block number Byte 1 – Destination Sector Number Byte 2 - Destination Block number Byte 3 – Key Type 0 = A, 1 = B Byte 4 to 9 = 6 Byte Key

Table 6.2-13 – MIFARE Plus EV1/EV2/SE/X SL1 (Security Level 1) Commands

Command	Length	Field Value	EV1	EV2	SE	X
First Authenticate (part1 and part2)	3	First Authenticate. Use this command to switch to higher security levels. This command is behaved as the last command. Device will provide a single beep after receiving a successful response from a card, otherwise, device will provide a double beep. Byte 0 = 0x70 Byte 1-2 = Level 2 Switch Key (MIFARE Plus X only), or Level 3 Switch Key. See NXP doc ds206234, table 113. Byte 3 = MIFARE Plus AES_Key# • 0x01 = AES_Key1 = 16 bytes value stored in **Property 1.2.1.1.4.5 MIFARE Plus AES_Key1.** • 0x02 = AES_Key2 = 16 bytes value stored in **Property 1.2.1.1.4.6 MIFARE Plus AES_Key2.** • 0x03 = AES_Key3 = 16 bytes value stored in **Property 1.2.1.1.4.7 MIFARE Plus AES_Key3.** • 0x04 = AES_Key4 = 16 bytes values stored in **Property 1.2.1.1.4.8 MIFARE Plus AES_Key4.** • 0x05 = AES_Key5 = 16 bytes values stored in **Property 1.2.1.1.4.9 MIFARE Plus AES_Key5.** • 0x06 = AES_Key6 = 16 bytes values stored in **Property 1.2.1.1.4.A MIFARE Plus AES_Key6.**	Y	Y	Y	Y
Following Authenticate (part1 and part2)	3	Following Authenticate. Use this command for an option to put the NFC tag in Security Level 1 AES Authenticated before sending MIFARE Classic commands. Byte 0 = 0x76 Byte 1-2 = Security Level 1 Card Authentication Key. See NXP doc ds206234, table 113. Byte 3 = MIFARE Plus AES_Key# • 0x01 = AES_Key1 = 16 bytes value stored in **Property 1.2.1.1.4.5 MIFARE Plus AES_Key1.** • 0x02 = AES_Key2 = 16 bytes value stored in **Property 1.2.1.1.4.6 MIFARE Plus AES_Key2.** • 0x03 = AES_Key3 = 16 bytes value stored in **Property 1.2.1.1.4.7 MIFARE Plus AES_Key3.** • 0x04 = AES_Key4 = 16 bytes values stored in **Property 1.2.1.1.4.8 MIFARE Plus AES_Key4.** • 0x05 = AES_Key5 = 16 bytes values stored in **Property 1.2.1.1.4.9 MIFARE Plus AES_Key5.** • 0x06 = AES_Key6 = 16 bytes values stored in **Property 1.2.1.1.4.A MIFARE Plus AES_Key6.**	Y	Y	Y	Y

Command	Length	Field Value	EV1	EV2	SE	X
READ_SIG	2	The READ_SIG command returns an IC-specific, 48-byte ECC originality check signature. Byte 0 = 0x3C Byte 1 = 0x00, RFU	Y	Y	N	N
Personalize UID	2	Set anti-collision, selection and authentication behavior. The execution of this command requires an authentication to MF Classic sector 0. Once this command has been issued and accepted by the PICC, the configuration is automatically locked. A subsequently issued 'Personalize UID Usage' command is not executed and fails. Byte 0 = 0x40 Byte 1 = Encoded type of UID usage: 0x00 = UIDF0 = anti-collision and selection with the double size UID (7-byte) according to ISO/IEC14443-3 0x40 = UIDF1 = anti-collision and selection with the double size UID (7-byte) according to ISO/IEC 14443-3 and optional usage of a selection process shortcut 0x20 = UIDF2 = anti-collision and selection with a single size random ID (4-byte) according to ISO/IEC14443-3. After the card is configured with random ID, it won't be able perform any MF Classic authentication since MF Classic authentication requires UID. 0x60 = UIDF3 = anti-collision and selection with a single size NUID (4-byte) according to ISO/IEC14443-3 where the NUID is calculated out of the 7-byte UID	Y	Y	N	N
CANCEL	1	This command is used to terminate the pass-through command session. Byte 0 = 0xFF	Y	Y	Y	Y

Table 6.2-14 - Response Data for Command 0x1101 – Pass Through Command for MIFARE Classic/MINI®/Plus SL1 (Security Level 1), Type 2

Tag	Len	Value / Description	Typ	Req	Default
Beginning of any wrappers, at minimum including Response Message found on page **33**					
1101 = **Command 0x1101 – Pass Through Command for MIFARE Classic/MINI®/Plus SL1 (Security Level 1), Type 2**					
81	var	Tag Response Code Byte 0 = 0x00 = Success Byte 0 = 0x01 = I/O Failed Byte 0 = 0x02 Authentication Failed Byte 1 = 0x01 = Block that Failed (optional)	B	R	N/A

Tag	Len	Value / Description	Typ	Req	Default
82	var	Encryption Control If encrypted, see **Table 6.2-9 - Payload for Encrypted NFC/MIFARE** Data If unencrypted see **Table 6.2-10 – Unencrypted NFC/MIFARE** Data	B	O	N/A
End of any wrappers, at minimum including Response Message found on page **33**					

If the request started successfully, the Request Status in the message wrapper is **OK, Started / Running, All good / requested operation was successful**.

Table 6.2-15 - Request Example (Read Sector 0, Block Number Start 0 - End 0, KeyType A, Key = FFFFFFFFFFFF)

Example (Hex)
AA 00 81 04 01 19 11 01 84 15 11 01 81 0B 30 00 00 00 00 FF FF FF FF FF FF 82 01 00 83 01 00

Table 6.2-16 - Response Example (Read Sector 0, Block Number Start 0 - End 0, KeyType A, Key = FFFFFFFFFFFF)

Example (Hex)
AA 00 81 04 82 19 11 01 82 04 01 00 00 00 84 1C 11 01 81 01 00 82 15 FC 13 DF 7A 10 A4 FB 0D 3E 6C 08 04 00 03 0D C0 90 EE BF BB 1D

6.2.2.1 Encrypted Data Format

Table 6.2-17 - Payload for Encrypted NFC/MIFARE Data

Tag	Len	Value / Description	Typ	Req	Default
/DFDF59	var	Encrypted Data Primitive Decrypt the value of this TLV data object using the algorithm and variant specified in the **Encrypted Data KSN** parameter and the **Encrypted Data Encryption Type** parameter to read its contents. The format of the decrypted data is shown in **Table 7.4-6**.	B	R	
/DFDF50	var	Encrypted Data KSN	B	R	
/DFDF51	01	Encrypted Data Encryption Type See section 4.4 Encryption Type for a list of valid values.	B	R	
End of any wrappers, at minimum including Response Message found on page **33**					

Table 6.2-18 – Unencrypted NFC/MIFARE Data

Tag	Len	Value / Description	Typ	Req	Default
FC	var	NFC/MIFARE Data Container	T	R	
/DF7A	var	NFC/MIFARE Data	B	O	

6.2.3 Command 0x1102 – Pass Through Command for MIFARE DESFire, Type 4

After a MIFARE DESFire Light/EV1/EV2/EV3 Tag is activated, the host uses this command to send commands and receive responses to and from a MIFARE DESFire Tag.
There will be a fixed 30 second timeout for commands that require multiple command/responses.

Table 6.2-19 - Request Data for Command 0x1102 – Pass Through Command for MIFARE DESFire, Type 4

Tag	Len	Value / Description	Typ	Req	Default
Beginning of any wrappers, at minimum including **Request Message** found on page 32					
1102 = **Command 0x1102 – Pass Through Command for MIFARE DESFire, Type 4**					
81	var	Command to Send. See DESFire Data Sheet (MF2DLHX0) Should follow ISO 7816-4 APDU format • C-APDU ○ CLA INS P1 P2 Lc Data Le	B	R	
82	01	00 – No Encrypt 01 - Encrypt			
83	01	00 – Expect More Commands 01 – FF (Last Command) If last command, Device will provide a single beep after receiving a successful response from tag, otherwise, device will provide a double beep	B	R	
End of any wrappers, at minimum including Request Message found on page **32**					

Table 6.2-20 - Response Data for Command 0x1102 – Pass Through Command for MIFARE DESFire, Type 4

Tag	Len	Value / Description	Typ	Req	Default
Beginning of any wrappers, at minimum including Response Message found on page **33**					
1102 = **Command 0x1102 – Pass Through Command for MIFARE DESFire, Type 4**					
81	02	Tag Response (SW1 SW2) See DESFire Data Sheet (MF2DLHX0) Should follow ISO 7816-4 APDU format • SW1 and SW2 of R-APDU If card is not able to respond: • SW1 = 0x64, SW2 = 0x00	B	R	N/A
82	var	Tag Data • Data of R-APDU Encryption Control If encrypted, see **Table 6.2-9 - Payload for Encrypted NFC/MIFARE** Data	B	O	N/A

Tag	Len	Value / Description	Typ	Req	Default
		If unencrypted see **Table 6.2-10 – Unencrypted NFC/MIFARE** Data			
End of any wrappers, at minimum including Response Message found on page **33**					

If the request started successfully, the Request Status in the message wrapper is **OK, Started / Running, All good / requested operation was successful**.

Table 6.2-21 - Request Example (Get Version Part 1)

Example (Hex)
AA 00 81 04 01 13 11 02 84 0F 11 02 81 05 90 60 00 00 00 82 01 00 83 01 00

Table 6.2-22 - Response Example (Get Version Part 1)

Example (Hex)
AA 00 81 04 82 13 11 02 82 04 01 00 00 00 84 14 11 02 81 02 91 AF 82 0C FC 0A DF 7A 07 04 08 01 30 00 13 05

6.2.3.1 Encrypted Data Format

Table 6.2-23 - Payload for Encrypted NFC/MIFARE Data

Tag	Len	Value / Description	Typ	Req	Default
/DFDF59	var	Encrypted Data Primitive Decrypt the value of this TLV data object using the algorithm and variant specified in the **Encrypted Data KSN** parameter and the **Encrypted Data Encryption Type** parameter to read its contents. The format of the decrypted data is shown in **Table 7.4-6**.	B	R	
/DFDF50	var	Encrypted Data KSN	B	R	
/DFDF51	01	Encrypted Data Encryption Type See section **4.4 Encryption Type** for a list of valid values.	B	R	
End of Notification Message found on page **38**					

Table 6.2-24 – Unencrypted NFC/MIFARE Data

Tag	Len	Value / Description	Typ	Req	Default
FC	var	NFC/MIFARE Data Container	T	R	
/DF7A	var	NFC/MIFARE Data	B	O	

6.2.4 Command 0x1103 – Pass Through Command for MIFARE Plus, Type 2

After a MIFARE Plus EV1/EV2/SE/X Tag is activated, the Host uses this command to send commands and receive responses to and from a MIFARE Plus tag.

For MIFARE Plus SE/X, the Device will not auto detect an error from the MIFARE Tag that has been removed to end the pass-through session. To end the pass-through session, the Host application can send the last command, CANCEL command (0xFF), or receive error response from the MIFARE Tag.

After the card is configured to successfully switch to Security Level 1, the card will be discovered as MIFARE Classic 1K/4K and can use the same functionality as MIFARE Classic 1K/4K commands.

For more details, please refer to NXP NDA documentation ds206234-Product data sheet MIFARE Plus Functionality of implementations on smart card controllers (3.4)

Table 6.2-25 - Command 0x1103 – Pass Through Command for MIFARE Plus, Type 2

Tag	Len	Value / Description	Typ	Req	Default
Beginning of any wrappers, at minimum including **Request Message** found on page 32					
1103 = Command 0x1103 – Pass Through Command for MIFARE Plus, Type 2					
81	var	Command to Send. See **Table 6.2-26 - MIFARE Plus EV1/EV2/SE/X SL0 (Security Level 0) Commands** See **Table 6.2-27 – MIFARE Plus EV1/EV2/SE/X SL3 (Security Level 3) Commands**	B	R	
82	01	00 – No Encrypt 01 - Encrypt			
83	01	00 – Expect More Commands 01 – FF (Last Command) If this is the last command, the Device will provide a single beep after receiving a successful response from the tag, otherwise, the device will provide a double beep	B	R	
End of any wrappers, at minimum including Request Message found on page **32**					

Table 6.2-26 - MIFARE Plus EV1/EV2/SE/X SL0 (Security Level 0) Commands

Command	Length	Field Value	EV1	EV2	SE	X
GET_VERSION	1	The GET_VERSION command is used to retrieve manufacturing related data of the MIFARE Plus EV1/EV2 cards Byte 0 = 0x60	Y	Y	N	N
READ_SIG	2	The READ_SIG command returns an IC-specific, 48-byte ECC originality check signature of MIFARE Plus EV1/EV2 cards. Byte 0 = 0x3C Byte 1 = 0x00, RFU	Y	Y	N	N

WRITE_PERSO	19	The WRITE_PERSO command is used to pre-personalize AES keys and data from the initial delivery configuration to a customer specific value. Byte 0 = 0xA8 Byte 1-2 = Number of Block or Key to be written to (MSB first). See NXP doc ds206234, table 113. Byte 3 to 18 = 16 bytes value of the key or data which shall be written (in plain)	Y	Y	Y	Y
COMMIT_PERSO	2	The COMMIT_PERSO command is used to finalize the personalization and switch up to security level 1 or security level 3. For MIFARE Plus EV1/EV2, the following mandatory AES keys must be written using the WRITE_PERSO command before it can be switched to security level 1 or security level 3. • Card Configuration Key • Card Master Key • Level 2 Switch Key • Level 3 Switch Key For MIFARE Plus SE, the following mandatory AES keys must be written using the WRITE_PERSO command before it can be switched to security level 1 (for L1 card) or security level 3 (for L3 card). • Card Configuration Key • Card Master Key • Level 3 Switch Key For MIFARE Plus X, the following mandatory AES keys must be written using the WRITE_PERSO command before it can be switched to security level 1 (for L1 card) or security level 3 (for L3 card). • Card Configuration Key • Card Master Key • Level 2 Switch Key (for L1 card) • Level 3 Switch Key (for L1 card) Byte 0 = 0xAA Byte 1 = Security Level Option for EV1 and EV2 cards • 0x01 = Security Level 1 • 0x03 = Security Level 3 • Other values = Invalid. Device will return error. Byte 1 = 0x00 for SE and X cards. The Device will return error for other values. It is also highly recommended to change all sector AES keys as well as the data within this security level in a secure environment. This command is behaved as the last command. The Device will provide a single beep after receiving a successful	Y	Y	Y	Y

		response from a card, otherwise, device will provide a double beep.				
CANCEL	1	This command is used to terminate the pass-through command session. Byte 0 = 0xFF	Y	Y	Y	Y

Table 6.2-27 – MIFARE Plus EV1/EV2/SE/X SL3 (Security Level 3) Commands

Command	Length	Field Value	EV1	EV2	SE	X
MIFARE Plus Authenticate commands						
First Authenticate (part1 and part2)	3	First Authenticate Byte 0 = 0x70 Byte 1-2 = Key Number of the key to be authenticated (MSB first). See NXP doc ds206234, table 113. Byte 3 = MIFARE Plus AES_Key# • 0x01 = AES_Key1 = 16 bytes value stored in **Property 1.2.1.1.4.5 MIFARE Plus AES_Key1.** • 0x02 = AES_Key2 = 16 bytes value stored in **Property 1.2.1.1.4.6 MIFARE Plus AES_Key2.** • 0x03 = AES_Key3 = 16 bytes value stored in **Property 1.2.1.1.4.7 MIFARE Plus AES_Key3.** • 0x04 = AES_Key4 = 16 bytes values stored in **Property 1.2.1.1.4.8 MIFARE Plus AES_Key4.** • 0x05 = AES_Key5 = 16 bytes values stored in **Property 1.2.1.1.4.9 MIFARE Plus AES_Key5.** • 0x06 = AES_Key6 = 16 bytes values stored in **Property 1.2.1.1.4.A MIFARE Plus AES_Key6.**	Y	Y	Y	Y

Command	Length	Field Value	EV1	EV2	SE	X
Following Authenticate (part1 and part2)	3	Following Authenticate Byte 0 = 0x76 Byte 1-2 = Key Number of the key to be authenticated (MSB first). See NXP doc ds206234, table 113. Byte 3 = MIFARE Plus AES_Key# • 0x01 = AES_Key1 = 16 bytes value stored in **Property 1.2.1.1.4.5 MIFARE Plus AES_Key1.** • 0x02 = AES_Key2 = 16 bytes value stored in **Property 1.2.1.1.4.6 MIFARE Plus AES_Key2.** • 0x03 = AES_Key3 = 16 bytes value stored in **Property 1.2.1.1.4.7 MIFARE Plus AES_Key3.** • 0x04 = AES_Key4 = 16 bytes values stored in **Property 1.2.1.1.4.8 MIFARE Plus AES_Key4.** • 0x05 = AES_Key5 = 16 bytes values stored in **Property 1.2.1.1.4.9 MIFARE Plus AES_Key5.** • 0x06 = AES_Key6 = 16 bytes values stored in **Property 1.2.1.1.4.A MIFARE Plus AES_Key6.**	Y	Y	Y	Y
ResetAuth	1	Reset the authentication Byte 0 = 0x78	Y	Y	Y	Y
READ commands						
Read	4	Reading encrypted, no MAC on response, MAC on command. This command offers the possibility to read the data from one or multiple blocks in an encrypted way. A MAC is only used on the command sent to the PICC, no MAC is attached to the response. Byte 0 = 0x30 Byte 1-2 = Block number of the 1st block to be read (MSB first). See NXP doc ds206234, table 113. Byte 3 = 0x01 – 0x0F = Number of blocks to be read. Sector Trailers do not count if Byte 3 > 1. Use Byte 3 = 1 for reading Sector Trailer.	Y	Y	Y	Y
Read MACed	4	Reading encrypted, MAC on response, MAC on Command. This command offers the possibility to read the data from one or multiple blocks in an encrypted way. A MAC is used on the command sent to the PICC and on the response received. Byte 0 = 0x31 Byte 1-2 = Block number of the 1st block to be read (MSB first). See NXP doc ds206234, table 113. Byte 3 = 0x01 – 0x0F = Number of blocks to be read. Sector Trailers do not count if Byte 3 > 1. Use Byte 3 = 1 for reading Sector Trailer.	Y	Y	Y	Y

Command	Length	Field Value	EV1	EV2	SE	X
Read Plain	4	Reading in plain, no MAC on response, MAC on command. This command offers the possibility to read the data in plain from one or multiple blocks. A MAC is used on the command and not on the response. Byte 0 = 0x32 Byte 1-2 = Block number of the 1st block to be read (MSB first). See NXP doc ds206234, table 113. Byte 3 = 0x01 – 0x0F = Number of blocks to be read. Sector Trailers do not count if Byte 3 > 1. Use Byte 3 = 1 for reading Sector Trailer.	Y	Y	Y	Y
Read Plain MACed	4	Reading in plain, MAC on response, MAC on command. This command offers the possibility to read the data in plain from one or multiple blocks. A MAC is used on the command and the response. Byte 0 = 0x33 Byte 1-2 = Block number of the 1st block to be read (MSB first). See NXP doc ds206234, table 113. Byte 3 = 0x01 – 0x0F = Number of blocks to be read. Sector Trailers do not count if Byte 3 > 1. Use Byte 3 = 1 for reading Sector Trailer.	Y	Y	Y	Y
Read UnMACed	4	Reading encrypted, no MAC on response, no MAC on command This command offers the possibility to read the data from one or multiple blocks in an encrypted way. By default, Read with MAC on command is required. To Read with no MAC on command, needs to modify the card MFP Configuration Block. Byte 0 = 0x34 Byte 1-2 = Block number of the 1st block to be read (MSB first). See NXP doc ds206234, table 113. Byte 3 = 0x01 – 0x0F = Number of blocks to be read. Sector Trailers do not count if Byte 3 > 1. Use Byte 3 = 1 for reading Sector Trailer.	Y	Y	Y	Y
Read UnMACed, Response MACed	4	Reading encrypted, MAC on response, no MAC on command This command offers the possibility to read the data from one or multiple blocks in an encrypted way. A MAC is used only on the response received. By default, Read with MAC on command is required. To Read with no MAC on command, needs to modify the card MFP Configuration Block. Byte 0 = 0x35 Byte 1-2 = Block number of the 1st block to be read (MSB first). See NXP doc ds206234, table 113. Byte 3 = 0x01 – 0x0F = Number of blocks to be read. Sector Trailers do not count if Byte 3 > 1. Use Byte 3 = 1 for reading Sector Trailer.	Y	Y	Y	Y

Command	Length	Field Value	EV1	EV2	SE	X
Read Plain UnMACed	4	Reading in plain, no MAC on response, no MAC on command. This command offers the possibility to read the data in plain from one or multiple blocks. AMAC is not used on the response and not on the command. By default, Read with MAC on command is required. To Read with no MAC on command, needs to modify the card MFP Configuration Block. Byte 0 = 0x36 Byte 1-2 = Block number of the 1st block to be read (MSB first). See NXP doc ds206234, table 113. Byte 3 = 0x01 – 0x0F = Number of blocks to be read. Sector Trailers do not count if Byte 3 > 1. Use Byte 3 = 1 for reading Sector Trailer.	Y	Y	Y	Y
Read Plain UnMACed, Response MACed	4	Reading in plain, MAC on response, no MAC on command This command offers the possibility to read the data in plain from one or multiple blocks. A MAC is used on the response and not on the command. By default, Read with MAC on command is required. To Read with no MAC on command, needs to modify the card MFP Configuration Block. Byte 0 = 0x37 Byte 1-2 = Block number of the 1st block to be read (MSB first). See NXP doc ds206234, table 113. Byte 3 = 0x01 – 0x0F = Number of blocks to be read. Sector Trailers do not count if Byte 3 > 1. Use Byte 3 = 1 for reading Sector Trailer.	Y	Y	Y	Y
WRITE commands			Y	Y	Y	Y
Write	20/36/52	Writing encrypted, no MAC on response, MAC on Command. This command offers the possibility to write the data to up to three blocks in an encrypted way. MAC is only used on the command sent to the PICC. Byte 0 = 0xA0 Byte 1-2 = Block number of the 1st to be written block (MSB first). See NXP doc ds206234, table 113. Byte 3 = 0x01/0x02/0x03 = number of blocks (16 byte) of the data to be written Byte 4 – n = Data to be written, equal to number of blocks * 16.	Y	Y	Y	Y

Command	Length	Field Value	EV1	EV2	SE	X
Write MACed	20/36/52	Writing encrypted, MAC on response, MAC on command. This command offers the possibility to write the data to up to three blocks in an encrypted way. A MAC is used on the command sent to the PICC and on the response received from the PICC. Byte 0 = 0xA1 Byte 1-2 = Block number of the 1st to be written block (MSB first). See NXP doc ds206234, table 113. Byte 3 = 0x01/0x02/0x03 = number of blocks (16 byte) of the data to be written Byte 4 – n = Data to be written, equal to number of blocks * 16.	Y	Y	Y	Y
Write Plain	20/36/52	Writing in plain, no MAC on response, MAC on command This command offers the possibility to write the data to up to three blocks in plain. A MAC is only used on the command sent to the PICC. Byte 0 = 0xA2 Byte 1-2 = Block number of the 1st to be written block (MSB first). See NXP doc ds206234, table 113. Byte 3 = 0x01/0x02/0x03 = number of blocks (16 byte) of the data to be written Byte 4 – n = Data to be written, equal to number of blocks * 16.	Y	Y	Y	Y
Write Plain MACed	20/36/52	Writing in plain, MAC on response, MAC on command. This command offers the possibility to write the data to up to three blocks in plain. A MAC is used on the command sent to the PICC as well as on the response from the PICC Byte 0 = 0xA3 Byte 1-2 = Block number of the 1st to be written block (MSB first). See NXP doc ds206234, table 113. Byte 3 = 0x01/0x02/0x03 = number of blocks (16 byte) of the data to be written Byte 4 – n = Data to be written, equal to number of blocks * 16.	Y	Y	Y	Y
VALUE operations						

Command	Length	Field Value	EV1	EV2	SE	X
Increment	7	Increment encrypted, no MAC on response, MAC on command This command offers the possibility to increment a value block where the command is secured by a MAC calculated, but not on the response. Byte 0 = 0xB0 Byte 1-2 = Source Block number. Block Number of the block to be incremented (MSB first). See NXP doc ds206234, table 113. Byte 3-6 = The 4 bytes value to be incremented in LSB order. For example, if the value to be incremented by 1, then the value will be 0x01 00 00 00	Y	Y	Y	Y
Increment MACed	7	Increment encrypted, MAC on response, MAC on command. This command offers the possibility to increment a value block where the command is secured by a MAC calculated, but not on the response. Byte 0 = 0xB1 Byte 1-2 = Source Block number. Block Number of the block to be incremented (MSB first). See NXP doc ds206234, table 113. Byte 3-6 = The 4 bytes value to be incremented in LSB order. For example, if the value to be incremented by 1, then the value will be 0x01 00 00 00	Y	Y	Y	Y
Decrement	7	Decrement encrypted, no MAC on response, MAC on command. This command offers the possibility to decrement a value block where the command is secured by a MAC calculated, but not on the response. Byte 0 = 0xB2 Byte 1-2 = Source Block number. Block Number of the block to be decremented (MSB first). See NXP doc ds206234, table 113. Byte 3-6 = The 4 bytes value to be decremented in LSB order. For example, if the value to be decremented by 1, then the value will be 0x01 00 00 00	Y	Y	Y	Y

Command	Length	Field Value	EV1	EV2	SE	X
Decrement MACed	7	Decrement encrypted, MAC on response, MAC on command This command offers the possibility to decrement a value block where the command is secured by a MAC calculated, as well as on the response. Byte 0 = 0xB3 Byte 1-2 = Source Block number. Block Number of the block to be decremented (MSB first). See NXP doc ds206234, table 113. Byte 3-6 = The 4 bytes value to be decremented in LSB order. For example, if the value to be decremented by 1, then the value will be 0x01 00 00 00	Y	Y	Y	Y
Transfer	3	Transfer, no MAC on response, MAC on command. The Transfer command stores the content of the Transfer Buffer to the specified address. The Transfer command can be applied to any block. The Transfer command can only be executed after an Increment, Decrement, IncrementTransfer, DecrementTransfer or Restore command has been successfully executed since the latest authentication. The command is secured by a MAC on a command. No MAC is calculated on the response. Byte 0 = 0xB4 Byte 1-2 = Destination Block number, whose content is to be replaced by the content of the Transfer Buffer (MSB first). See NXP doc ds206234, table 113.	Y	Y	Y	Y
Transfer MACed	3	Transfer, MAC on response, MAC on command. The Transfer command stores the content of the Transfer Buffer to the specified address. The Transfer command can be applied to any block. The Transfer command can only be executed after an Increment, Decrement, IncrementTransfer, DecrementTransfer or Restore command has been successfully executed since the latest authentication. The command is secured by a MAC on a command. A MAC is calculated on the response. Byte 0 = 0xB5 Byte 1-2 = Destination Block number, whose content is to be replaced by the content of the Transfer Buffer (MSB first). See NXP doc ds206234, table 113.	Y	Y	Y	Y

Command	Length	Field Value	EV1	EV2	SE	X
Increment Transfer	9	Increment Transfer encrypted, no MAC on response, MAC on Command. This command offers the possibility to make a combined increment and transfer within one command on a value block where the command is secured by a MAC calculated, no MAC on the response. The command updates the Transfer Buffer in the same way as if a separate Increment and Transfer commands were given Byte 0 = 0xB6 Byte 1-2 = Source Block number. Block Number of the block to be incremented. Byte 3-4 = Destination Block number, whose content is to be replaced by the content of the Transfer Buffer. Byte 5-8 = The 4 bytes value to be incremented in LSB order. For example, if the value to be incremented by 1, then the value will be 0x01 00 00 00	Y	Y	Y	Y
Increment Transfer MACed	9	Increment Transfer encrypted, MAC on response, MAC on command. This command offers the possibility to make a combined increment and transfer within one command on a value block where the command is secured by a MAC calculated, and as well as a MAC on the response. The command updates the Transfer Buffer in the same way as if a separate Increment and Transfer commands were given. Byte 0 = 0xB7 Byte 1-2 = Source Block number. Block Number of the block to be incremented (MSB first). Byte 3-4 = Destination Block number, whose content is to be replaced by the content of the Transfer Buffer (MSB first). Byte 5-8 = The 4 bytes value to be incremented in LSB order. For example, if the value to be incremented by 1, then the value will be 0x01 00 00 00.	Y	Y	Y	Y

Command	Length	Field Value	EV1	EV2	SE	X
Decrement Transfer	9	Decrement Transfer encrypted, no MAC on response, MAC on command This command offers the possibility to make a combined decrement and transfer within one command on a value block where the command is secured by a MAC, but no MAC on the response. The command updates the Transfer Buffer in the same way as if a separate Decrement and Transfer commands were given. Byte 0 = 0xB8 Byte 1-2 = Source Block number. Block Number of the block to be decremented (MSB first). Byte 3-4 = Destination Block number, whose content is to be replaced by the content of the Transfer Buffer (MSB first). Byte 5-8 = The 4 bytes value to be incremented in LSB order. For example, if the value to be decremented by 1, then the value will be 0x01 00 00 00.	Y	Y	Y	Y
Decrement Transfer MACed	9	Decrement Transfer encrypted, MAC on response, MAC on Command This command offers the possibility to make a combined decrement and transfer within one command on a value block where both the command and the response are secured by a MAC. The command updates the Transfer Buffer in the same way as if a separate Decrement and Transfer commands were given. Byte 0 = 0xB9 Byte 1-2 = Source Block number. Block Number of the block to be decremented (MSB first). Byte 3-4 = Destination Block number, whose content is to be replaced by the content of the Transfer Buffer (MSB first). Byte 5-8 = The 4 bytes value to be incremented in LSB order. For example, if the value to be decremented by 1, then the value will be 0x01 00 00 00.	Y	Y	Y	Y
Restore	3	Restore encrypted, no MAC on response, MAC on command The Restore command copies the Content found in the Value Block at the given address to the Transfer Buffer. The Restore command can only be applied to value blocks. The command is secured by a MAC on a command, no MAC is calculated on the response. Byte 0 = 0xC2 Byte 1-2 = Source Block number. Block Number of the block which content is to be copied to the Transfer Buffer.	Y	Y	Y	Y

Command	Length	Field Value	EV1	EV2	SE	X
Restore MACed	3	Restore encrypted, MAC on response, MAC on command. The Restore command copies the Content found in the Value Block at the given address to the Transfer Buffer. The Restore command can only be performed to value blocks. The command is secured by a MAC on a command and a MAC is calculated on the response. Byte 0 = 0xC3 Byte 1-2 = Source Block number. Block Number of the block which content is to be copied to the Transfer Buffer.	Y	Y	Y	Y
Others						
GET_VERSION	1	The GET_VERSION command is used to retrieve manufacturing related data of the MIFARE Plus EV1/EV2 cards Byte 0 = 0x60	Y	Y	N	N
READ_SIG	2	The READ_SIG command returns an IC-specific, 48-byte ECC originality check signature of MIFARE Plus EV1/EV2 cards. Byte 0 = 0x3C Byte 1 = 0x00, RFU	Y	Y	N	N
CANCEL	1	This command is used to terminate the pass-through command session. Byte 0 = 0xFF	Y	Y	Y	Y

Table 6.2-28 - Response Data for Command 0x1103 – Pass Through Command for MIFARE Plus, Type 2

Tag	Len	Value / Description	Typ	Req	Default
Beginning of any wrappers, at minimum including Response Message found on page **33**					
1103 = **Command 0x1103 – Pass Through Command for MIFARE Plus, Type 2**					
81	01	Tag Response Code 0x00 = Success 0x01 = Failed	B	R	N/A
82	Var	Encryption Control If encrypted, see **Table 6.2-9 - Payload for Encrypted NFC/MIFARE** Data If unencrypted see **Table 6.2-10 – Unencrypted NFC/MIFARE** Data	B	O	N/A
End of any wrappers, at minimum including Response Message found on page **33**					

If the request started successfully, the Request Status in the message wrapper is **OK, Started / Running, All good / requested operation was successful**.

Table 6.2-29 - Request Example (Get Version)

Example (Hex)
AA 00 81 04 01 DA 11 03 84 0B 11 03 81 01 60 82 01 00 83 01 00

Table 6.2-30 - Response Example (Get Version)

Example (Hex)
AA 00 81 04 82 DA 11 03 82 04 01 00 00 00 84 28 11 03 81 01 00 82 21 FC 1F DF 7A 1C 04 02 01 11 00 16 04 04 02 01 01 01 16 04 04 4D 59 5A 3E 18 90 CF 8D 15 61 51 21 23

6.2.4.1 Encrypted Data Format

Table 6.2-31 - Payload for Encrypted NFC/MIFARE Data

Tag	Len	Value / Description	Typ	Req	Default
/DFDF59	var	Encrypted Data Primitive Decrypt the value of this TLV data object using the algorithm and variant specified in the **Encrypted Data KSN** parameter and the **Encrypted Data Encryption Type** parameter to read its contents. The format of the decrypted data is shown in **Table 7.4-6**.	B	R	
/DFDF50	var	Encrypted Data KSN	B	R	
/DFDF51	01	Encrypted Data Encryption Type See section **4.4 Encryption Type** for a list of valid values.	B	R	
End of Notification Message found on page **38**					

Table 6.2-32 – Unencrypted NFC/MIFARE Data

Tag	Len	Value / Description	Typ	Req	Default
FC	var	NFC/MIFARE Data Container	T	R	
/DF7A	var	NFC/MIFARE Data	B	O	

6.3 Command Group 0x18nn - User Interface

6.3.1 Command 0x1802 - Report Cardholder Selection

The host uses this command to provide a cardholder selection to the device when the device itself does not have a display or inputs to prompt the cardholder for a selection.

The sequence of events is as follows:

1) The host has already invoked **Command 0x1001 - Start Transaction** and the transaction is still in process.
2) During the transaction, if the device does not have a display or touchscreen but needs to show information to the cardholder or needs the cardholder to make a selection, it sends the host

Notification 0x1803 - User Interface Host Action Request to report Display / Cardholder Selection and supporting information.

3) The host uses its user interface to request a selection from the cardholder based on the information and selectable items provided by the notification message.
4) The host sends the user selection to the device by sending the command in the format below.

Table 6.3-1 - Request Data for Command 0x1802 - Report Cardholder Selection

Tag	Len	Value / Description	Typ	Req	Default
Beginning of any wrappers, at minimum including **Request Message** found on page **32**					
1802 = **Command 0x1802 - Report Cardholder Selection**					
81	01	Cardholder Selection Request Status • 0x00 = Cardholder Selection Request completed, see Selection Result parameter. • 0x01 = Cardholder Selection Request canceled by cardholder, Transaction Aborted. • 0x02 = Cardholder Selection Request timed out, Transaction Aborted.	B	R	
82	01	Selection Result Menu item index the cardholder selected. If the cardholder made no selection or the operation terminated abnormally, the device does not include this parameter.	B	O	
End of any wrappers, at minimum including **Request Message** found on page **32**					

Table 6.3-2 - Response Data for Command 0x1802 - Report Cardholder Selection

Tag	Len	Value / Description	Typ	Req	Default
Beginning of any wrappers, at minimum including **Response Message** found on page **33**					
1802 = **Command 0x1802 - Report Cardholder Selection**					
No parameters.					
End of any wrappers, at minimum including **Response Message** found on page **33**					

If the request started successfully, the Request Status in the message wrapper is **OK, Started / Running, All good / requested operation was successful**.

Table 6.3-3 - Request Example

Example (Hex)
AA 00 81 04 01 00 18 02 84 08 18 02 81 01 00 82 01 00

Table 6.3-4 - Response Example

Example (Hex)
AA 00 81 04 82 00 18 02 82 04 01 00 00 00

6.3.2 Command 0x1804 - Read Barcode (BCR Only)

The host uses this command to direct the device to arm or disarm the barcode reader for reading a barcode outside the scope of a transaction. This is an immediate directive. To read barcodes within the scope of a transaction, use **Command 0x1001 - Start Transaction** and its barcode reader parameters instead.

The sequence of events is as follows:

1) The host ensures the device is not currently running another command, for example, that it is not running a transaction using **Command 0x1001 - Start Transaction**.
2) The host composes a command request in the format below and sends it to the device.
3) If the device has a display, it shows a prompt **SCAN BARCODE**.
4) The device enables the barcode reader.
   a) If the **Timeout** parameter is set to **Infinite**:
      i) The device returns a command response message with Response Status, Operation Status Summary byte set to 0x00 (OK, Done) after which the host is free to send further commands.
      ii) The host may end the barcode reading session by calling this command again with the **Enable** parameter set to **Disable**.
   b) If the **Timeout** parameter is set to a value other than **Infinite**:
      i) The device returns a command response message with its Response Status, Operation Status Summary byte set to 0x01 (OK, Started / Running).
      ii) While the host is waiting for the timeout to expire, it should not send any commands to the device, because the device is busy processing the current command.
      iii) After the device reads a barcode or the timeout period expires, the device sends
      iv) **Notification 0x1805 - User Interface Operation** Complete to report Barcode Reader / Read Barcode Result and additional supporting information.

Table 6.3-5 - Request Data for Command 0x1804 - Read Barcode (BCR Only)

Tag	Len	Value / Description	Typ	Req	Default
Beginning of any wrappers, at minimum including **Request Message** found on page **32**					
1804 = **Command 0x1804 - Read Barcode (BCR Only)**					
81	01	Enable • 0x00 = Disable. The device disables the barcode reader. In this case, the device ignores all other parameters. • 0x01 = Enable. The device enables the barcode reader.	B	R	0x00
82	01	Timeout • 0x00 = Infinite. The device leaves the barcode reader enabled until it reads a barcode, or until the host sends this command again to disable the barcode reader. • All other values = Timeout in seconds for the device to leave the barcode reader enabled without reading a barcode.	B	O	0x00
83	01	Encrypt Barcode Data • 0x00 = Do Not Encrypt. The device does not encrypt the barcode data when it sends • **Notification 0x1805 - User Interface Operation** Complete.	B	O	0x00

Tag	Len	Value / Description	Typ	Req	Default
		• 0x01 = Encrypt. The device encrypts the barcode data when it sends • **Notification 0x1805 - User Interface Operation** Complete.			
End of any wrappers, at minimum including **Request Message** found on page **32**					

Table 6.3-6 - Response Data for Command 0x1804 - Read Barcode (BCR Only)

Tag	Len	Value / Description	Typ	Req	Default
Beginning of any wrappers, at minimum including **Response Message** found on page **33**					
1804 = **Command 0x1804 - Read Barcode (BCR Only)**					
No parameters.					
End of any wrappers, at minimum including **Response Message** found on page **33**					

Table 6.3-7 - Request Example

Example (Hex)
AA00 810401031804 840B 1804 810101 82010F 830101

Table 6.3-8 - Response Example

Example (Hex)
AA00 810482031804 820401000000

6.3.3 Command 0x1805 - Buzzer

The host uses this command to start a buzzer for playing a sequence of tones. Each sequence can have a minimum of 1 to maximum of 10 tones.

The sequence of events is as follows:

1) The host ensures the device is not currently running another command, for example, that it is not running a transaction using **Command 0x1001 - Start Transaction**.
2) The host composes a command request in the format below and sends it to the device.
3) The device plays a specific tone sequence as the command specified. After finish, the device sends
4) **Notification 0x1805 - User Interface Operation** Complete to report Buzzer/Buzzer Result.

   The host should wait for

   Notification 0x1805 - User Interface Operation Complete before sending another command.

   If the buzzer is current playing a sequence of tones and any transaction that uses the buzzer to make a sound is started, the device will stop the buzzer for that transaction to take over.

Table 6.3-9 - Request Data for Command 0x1805 - Buzzer

Tag	Len	Value / Description	Typ	Req	Default
Beginning of any wrappers, at minimum including **Request Message** found on page **32**					
1805 = **Command 0x1805 - Buzzer**					

Tag	Len	Value / Description	Typ	Req	Default
81	N*4	N = Number of tones • 0x01 – Min (1 tone) • 0x0A – Max (10 tones) 4 = 4 bytes data parameter for each tone in the sequence Byte0-Byte1 – Frequency in units of 1 Hz • 0x0000..0x0031 (< 50 Hz, Silent) • 0x0032 - Min (50 Hz) • 0x0FA0 - Max (4000 Hz) • 0x0FA1..0xFFFF (> 4000 Hz, Error) Byte 2-Byte3 – Duration of tone in units of 1 millisecond • 0x0001 – Min (1 ms) • 0xFFFF – Max (65535 ms)	B	R	
End of any wrappers, at minimum including **Request Message** found on page **32**					

Table 6.3-10 - Response Data for Command 0x1805 - Buzzer

Tag	Len	Value / Description	Typ	Req	Default
Beginning of any wrappers, at minimum including **Response Message** found on page **33**					
1805 = **Command 0x1805 - Buzzer**					
No parameters.					
End of any wrappers, at minimum including **Response Message** found on page **33**					

Table 6.3-11 - Request Example for a sequence of 5 tones

Example (Hex)
AA00 810401031805 8418 1805 8114 00C8 01F4 0190 01F4 0258 01F4 0190 01F4 00C8 01F4

Table 6.3-12 - Response Example

Example (Hex)
AA00 810482031805 820401000000

6.3.4 Command 0x1840 – Card Emulation

Card emulation is initiated by receiving a 0x1840 command from the host. The device will prepare card emulation with the parameters provided in the command and start card emulation.

The sequence of events is as follows:

1) The host ensures the device is not currently running another command, for example, that it is not running a transaction or PIN entry.
2) The host composes a command request in the format below and sends it to the device.
3) The device receives the command and verifies that the parameters are valid and the device is in a state that allows the execution of card emulation.

4) If the device has a display, a prompt will be displayed asking the customer to tap their phone to the device.
5) If the timeout parameter is not included or set to 0x00, then there is no timeout. Or, if the timeout parameter is set to a specific number of seconds, the device returns a command response message with its Operation Status Summary byte set to 0x01 (OK, Started / Running).
6) The host may issue a 0x1840 command with Tag 0x81 set to 0x00 to cancel the execution of card emulation.
7) After the timeout expires, host cancel or the card is read, the device sends a 0x1805 notification to inform the host.

Table 6.3-13 - Request Data for Command 0x1840 – Card Emulation

Tag	Len	Value / Description	Req	Default
/81	01	Start/Cancel • 0x00 = Cancel (See the example of 0x1840 cancel command below) • 0x01 = Start	R	
/82	01	Timeout in seconds • 0x00 = No timeout • 0x01 to 0xFF = 1 to 255 seconds	O	0x00
/83	<= 254	URL URL to use as card data. Required when starting card emulation. Optional and ignored if canceling emulation. Example: https://www.magtek.com/	O/R	

Table 6.3-14 - Request Example

Example (Hex)
AA 00 81 04 01 01 18 40 84 21 18 40 81 01 01 82 01 00 83 17 68 74 74 70 73 3A 2F 2F 77 77 77 2E 6D 61 67 74 65 6B 2E 63 6F 6D 2F

Table 6.3-15 - Response Example

Example (Hex)
AA 00 81 04 82 01 18 40 82 04 01 00 00 00

Table 6.3-16 - Example of 0x1840 Cancel Command

Example (Hex)
AA 00 81 04 01 01 18 40 84 05 18 40 81 01 00

6.4 Command Group 0x1Fnn - Device Control

6.4.1 Command 0x1F01 - Reset Device

The host uses this command to reset the device.

The sequence of events is as follows:

1) The host constructs the command request for **Command 0x1F01 - Reset Device** in the format below.
2) The host sends the command request to the device.
3) The device sends a response in the format below to the host.
4) The device starts an automatic reset within 500ms.

Table 6.4-1 - Request Data for Command 0x1F01 - Reset Device

Tag	Len	Value / Description	Typ	Req	Default
Beginning of any wrappers, at minimum including **Request Message** found on page **32**					
1F01 = **Command 0x1F01 - Reset Device**					
81	01	Power Off Option • 0x00 = Reset • 0x01 = Power Off Power off only works while a device is running on its battery. If a device is powered off while it is powered by USB, the device will immediately turn back on.	B	O	0x00
End of any wrappers, at minimum including **Request Message** found on page **32**					

Table 6.4-2 - Response Data for Command 0x1F01 - Reset Device

Tag	Len	Value / Description	Typ	Req	Default
Beginning of any wrappers, at minimum including **Response Message** found on page **33**					
1F01 = **Command 0x1F01 - Reset Device**					
No parameters.					
End of any wrappers, at minimum including **Response Message** found on page **33**					

Table 6.4-3 - Request Example

Example (Hex)
AA 00 81 04 01 12 1F 01 84 02 1F 01

Table 6.4-4 - Response Example

Example (Hex)
AA 00 81 04 82 12 1F 01 82 04 00 00 00 00

6.4.2 Command 0x1F02 - Set Notification Subscriptions

The host uses this command to specify which notifications the device should send on each of its available interfaces. By default, the device sends notifications to the host on all interfaces.

The sequence of events is as follows:

1) The host constructs the command request in the format below.
2) The host sends the command request to the device.
3) The device sends a response in the format below to the host.
4) The device immediately begins routing notifications per the request.
5) If the device restarts or loses power, the device resets its notification subscriptions to defaults, and the host must call this command again to change them.

Table 6.4-5 - Request Data for Command 0x1F02 - Set Notification Subscriptions

Tag	Len	Value / Description	Typ	Req	Default
Beginning of any wrappers, at minimum including **Request Message** found on page **32**					
1F02 = **Command 0x1F02 - Set Notification Subscriptions**					
81	01	Subscribe • 0x00 = Unsubscribe • 0x01 = Subscribe	B	O	0x01
82	01	Notifications Affected • 0x00 = Only subscribe or unsubscribe to notification messages in the Notification Message ID List parameter • 0x01 = Subscribe or unsubscribe to all notifications	B	O	0x01
83	var	Notification Message ID List List of two-byte Notification Message IDs (MSB first) from section **7 Notifications** to be subscribed / unsubscribed by this command. For example, to subscribe to **Notification 0x0105 - Transaction Operation Complete** on the interface being used to send this command, the host would include 0x0105 as two bytes in the list. The device ignores any Notification Message IDs in the list that do not exist.	B	O	Null
A4	var	Interfaces List of interfaces this command should change the subscription settings for. If the host does not specify any interfaces here, the command applies only to the interface the host is using to send the command.	B	O	Null
/81	00	Apply changes to the USB interface		O	
/82	00	Apply changes to the WLAN interface		O	
/84	00	Apply changes to the UART interface		O	
End of any wrappers, at minimum including **Request Message** found on page **32**					

Table 6.4-6 - Response Data for Command 0x1F02 - Set Notification Subscriptions

Tag	Len	Value / Description	Typ	Req	Default
Beginning of any wrappers, at minimum including **Response Message** found on page **33**					
1F02 = **Command 0x1F02 - Set Notification Subscriptions**					
No parameters.					

Tag	Len	Value / Description	Typ	Req	Default
End of any wrappers, at minimum including **Response Message** found on page **33**					

Table 6.4-7 - Request Example

Example (Hex)
AA00 810401551F02 8402 1F02

Table 6.4-8 - Response Example

Example (Hex)
AA00 810482551F02 820400000000 8402 1F02

6.5 Command Group 0xD1nn - Settings and Information

6.5.1 Command 0xD101 - Get Property

The host uses this command to get information about the device or its configuration / settings.

Each data element representing device information or device configuration is part of a tree of values and is uniquely identified by an Object Identifier (also known as an Object ID or OID) as defined in ***ITU-T X.660 | ISO/IEC 9834-1***, which can be found by searching for **X.660** in the publications on www.itu.int. This document refers to these data elements collectively as **Properties**. The list of all properties and their corresponding OIDs and other characteristics is provided in section **8 Configuration**.

This command can be used in multiple ways. For simplicity, this document describes one possible way that does not require detailed knowledge of the ***X.660*** specification.

To get a property, the sequence of events is as follows:

1) The host determines which property or tree branch of properties it wants to get from the device (see section **8 Configuration**).
2) The host composes the command request in the format below, and sends it to the device.
3) The device sends a response. If the request succeeded, the response includes the value(s) of the requested property or properties. If it did not succeed, the device returns a failure response with no command-specific parameters.

Table 6.5-1 - Request Data for Command 0xD101 - Get Property

Tag	Len	Value / Description	Typ	Req	Default
Beginning of any wrappers, at minimum including **Request Message** found on page **32**					
D101 = **Command 0xD101 - Get Property**					
81	var	Company ID This value is the root of the “long form” of the Property OID, and is the same for all MagTek devices. Leave this parameter empty and use the default.	B	O	2B 06 01 04 01 F6 09
82	03	Device Family ID	B	O	Product dependent

Tag	Len	Value / Description	Typ	Req	Default
		This value is the second portion of the "long form" of the Property OID, and is the same for all similar MagTek devices within the same product family. Unless you have a specific use case that uses this parameter, leave this parameter empty and use the default otherwise your software may not work with multiple products. Byte 1 Platform • 0x02 = Apollo Platform Byte 2 Product • 0x01 = DynaFlex, 0x02 = DynaProx, 0x03 = DynaFlex II PED, 0x04 = DynaFlex II, 0x05 = DynaFlex II Go Byte 3 Device Variant • 0x00 = Standard			
85	01	Property Type This parameter contains the first number of the Property OID as documented in section **8 Configuration**. • 0x01 = Device Settings • 0x02 = Device Information	B	R	
87	var	Property OID Tree Prefix This optional parameter contains subsequent numbers of the Property's OID as documented in section **8 Configuration**, but can not include the final number. This can also be populated with fewer numbers from the OID, in which case the remaining numbers of the OID of the Property or set of Properties you wish to retrieve must be included in the **Property OID Remainder**. For simplicity, populate this with the 2nd through the second-to-last number in the property's OID.	B	O	Null
89	var	Property OID Remainder This contains the remaining numbers of the Property's OID, BER TLV encoded per ***X.660*** section ***8 Basic encoding rules***. For details about TLV encoding an OID. To request a set of properties in a branch of the Property OID structure, the host should pass a partial Property OID, and the device returns the value of all properties from the specified tree level downward. For simplicity, include all numbers except the final number of the property's OID in **Property Type** and **Property OID Tree Prefix**, and include the final number of the OID OR 0xC0 here, then append constant byte 0x00. These two bytes represent a single empty BER TLV primitive data object.	B	R	

Tag	Len	Value / Description	Typ	Req	Default
End of any wrappers, at minimum including **Request Message** found on page **32**					

Table 6.5-2 - Response Data for Command 0xD101 - Get Property

Tag	Len	Value / Description	Typ	Req	Default
Beginning of any wrappers, at minimum including **Response Message** found on page **33**					
D101 = **Command 0xD101 - Get Property**					
81	var	Company ID This contains the Company ID the host included in the request message. If this parameter is not included in the request, the response does not include it.	B	O	N/A
82	03	Device Family ID This contains the Device Family ID the host included in the request message. If this parameter is not included in the request, the response does not include it.	B	O	N/A
85	01	Property Type This contains the Property Type the host included in the request message.	B	R	N/A
87	var	Property OID Tree Prefix This contains the Property OID Tree Prefix the host included in the request message. If this parameter is not included in the request, the response does not include it.	B	O	N/A
89	var	Property OID Remainder This contains the same TLV-encoded portion of the OID the host included in the Property OID Remainder of the request message, with leaf nodes populated with actual values. If the host requested a set of properties in a branch of the Property OID structure, this contains the set of requested branches, including branch OIDs, leaf node IDs, and values. If the host follows the "for simplicity" recommendation in the request message to request a single property, it can retrieve the value of the requested property by stripping off the first few bytes, which represent the TLV-encoded last number in the OID and the length of the property's value, as follows; the remaining bytes are the value of the property: • If the second byte is 7F or less, strip off the first two bytes. • If the second byte is 81, strip off the first three bytes. • If the second byte is 82, strip off the first four bytes.	B	R	N/A
End of any wrappers, at minimum including **Response Message** found on page **33**					

If the request started successfully, the Request Status in the message wrapper is **OK, Started / Running, All good / requested operation was successful**.

Table 6.5-3 - Request Example

Example (Hex)
Get **Property 1.2.7.1.1.1 Device Reset Occurred Notification Control** using "simple" form: AA00 8104 0155D101 840F D101 8501 01 8704 02070101 8902 C100

Table 6.5-4 - Response Example

Example (Hex)
Get **Property 1.2.7.1.1.1 Device Reset Occurred Notification Control** using "simple" form: AA00 8104 8255D101 8204 00000000 84820010 D101 8501 01 8704 02070101 8903 C101 00

Table 6.5-5 - Request Example

Example (Hex)
Get **Property 1.2.7.1.1.1 Device Reset Occurred Notification Control** using longer Property OID Remainder: AA00 8104 0155D101 8411 D101 8501 01 890A E208 E706 E104 E102 C100

Table 6.5-6 - Response Example

Example (Hex)
Get **Property 1.2.7.1.1.1 Device Reset Occurred Notification Control** using longer Property OID Remainder: AA00 8104 8255D101 8204 00000000 84820012 D101 8501 01 890B E209 E707 E105 E103 C101 00

Table 6.5-7 - Request Example

Example (Hex)
Get **Property Subgroup 2.1.2.2.nn Core Firmware Information** using "simple" form: AA00 8104 0155D101 840D D101 8501 02 8702 0102 8902 C200

Table 6.5-8 - Response Example

Example (Hex)
Get **Property Subgroup 2.1.2.2.nn Core Firmware Information** using "simple" form: AA00 81 04 8255D101 82 04 00000000 84 820056 D101 85 01 02 87 02 0102 89 820049 E2 820045 E1 820004 C1 00 C2 00 E2 820039 C1 0D 44796E61466C65782050726F00 C2 13 31303030303037313833 2D41352D5043490000 C3 00 C4 0B 313030303030373138330 0 C5 00 C6 02 FF00

6.5.2 Command 0xD111 - Set Property (Unsecured)

NOTICE

Properties are stored in flash memory, which inherently has a limited number of read-write cycles before it begins to wear. For this reason, MagTek recommends setting properties as few times as possible over the lifecycle of the device.

The host uses this command to set device configuration / settings that do not require security. For setting properties that require security see **Command 0xD112 - Set Property (Secured)**.

Each data element representing device configuration is part of a tree of values and is uniquely identified by an Object Identifier (also known as an Object ID or OID) as defined in ***ITU-T X.660 | ISO/IEC 9834-1***, which can be found by searching for **X.660** in the publications on www.itu.int. This document refers to these data elements collectively as **Properties**. The list of all properties and their corresponding OIDs and other characteristics is provided in section **8 Configuration**.

This command can be used in multiple ways. For simplicity, this document describes one possible way that does not require detailed knowledge of the ***X.660*** specification.

To set a property, the sequence of events is as follows:

1) The host determines which property it wants to set and the value it wants to set in the device (see section **8 Configuration**).
2) The host composes a command request in the format below, and sends it to the device.
3) The device sends a response in the format below. If the request succeeded, the response payload is identical to the request payload. If it did not succeed, the device returns a failure response with no command-specific parameters.

Table 6.5-9 - Request Data for Command 0xD111 - Set Property (Unsecured)

Tag	Len	Value / Description	Typ	Req	Default
Beginning of any wrappers, at minimum including **Request Message** found on page **32**					
D111 = **Command 0xD111 - Set Property (Unsecured)**					
81	var	Company ID This value is the root of the "long form" of the Property OID, and is the same for all MagTek devices. Leave this parameter empty and use the default.	B	O	2B 06 01 04 01 F6 09
82	03	Device Family ID This value is the second portion of the "long form" of the Property OID, and is the same for all similar MagTek devices within the same product family. Unless you have a specific use case that uses this parameter, leave this parameter empty and use the default otherwise your software may not work with multiple products. Byte 1 Platform 0x02 = Apollo Platform Byte 2 Product • 0x01 = DynaFlex, 0x02 = DynaProx, 0x03 = DynaFlex II PED, 0x04 = DynaFlex II, 0x05 = DynaFlex II Go Byte 3 Device Variant • 0x00 = Standard	B	O	Product dependent
85	01	Property Type This parameter contains the first number of the Property OID as documented in section **8 Configuration**. • 0x01 = Device Settings • 0x02 = Device Information	B	R	
87	var	Property OID Tree Prefix This optional parameter contains subsequent numbers of the Property's OID as documented in section **8 Configuration**, but can not include the final number. For simplicity, populate this with the 2nd through the second-to-last number in the property's OID. This can also be populated with fewer numbers from the OID, in which case the remaining numbers of the OID must be included in the **Property OID Remainder**.	B	O	Null
89	var	Property OID Remainder This contains the remaining numbers of the Property's OID, BER TLV encoded per ***X.660*** section ***8 Basic encoding rules***. For details about TLV encoding an OID.	B	R	

Tag	Len	Value / Description	Typ	Req	Default
		For simplicity, include all numbers except the final number of the property's OID in **Property Type** and **Property OID Tree Prefix**, and include the final number of the OID OR 0xC0 here, then append a length corresponding one of the following, then append the value to set the property to: • If the length of the value you are setting is 0x7F or shorter, include one byte equal to the length of the value. • If the length of the value you are setting is greater than 0x7F but less than 0xFFFF, append 82, then two bytes equal to the length of the value.			
End of any wrappers, at minimum including **Request Message** found on page **32**					

Table 6.5-10 - Response Data for Command 0xD111 - Set Property (Unsecured)

Tag	Len	Value / Description	Typ	Req	Default
Beginning of any wrappers, at minimum including **Response Message** found on page **33**					
D111 = **Command 0xD111 - Set Property (Unsecured)**					
81	var	Company ID This contains the Company ID the host included in the request message. If this parameter is not included in the request, the response does not include it.	B	O	N/A
82	03	Device Family ID This contains the Device Family ID the host included in the request message. If this parameter is not included in the request, the response does not include it.	B	O	N/A
85	01	Property Type This contains the Property Type the host included in the request message.	B	R	N/A
87	var	Property OID Tree Prefix This contains the Property OID Tree Prefix the host included in the request message. If this parameter is not included in the request, the response does not include it.	B	O	N/A
89	var	Property OID Remainder This contains the same TLV-encoded portion of the OID the host included in the Property OID Remainder of the request message.	B	R	N/A
End of any wrappers, at minimum including **Response Message** found on page **33**					

Table 6.5-11 - Request Example

Example (Hex)
Set **Property 1.2.7.1.1.1 Device Reset Occurred Notification Control** AA 00 81 04 01 55 D1 11 84 10 D1 11 85 01 01 87 04 02 07 01 01 89 03 C1 01 00

Table 6.5-12 - Response Example

Example (Hex)
Set **Property 1.2.7.1.1.1 Device Reset Occurred Notification Control** AA 00 81 04 82 55 D1 11 82 04 00 00 00 00 84 82 00 10 D1 11 85 01 01 87 04 02 07 01 01 89 03 C1 01 00

6.5.3 Command 0xD112 - Set Property (Secured)

NOTICE

Properties are stored in flash memory, which inherently has a limited number of read-write cycles before it begins to wear. For this reason, MagTek recommends setting properties as few times as possible over the lifecycle of the device.

The host uses this command to set device configuration / settings securely. Properties that require security should specify that they do in their documentation. This command can also be used for properties that do not require security.

The details, request data and response data of this command are identical to what is documented in **Command 0xD111 - Set Property (Unsecured),** however, the command must be structured and sent according to what is documented in sequence of events 1-5 of **Command 0xD811 - Start Send File to Device (Secured)**.

6.6 Command Group 0xD8nn - File Operations

6.6.1 About Files

Large blobs of data uploaded to / downloaded from the device are referred to as "files" and share a common set of commands documented here, and special message type **Data File Message**. Some file types can be sent in the File Payload fields in "raw" form (e.g. certificates and images) with metadata coming from the command request and response. Other file types require the addition of MagTek metadata included inside the File Payload blob; these are documented in the "File Type" subsections of section **4 Data Types and Shared TLV Data Objects**.

Files Types that may come from the host include:

- EMV configuration
- Firmware updates
- Public Key Infrastructure (PKI) Certificates
- User interface images and prompts
- EMV kernels
- SRED BIN tables

File Types that may come from the device include:

- Read back of the above file types
- Signature Capture
- Logs

- Certificate Requests

The commands in this section share a common list of 4-byte file types, listed in Table. File types marked as Secured = Yes must be loaded using **Command 0xD811 - Start Send File to Device (Secured)**; file types that are marked as Secured = No can be loaded using **Command 0xD812 - Start Send File to Device (Unsecured)**.

Table 6.6-1 - File Types

Description	Secured	File Type	File Type Version	File Subtype	File Instance
EMV configuration, terminal file. See file definition in section 4.8 on page 58.	Get: No Set: No	0x00	0x00	0x00	0x00
EMV configuration, processing file. See file definition in section 4.9 on page 60.	Get: No Set: No	0x00	0x00	0x01	0x00
EMV configuration, entry point file. See file definition in section 4.10 on page 62.	Get: No Set: No	0x00	0x00	0x02	0x00
EMV configuration, CA keys file. See file definition in section 0 on page 75.	Get: No Set: No	0x00	0x00	0x03	0x00
EMV configuration, Visa DRL set. Reserved for future use.	Get: No Set: No	0x00	0x00	0x04	0x00
EMV configuration, American Express DRL set. See file definition in section 4.12 on page 80.	Get: No Set: No	0x00	0x00	0x05	0x00
EMV configuration, MasterCard update conditions. Reserved for future use.	Get: No Set: No	0x00	0x00	0x06	0x00
EMV configuration, American Express update conditions. Reserved for future use.	Get: No Set: No	0x00	0x00	0x08	0x00
EMV configuration, Discover update conditions. Reserved for future use.	Get: No Set: No	0x00	0x00	0x09	0x00
EMV configuration, CA revocation list. Reserved for future use.	Get: No Set: No	0x00	0x00	0x0A	0x00
EMV configuration, exception file list. Reserved for future use.	Get: No Set: No	0x00	0x00	0x0B	0x00
EMV configuration, DPAS data storage. Reserved for future use.	Get: No Set: No	0x00	0x00	0x0C	0x00
UI configuration file.	Get: No Set: No	0x06	0x00	0x00	0x00

6.6.2 Command 0xD801 - Load Firmware File

The host uses this command to send a firmware image file, signed by MagTek, to the device as the first step in updating firmware. The host also uses this command to send a firmware image file, signed by MagTek, to the device as the first step in updating firmware. If the battery charge is five percent or less, a

response is returned indicating that the command has not been executed. See **Table 6.6-4 - Response Example for Command 0xD801**.

The sequence of events is as follows:

1) The host is assumed to have access to a binary file containing a firmware image signed by MagTek.
2) The host composes a command request in the format below, using the binary file as the **Payload**, and sends it to the device.
3) The device sends a response to the host in the format below to acknowledge it has received the request. The device will not allow the Load Firmware File command to execute if the battery charge is 5 percent or lower.
4) The device validates the request and authenticates the firmware file with the algorithm specified in the firmware file payload.

   If the upload was not successful, then go to the next step. If the upload was successful and auto-commit was disabled, then go to the next step. Else, the device will commit the image automatically. If it was successful, the device sends **Notification 0x0905 - Firmware Update Successful** to the host. If it was unsuccessful, the device sends **Notification 0x0906 - Firmware Update Failed**to the host. Commit Firmware Notification Detail Codes are used for auto-commit mode. In both cases, the device automatically resets.

Table 6.6-2 - Request Data for Command 0xD801 - Load Firmware File

Tag	Len	Value / Description	Typ	Req	Default
Beginning of any wrappers, at minimum including **Request Message** found on page **32**					
D801 = **Command 0xD801 - Load Firmware File**					
81	01	Progress Indicator Reserved for future use. Populate with 0x03.	B	R	
85	02	Image Type • 0x0000 = Boot Loader 1 image • 0x0001 = Main App image • 0x0002 = WiFi Module image • 0x0003 = BLE Module image	B	R	
86	20	Hash Checksum	B	R/O	
87	var	Payload	B	R	
88	01	Load Options 0x00 = Default mode 0x01 = Auto Commit	B	O	0x00
End of any wrappers, at minimum including **Request Message** found on page **32**					

Table 6.6-3 - Response Data for Command 0xD801 - Load Firmware File

Tag	Len	Value / Description	Typ	Req	Default
Beginning of any wrappers, at minimum including **Response Message** found on page **33**					
D801 = **Command 0xD801 - Load Firmware File**					
No parameters.					

Tag	Len	Value / Description	Typ	Req	Default
End of any wrappers, at minimum including **Response Message** found on page **33**					

Table 6.6-4 - Response Example for Command 0xD801 Battery Charge State

Example (Hex)
AA 00 81 04 82 01 D8 01 82 04 80 02 03 16

If the request started successfully, the Request Status in the message wrapper is **OK, Started / Running, All good / requested operation was successful**.

Table 6.6-5 - Request Example

Example (Hex)
AA 00 81 04 01 01 D8 01 84 83 0C 76 58 D8 01 81 01 03 85 02 00 01 86 20 DF C7 1E 09 A3 CE 8E 86 B0 F5 B6 75 BE B7 7A 0E 82 33 BF F1 8A CD 8F 38 34 B0 DB 20 D9 40 4B 28 87 83 0C 76 28 Plus 0C7628 bytes of firmware Payload, excluded here for brevity.

Table 6.6-6 - Response Example

Example (Hex)
AA 00 81 04 82 01 D8 01 82 04 00 00 00 00

6.6.3 Command 0xD811 - Start Send File to Device (Secured)

The host uses this command to start sending secured files to the device for storage or processing. It is similar to **Command 0xD812 - Start Send File to Device (Unsecured)**, but is used to send a different subset of file types that impact device security and require some form of authentication from the host. Refer to Table 6.6-1 to determine which file type requires a secure command All files require the command to be authorized via a secure wrapper. In some cases, files include additional signatures within the file structure itself. This command is paired with **Command 0xD821 - Start Get File from Device**, which the host can use to retrieve files. However, some file types are "one way only" and can not be retrieved using that command after the host sends them to the device.

The sequence of events is as follows:

1) The host uses **Command 0xE001 - Get Challenge** to establish a secure session with the device.
2) The host determines which file type it will send to the device (see section **6.6.1 About Files**), and either opens an existing file in its file system for reading, or begins constructing it.
3) The host constructs **Command 0xD811 - Start Send File to Device (Secured)** per **Table 6.6-7**.
4) The host constructs **Command 0xEEEE - Send Secured Command to Device** using the previously constructed command as the payload, and sends that command to the device as a **Request Message** to start the process of uploading a file.
   a) Build the **Security Parameters Type** portion of the wrapper with:
      i) **Security Operation Type** populated with the following values:
         (1) Operation Type = **Command Authorization Using MAC**
         (2) Operation Algorithm = CMAC
         (3) Operation Cipher=AES-256
         (4) Padding = One and zeros

(5) **MAC Block Size** with any number

ii) **Key Information Type** populated with the key information gathered earlier.

5) The device sends a **Response Message** so the host knows it can begin sending the file.
6) The host sends a **Data File Message** to the device. If the device does not receive file data within a reasonable period of time, it times out and stops listening for the data file.
7) The device checks to make sure the **File ID** and the length and hash of the File Payload match with the values the host specified in this command.
8) The device repeats the same **Response Message**, this time with the **Message Reference Number** set to the same value the host used in the **Data File Message**.

Table 6.6-7 - Request Data for Command 0xD811 - Start Send File to Device (Secured)

Tag	Len	Value / Description	Typ	Req	Default
Beginning of any wrappers, at minimum including **Request Message** found on page **32**					
D811 = **Command 0xD811 - Start Send File to Device (Secured)**					
81	04	File ID from Table 6.6-1	B	R	
A2	var	File transfer properties	T	R	
/81	var	Length of File Payload This is the length of the **File Payload** parameter in the **Data File Message** the host sends to the device.	B	R	
/82	01	Hash Checksum Type • 0x04 = SHA-256	B	R	
/83	20	Hash Checksum Anticipated checksum calculated against the **File Payload**, according to the standard specified in **Hash Checksum Type**.	B	R	
A3	var	File Description The host should populate this value to help identify the file using **Command 0xD825 - Get File Info from Device**.	T	R	
/81	var	File Name Maximum length 32 bytes Reserved for future use. Leave empty.	B	O	Null
/82	var	File Label Maximum length 16 bytes Reserved for future use. Leave empty.	B	O	Null
/83	var	File Version Maximum length 7 bytes Reserved for future use. Leave empty.	B	O	Null
/84	var	File Date Maximum length 20 bytes Reserved for future use. Leave empty.	B	O	Null
87	01	Reserved for future use. Leave empty.	B	O	Null
End of any wrappers, at minimum including **Request Message** found on page **32**					

Table 6.6-8 - Response Data for Command 0xD811 - Start Send File to Device (Secured)

Tag	Len	Value / Description	Typ	Req	Default
Beginning of any wrappers, at minimum including **Response Message** found on page **33**					
D811 = **Command 0xD811 - Start Send File to Device (Secured)**					
No parameters.					
End of any wrappers, at minimum including **Response Message** found on page **33**					

If the request started successfully, the Request Status in the message wrapper is **OK, Started / Running, All good / requested operation was successful**.
Note: For additional support, please contact MagTek Support.

Table 6.6-9 - Request Example

Example (Hex)
AA 00 81 04 01 04 D8 11 84 81 8F EE EE A1 19 81 05 03 03 06 02 08 84 00 85 00 A8 0A 81 02 11 02 82 00 86 00 88 00 A9 00 82 04 FF FF FF F0 83 08 C9 65 45 F2 97 69 85 B1 84 4E D8 11 81 04 00 00 03 00 A2 2B 81 04 00 00 02 99 82 01 04 83 20 87 A4 B3 54 61 C5 CB D3 1D DC BA 9D 65 25 5A D4 6A 22 FA 51 5E FD 65 87 AF AC A8 8C 4F AF 80 9B A3 14 38 31 30 38 33 30 33 30 33 30 33 30 33 30 33 33 33 30 33 30 87 01 01 9E 10 7D E4 27 C8 A0 70 72 08 19 0A 1E 0A 3F 48 BB F1

Table 6.6-10 - Response Example

Example (Hex)
AA 00 81 04 82 0C D8 11 82 04 00 00 00 00

6.6.4 Command 0xD812 - Start Send File to Device (Unsecured)

The host uses this command to start sending unsecured files to the device for storage or processing. It is similar to **Command 0xD811 - Start Send File to Device (Secured)** but is used to send a different subset of file types that do not impact device security, Refer to Table 6.6-1 to determine which file type can use unsecure command This command is paired with **Command 0xD821 - Start Get File from Device**, which the host can use to retrieve files. However, some file types are "one way only" and can not be retrieved using that command after the host sends them to the device.

The sequence of events is as follows:

1) The host determines which file type it will send to the device (see section **6.6.1 About Files**), and either opens an existing file in its file system for reading, or begins constructing it.
2) The host constructs **Command 0xD812 - Start Send File to Device (Unsecured)** per **Table 6.6-11**.
3) The host sends that command to the device as a **Request Message** to start the process of uploading a file.
4) The device sends a **Response Message** so the host knows it can begin sending the file.
5) The device checks to make sure the **File ID** and the length and hash of the **File Payload** match with the values the host specified in this command.
6) The device repeats the same **Response Message**, this time with the **Message Reference Number** set to the same value the host used in the **Data File Message**.

Table 6.6-11 - Request Data for Command 0xD812 - Start Send File to Device (Unsecured)

Tag	Len	Value / Description	Typ	Req	Default
Beginning of any wrappers, at minimum including **Request Message** found on page **32**					
D812 = **Command 0xD812 - Start Send File to Device (Unsecured)**					
81	04	File ID from Table 6.6-1	B	R	
A2	var	File transfer properties	T	R	
/81	var	Length of File Payload This is the length of the **File Payload** parameter in the **Data File Message** the host sends to the device.	B	R	
/82	01	Hash Checksum Type • 0x04 = SHA-256	B	R	
/83	20	Hash Checksum Anticipated checksum calculated against the **File Payload**, according to the standard specified in **Hash Checksum Type**.	B	R	
A3	var	File Description The host should populate this value to help identify the file using **Command 0xD825 - Get File Info from Device**.	T	R	
/81	var	File Name Maximum length 32 bytes Reserved for future use. Leave empty.	B	O	Null
/82	var	File Label Maximum length 16 bytes Reserved for future use. Leave empty.	B	O	Null
/83	var	File Version Maximum length 7 bytes Reserved for future use. Leave empty.	B	O	Null
/84	var	File Date Maximum length 20 bytes Reserved for future use. Leave empty.	B	O	Null
87	01	Reserved for future use. Leave empty.	B	O	Null
End of any wrappers, at minimum including **Request Message** found on page **32**					

Table 6.6-12 - Response Data for Command 0xD812 - Start Send File to Device (Unsecured)

Tag	Len	Value / Description	Typ	Req	Default
Beginning of any wrappers, at minimum including **Response Message** found on page **33**					
D812 = **Command 0xD812 - Start Send File to Device (Unsecured)**					
No parameters.					
End of any wrappers, at minimum including **Response Message** found on page **33**					

If the request started successfully, the Request Status in the message wrapper is **OK, Started / Running, All good / requested operation was successful**.

Note: For additional support, please contact MagTek Support.

Table 6.6-13 - Request Example

Example (Hex)
AA 00 81 04 01 07 D8 12 84 44 D8 12 81 04 02 00 00 00 A2 2B 81 04 00 02 58 38 82 01 04 83 20 D5 B8 BF 2F 3A 15 D9 EE 1D 0D E5 8E DD 68 37 73 18 51 C7 3C 3D 79 58 2B A6 07 90 5C 2B 86 3C E5 A3 0A 81 08 30 32 30 30 30 30 30 30 87 01 01

Table 6.6-14 - Response Example

Example (Hex)
AA 00 81 04 82 07 D8 12 82 04 00 00 00 00

6.6.5 Command 0xD821 - Start Get File from Device

The host uses this command to request a file stored on the device. File types include standard files (images and certificates), MagTek custom files (configuration, firmware), and in some cases even large data blob output (such as signature capture data). In many cases, the files retrieved by this command have been sent by a host previously using **Command 0xD811 - Start Send File to Device (Secured)** or **Command 0xD812 - Start Send File to Device (Unsecured)**. In other cases, such as retrieving signature capture data, the data may originate with the device and the host uses this command to retrieve it. Such data and is not persistent, in the sense that the device does not retain it through power cycles.

The sequence of events is as follows:

1) The host composes a command request in the format below, and sends it to the device.
2) The device sends a response in the format below so the host knows it can begin listening for a file message.
3) The device sends a **Data File Message** to the host. If the host does not receive file data within a reasonable period of time, it should time out and stop listening for the data file.
4) Upon receiving the end of the **Data File Message**, the host should check to make sure the **File ID**, length, and hash of the **File Payload** in the **Data File Message** match the values the device specified in its response to ensure the file has not been tampered with.

Table 6.6-15 - Request Data for Command 0xD821 - Start Get File from Device

Tag	Len	Value / Description	Typ	Req	Default
Beginning of any wrappers, at minimum including **Request Message** found on page **32**					
D821 = **Command 0xD821 - Start Get File from Device**					
81	04	File ID from Table 6.6-1	B	R	

Tag	Len	Value / Description	Typ	Req	Default
87	01	Progress indicator behavior (Reserved for future use / Subject to change) • 0x00 = None • 0x01 = LED	B	O	Null
End of any wrappers, at minimum including **Request Message** found on page **32**					

Table 6.6-16 - Response Data for Command 0xD821 - Start Get File from Device

Tag	Len	Value / Description	Typ	Req	Default
Beginning of any wrappers, at minimum including **Response Message** found on page **33**					
D821 = **Command 0xD821 - Start Get File from Device**					
81	04	File ID from Table 6.6-1	B	R	
A2	var	File transfer properties	T	R	
/81	var	Length of File Payload This is the length of the **File Payload** parameter in the **Data File Message** the device sends to the host.	B	R	
/82	01	Hash Checksum Type • 0x04 = SHA-256	B	R	
/83	20	Hash Checksum Anticipated checksum calculated against the **File Payload**, according to the standard specified in **Hash Checksum Type**.	B	R	
A3	var	File Description The values the host populated for convenience when it sent the file to help identify the file. Not all values are required.	T	R	
/81	var	File Name Maximum length 32 bytes Reserved for future use.	B	O	Null
/82	var	File Label Maximum length 16 bytes Reserved for future use.	B	O	Null
/83	var	File Version Maximum length 7 bytes Reserved for future use.	B	O	Null
/84	var	File Date Maximum length 20 bytes Reserved for future use.	B	O	Null
End of any wrappers, at minimum including **Response Message** found on page **33**					

If the request started successfully, the Request Status in the message wrapper is **OK, Done.**

Table 6.6-17 - Request Example

Example (Hex)
AA 00 81 04 01 08 D8 21 84 0B D8 21 81 04 00 00 00 01 87 01 01

Table 6.6-18 - Response Example

Example (Hex)
AA 00 81 04 82 08 D8 21 82 04 00 00 00 00 84 54 D8 21 81 04 00 00 00 01 A2 2B 81 04 00 00 00 40 82 01 04 83 20 FD EA B9 AC F3 71 03 62 BD 26 58 CD C9 A2 9E 8F 9C 75 7F CF 98 11 60 3A 8C 44 7C D1 D9 15 11 08 A3 1D 81 0B 54 45 53 54 5F 31 4B 2E 62 69 6E 82 05 4C 61 62 65 6C 83 07 31 2E 30 2E 30 2E 31

Note: For additional support, please contact MagTek Support.

6.6.6 Command 0xD825 - Get File Info from Device

The host uses this command to request the file information of a file stored on the device. File types include standard files (images and certificates), MagTek custom files (configuration, firmware), and in some cases even large data blob output (such as signature capture data). In many cases, the file information retrieved by this command have been sent by a host previously using **Command 0xD811 - Start Send File to Device (Secured)** or **Command 0xD812 - Start Send File to Device (Unsecured)**. In other cases, such as retrieving file information of signature capture data, the data may originate with the device and the host uses this command to retrieve the information. Such information is not persistent, in the sense that the device does not retain it through power cycles.

The sequence of events is as follows:

1) The host composes a command request in the format below, and sends it to the device.
2) The device sends a response in the format below. The response contains the file information.
3) If the file can not be found, then a response of failure will be sent to the host.

Table 6.6-19 - Request Data for Command 0xD825 - Get File Info from Device

Tag	Len	Value / Description	Typ	Req	Default
Beginning of any wrappers, at minimum including **Request Message** found on page **32**					
D825 = **Command 0xD825 - Get File Info from Device**					
81	04	File ID from Table 6.6-1	B	R	
End of any wrappers, at minimum including **Request Message** found on page **32**					

Table 6.6-20 - Response Data for Command 0xD825 - Get File Info from Device

Tag	Len	Value / Description	Typ	Req	Default
Beginning of any wrappers, at minimum including **Response Message** found on page **33**					
D825 = **Command 0xD825 - Get File Info from Device**					
81	04	File ID from Table 6.6-1	B	R	
A2	var	File transfer properties	T	R	
/81	var	Length of File This is the length of the file.	B	R	

Tag	Len	Value / Description	Typ	Req	Default
/82	01	Hash Checksum Type • 0x04 = SHA-256	B	R	
/83	20	Hash Checksum Anticipated checksum calculated against the file, according to the standard specified in **Hash Checksum Type**.	B	R	
A3	var	File Description The values the host populated for convenience when it sent the file to help identify the file. Not all values are required.	T	R	
/81	var	File Name Maximum length 32 bytes Reserved for future use.	B	O	Null
/82	var	File Label Maximum length 16 bytes Reserved for future use.	B	O	Null
/83	var	File Version Maximum length 7 bytes Reserved for future use.	B	O	Null
/84	var	File Date Maximum length 20 bytes Reserved for future use.	B	O	Null
End of any wrappers, at minimum including **Response Message** found on page **33**					

If the request started successfully, the Request Status in the message wrapper is **OK, Started / Running, All good / requested operation was successful**.

Table 6.6-21 - Request Example

Example (Hex)
AA 00 81 04 01 08 D8 21 84 08 D8 25 81 04 00 00 00 01

Table 6.6-22 - Response Example

Example (Hex)
AA 00 81 04 82 08 D8 25 82 04 00 00 00 00 84 54 D8 25 81 04 00 00 00 01 A2 2B 81 04 00 00 00 40 82 01 04 83 20 FD EA B9 AC F3 71 03 62 BD 26 58 CD C9 A2 9E 8F 9C 75 7F CF 98 11 60 3A 8C 44 7C D1 D9 15 11 08 A3 1D 81 0B 54 45 53 54 5F 31 4B 2E 62 69 6E 82 05 4C 61 62 65 6C 83 07 31 2E 30 2E 30 2E 31

Note: For additional support, please contact MagTek Support.

6.6.7 Command 0xD831 – Delete File from Device

The host uses this command to request the deletion of a file stored on the device. File types include Custom Idle Page Image 1 to 4 in **Table 6.6-11**.

The sequence of events is as follows:

1) The host composes a command request in the format below and sends it to the device.

2) The device reads and erases the file and sends a response to the host in the format below.
3) If the file read or the file erase fails, a response of failure will be sent to the host.

Table 6.6-23 - Request Data for Command 0xD831 – Delete File from Device

Tag	Len	Value / Description	Typ	Req	Default
Beginning of any wrappers, at minimum including **Request Message** found on page **32**					
D831 = **Command 0xD831 – Delete File from Device**					
81	04	File ID from Table 6.6-1	B	R	
End of any wrappers, at minimum including **Request Message** found on page **32**					

Table 6.6-24 - Response Data for Command 0xD831 – Delete File from Device

Tag	Len	Value / Description	Typ	Req	Default
Beginning of any wrappers, at minimum including **Response Message** found on page **33**					
D831 = **Command 0xD831 – Delete File from Device**					
No parameters.					
End of any wrappers, at minimum including **Response Message** found on page **33**					

Table 6.6-25 - Request Example

Example (Hex)
AA 00 81 04 01 05 D8 31 84 08 D8 31 81 04 02 00 00 00

Table 6.6-26 - Response Example

Example (Hex)
AA 00 81 04 82 05 D8 31 82 04 00 00 00 00

6.7 Command Group 0xD9nn - Process Files

6.7.1 Command 0xD901 - Commit Firmware from File

The host uses this command to commit a file previously uploaded using **Command 0xD801 - Load Firmware File** into the device's permanent memory after the device has authenticated the file.

The sequence of events is as follows:

1) The host composes a command request in the format below, and sends it to the device.
2) The device sends a response in the format below.
3) The device writes the image file to permanent storage.
4) If the commit operation was successful, the device sends **Notification 0x0905 - Firmware Update Successful** to the host. If the commit operation was not successful, the device sends **Notification 0x0906 - Firmware Update Failed** to the host. In both cases, the device automatically resets.

Table 6.7-1 - Request Data for Command 0xD901 - Commit Firmware from File

Tag	Len	Value / Description	Typ	Req	Default
Beginning of any wrappers, at minimum including **Request Message** found on page **32**					
D901 = **Command 0xD901 - Commit Firmware from File**					
81	01	Progress Indicator Reserved for future use. Populate with 0x03.	B	R	
82	01	Operation Options Reserved for future use. Populate with 0x00.	B	R	
85	02	Image Type • 0x0000 = Boot Loader 1 image • 0x0001 = Main App image • 0x0002 = WiFi Module image • 0x0003 = BLE Module image	B	R	
86	20	Hash Checksum	B	R	
End of any wrappers, at minimum including **Request Message** found on page **32**					

Table 6.7-2 - Response Data for Command 0xD901 - Commit Firmware from File

Tag	Len	Value / Description	Typ	Req	Default
Beginning of any wrappers, at minimum including **Response Message** found on page **33**					
D901 = **Command 0xD901 - Commit Firmware from File**					
No parameters.					
End of any wrappers, at minimum including **Response Message** found on page **33**					

If the request started successfully, the Request Status in the message wrapper is **OK, Started / Running, All good / requested operation was successful**.

Table 6.7-3 - Request Example

Example (Hex)
AA 00 81 04 01 02 D9 01 84 2E D9 01 81 01 03 82 01 00 85 02 00 01 86 20 DF C7 1E 09 A3 CE 8E 86 B0 F5 B6 75 BE B7 7A 0E 82 33 BF F1 8A CD 8F 38 34 B0 DB 20 D9 40 4B 28

Table 6.7-4 - Response Example

Example (Hex)
AA 00 81 04 82 02 D9 01 82 04 00 00 00 00

Table 6.7-5 - Notification Example

Example (Hex)
AA 00 81 04 83 00 09 05 82 04 08 01 0A 03

Note: For additional support, please contact MagTek Support.

6.8 Command Group 0xDFnn - Diagnostics and Utilities

6.8.1 Command 0xDF01 - Echo

The host uses this command to prompt the device for a response that contains the same payload it sent.

The sequence of events is as follows:

1) The host constructs the command request for **Command 0xDF01 - Echo** in the format below, populating any of the available parameters with any data. The total length of data to be echoed across all parameters must not exceed 128 bytes.
2) The host sends the command request to the device.
3) The device sends a command response in the format below to the host, echoing back the exact parameters the host sent in the command request.

Table 6.8-1 - Request Data for Command 0xDF01 - Echo

Tag	Len	Value / Description	Typ	Req	Default
Beginning of any wrappers, at minimum including **Request Message** found on page **32**					
DF01 = **Command 0xDF01 - Echo**					
81	var	Data to be echoed	B	O	
82	var	Data to be echoed	B	O	
End of any wrappers, at minimum including **Request Message** found on page **32**					

Table 6.8-2 - Response Data for Command 0xDF01 - Echo

Tag	Len	Value / Description	Typ	Req	Default
Beginning of any wrappers, at minimum including **Response Message** found on page **33**					
DF01 = **Command 0xDF01 - Echo**					
81	var	Data being echoed	B	O	
82	var	Data being echoed	B	O	
End of any wrappers, at minimum including **Response Message** found on page **33**					

Table 6.8-3 - Request Example

Example (Hex)
AA 00 81 04 01 01 DF 01 84 07 DF 01 81 03 01 02 03

Table 6.8-4 - Response Example

Example (Hex)
AA 00 81 04 82 01 DF 01 82 04 00 00 00 00 84 07 DF 01 81 03 01 02 03

6.9 Command Group 0xEnnn - Security

6.9.1 Command 0xE001 - Get Challenge

The host uses this command to request challenge data from the device, which the host can then use to perform a specific sensitive operation / modify a specific type of device setting. Information about how

the host should pass the required challenge data to the device is included in the documentation for all commands that use this security mechanism.

The sequence of events is as follows:

1) The host already wants to perform a secured operation that requires a challenge [for example **Command 0xEEEE - Send Secured Command to Device**].
2) The host constructs the command request for **Command 0xE001 - Get Challenge** in the format below.
3) The host sends the command request to the device.
4) The device generates a random number for the challenge, stores it locally, and sends a response in the format below to the host.
5) The device starts a 5 minute countdown timer during which the challenge is valid. If the host takes no action within 5 minutes, the timer expires, the device erases the challenge data, and the device must retrieve a fresh challenge to perform the operation it wants to perform. This binding of the command to a specific time period allows the device to detect and reject commands that have been captured/intercepted at one point in time and replayed later.

Table 6.9-1 - Request Data for Command 0xE001 - Get Challenge

Tag	Len	Value / Description	Typ	Req	Default
Beginning of any wrappers, at minimum including **Request Message** found on page **32**					
E001 = **Command 0xE001 - Get Challenge**					
81	02	Request ID to be protected	B	R	
End of any wrappers, at minimum including **Request Message** found on page **32**					

Table 6.9-2 - Response Data for Command 0xE001 - Get ChallengeGet Challenge

Tag	Len	Value / Description	Typ	Req	Default
Beginning of any wrappers, at minimum including **Response Message** found on page **33**					
E001 = **Command 0xE001 - Get Challenge**					
81	02	Request ID to be protected	B	R	
82	04	Device Serial Number	B	R	
83	08	Challenge Token A challenge token includes 8 byte random numbers and must be used within 5 minutes of being issued. Only one token can be active at a time. Attempts to use a token for requests other than the one specified will cause the token to be revoked/erased	B	R	
End of any wrappers, at minimum including **Response Message** found on page **33**					

Table 6.9-3 - Request Example

Example (Hex)
AA 00 81 04 01 13 E0 01 84 06 E0 01 81 02 F0 12

Table 6.9-4 - Response Example

Example (Hex)
AA 00 81 04 82 13 E0 01 82 04 00 00 00 00 84 16 A2 14 81 02 E0 01 82 04 B5 03 3D A0 83 08 3B 4F A0 62 69 BB 73 38

6.9.2 Command 0xEEEE - Send Secured Command to Device

The host uses this command to transmit another command securely. This "secure wrapper" mechanism provides the device a means to ensure the wrapped command originated from an authentic, authorized host. In addition, its implementation includes an operation that starts a countdown timer, which ensures the command is current and is not an unauthorized replay of a previously intercepted / stored command. This command can use multiple authentication methods, including MAC or ECDSA Signature. The method and parameters to use are specific to the command being wrapped, and are specified in the documentation for that command.

The sequence of events is as follows:

1) The host determines what command it wants to call from section **6 Commands**, determines the command must be secured, and uses the Request Data table for that command to compose **Message Payload**.
2) The host uses **Command 0xE001 - Get Challenge** to retrieve a **Challenge Token** and unlock the device for receiving the desired command for a limited period of time. When the time expires, the device will no longer accept the Challenge Token and the host will have to retrieve another one.
3) The host creates an instance of **Command 0xEEEE - Send Secured Command to Device** in the format below, and includes the Message Payload and Challenge Token inside it. In the **Request Message**, it fills in **Command ID** as the command number of the wrapped Message Payload, instead of 0xEEEE. Some parameters are command-specific; see the documentation for the command that is being wrapped to determine what values to use.
4) The host sends the resulting composite command request to the device.
5) The device validates the serial number and challenge token, then examines the parameters to determine which authentication method is being used, and authenticates the command accordingly.
6) If the device determines the command request is authentic, it will start executing the secure command defined by the Message Payload.
7) The device sends a response to the host reporting success or failure. In both cases, the response uses the format that corresponds to the command invoked by the Message Payload. See the documentation for that command to determine the format of the response.

Table 6.9-5 - Request Data for Command 0xEEEE - Send Secured Command to Device

Tag	Len	Value / Description	Typ	Req	Default
Beginning of any wrappers, at minimum including **Request Message** found on page **32**					
EEEE = **Command 0xEEEE - Send Secured Command to Device**					
A1	var	Security Parameters This parameter describes how the Signature parameter in this data object is calculated, and is a **Security Parameters Type** TLV data object. To determine which values to use in that TLV data object, see the documentation for the command being wrapped.	T	R	

Tag	Len	Value / Description	Typ	Req	Default
82	04	Serial Number	B	R	
83	08	Challenge Token The token the device returned when the host called **Command 0xE001 - Get Challenge**.	B	R	
84	var	Message Payload	B	R	
9E	var	MAC or Signature	B	R	
End of any wrappers, at minimum including **Request Message** found on page **32**					

Table 6.9-6 - Request Example Using MAC

Example (Hex)
This example wraps **Command 0xD811 - Start Send File to Device (Secured)**. AA 00 81 04 01 04 D8 11 84 81 8F EE EE A1 19 81 05 03 03 06 02 08 84 00 85 00 A8 0A 81 02 11 02 82 00 86 00 88 00 A9 00 82 04 FF FF FF F0 83 08 C9 65 45 F2 97 69 85 B1 84 4E D8 11 81 04 00 00 03 00 A2 2B 81 04 00 00 02 99 82 01 04 83 20 87 A4 B3 54 61 C5 CB D3 1D DC BA 9D 65 25 5A D4 6A 22 FA 51 5E FD 65 87 AF AC A8 8C 4F AF 80 9B A3 14 38 31 30 38 33 30 33 30 33 30 33 30 33 30 33 33 33 30 33 30 87 01 01 9E 10 7D E4 27 C8 A0 70 72 08 19 0A 1E 0A 3F 48 BB F1

Note: For additional support, please contact MagTek Support.

6.9.3 Command 0xEF01 - Load Key Using TR-31

The host uses this command to load a key into one of several available slots in the device's secure memory.

Table 6.9-7 - Device Key ID / Slot

ID	Label	Description	Load TK
1000	TMPTK	Temporary KBPK	agree
1001	MTK	Master Transport	TMPTK
1002	DEVTK	Device Master	MTK
1003	FINTK	Financial Master	MTK
1021	PRODTK	Production - MagTek Internal Only	DEVTK
1022	MFGTK	MagTek Only Internal/External	DEVTK
1081	MKIFTK	MagTek KIF Financial Keys	FINTK
1101	FREQMK	Factory Request MAC	PRODTK
1102	MREQMK	Mfg Device Request MAC	MFGTK
1111	MFRQMK	Mfg Financial Request MAC	MKIFTK
20xx	DKPTM0-1F	MagTek DUKPT Initial Key	MKIFTK

To inject a specific key in the above table, the corresponding Load TK shall be injected previously

As shown in the table, MTK injection requires that a TMPTK has been created. See **Command 0xF017 - Establish Ephemeral KBPK**.

After MTK has been injected successfully, the sequence of injecting other keys is as follows:

1) The host uses **Command 0xE001 - Get Challenge** to establish a secure session with the device.
2) The host constructs a TR-31 (X9.143) key block for the key it is going to load. (Note that the Load Key must be injected previously.)
3) The host constructs the command request for **Command 0xEF01 - Load Key Using TR-31** in the format below.
4) The host sends the command request to the device.
5) The device sends a response in the format below to the host.

Table 6.9-8 - Request Data for Command 0xEF01 - Load Key Using TR-31

Tag	Len	Value / Description	Typ	Req	Default
Beginning of any wrappers, at minimum including **Request Message** found on page **32**					
EF01 = **Command 0xEF01 - Load Key Using TR-31**					
84	var	Key Block This is a populated, secured **TR-31 Key Block Type**.	B	R	
End of any wrappers, at minimum including **Request Message** found on page **32**					

Table 6.9-9 - Response Data for Command 0xEF01 - Load Key Using TR-31

Tag	Len	Value / Description	Typ	Req	Default
Beginning of any wrappers, at minimum including **Response Message** found on page **33**					
EF01 = **Command 0xEF01 - Load Key Using TR-31**					
No parameters.					
End of any wrappers, at minimum including **Response Message** found on page **33**					

Table 6.9-10 - Request Example

Example (Hex)
AA 00 81 04 01 1A EF 01 84 82 01 36 EF 01 84 82 01 30 44 30 33 30 34 42 31 54 58 30 30 4E 30 36 30 30 49 4B 31 38 46 46 46 46 39 38 37 36 35 34 33 32 31 30 33 30 30 30 30 30 32 31 35 38 4D 47 54 4B 31 30 30 31 54 31 31 30 34 32 30 30 37 31 32 30 34 31 30 38 31 32 31 30 34 30 30 33 46 33 31 30 37 42 35 30 41 46 44 32 33 32 31 30 43 39 33 36 31 39 44 44 41 41 32 31 33 36 43 37 33 33 31 30 32 30 32 30 31 32 30 34 54 31 37 31 36 33 30 5A 4B 50 30 45 30 31 39 33 36 46 41 33 32 45 4B 43 30 41 30 30 34 35 30 30 54 53 31 34 32 30 32 30 30 39 30 32 54 31 35 35 38 30 32 5A 50 42 30 34 35 37 32 31 37 46 33 34 37 31 34 43 32 42 38 38 46 33 39 35 35 32 32 32 46 46 35 39 41 41 30 35 37 44 39 39 41 46 38 32 41 37 35 37 32 46 39 33 38 46 38 33 38 42 43 36 35 45 45 35 34 46 39 34 37 46 35 39 41 30 36 43 44 34 35 35 31 39 32 32 37 41 32 35 35 43 37 44 35 44 37 43 38 36 37 34 35 30 33 46 41 43 36 46 41 37 31 33 32 43 38 46 41 39 39 36 42 34 45 42 36 41 41 31 31 34 46 45

Table 6.9-11 - Response Example

Example (Hex)
AA 00 81 04 82 1A EF 01 82 04 00 00 00 00

Note: For additional support, please contact MagTek Support.

6.9.4 Command 0xEF06 – Change Device Lock State

The host can use this command to change the device's lock state. To get the device's lock state or to set it using MagTek security see **Property 1.2.5.2.1.1 Device Lock State**. The value of the device lock state will revert to the value of **Property 1.2.5.2.1.2 Device Lock State After Reset** after a reset or a power cycle. See **Device Lock Feature** for more information.

Table 6.9-12 - Request Data for Command 0xEF06 – Change Device Lock State

Tag	Len	Value / Description	Typ	Req	Default
Beginning of any wrappers, at minimum including **Request Message** found on page **32**					
EF06 = **Command 0xEF06 – Change Device Lock State**					
81	01	Device Lock State • 0x00 = Unlocked • 0x01 = Locked	B	M	
82	01	Passcode Format • 0x00 = Clear • 0x01 = Fixed SHA-256 • 0x02 = Variable SHA-256	B	M	
83	04-63	Passcode The value of the passcode depends on the value of the passcode format parameter. If the passcode format is set to clear, then the value of the passcode is the passcode in the clear and can have a length of 4-63 bytes. If the passcode format is set to fixed SHA-256, then the value of the passcode is the 32 byte SHA-256 hash value of the passcode. If the passcode format is set to variable SHA-256, then the value of the passcode is the 32 byte SHA-256 hash value of a 8 byte random challenge token followed by the 4-63 byte passcode. The challenge token must have been retrieved from the device within the last 5 minutes using Command 0xE001 - Get Challenge.	B	M	
End of any wrappers, at minimum including **Request Message** found on page **32**					

Table 6.9-13 - Response Data for Command 0xEF06 – Change Device Lock State

Tag	Len	Value / Description	Typ	Req	Default
Beginning of any wrappers, at minimum including **Response Message** found on page **33**					

Tag	Len	Value / Description	Typ	Req	Default
EF06 = **Command 0xEF06 – Change Device Lock State**					
No parameters.					
End of any wrappers, at minimum including **Response Message** found on page **33**					

Table 6.9-14 - Request Example

Example (Hex)
AA00 81 04 0155EF06 84 0E EF06 810100 820100 8304 34333231

Table 6.9-15 - Response Example

Example (Hex)
AA00 81048255EF06 820400000000 8402EF06

6.9.5 Command 0xEF07 – Change Device Lock Passcode

The host can use this command to change the device's lock passcode. The value of the device lock passcode is stored in non-volatile memory so changes made to it will persist after the device is reset or power cycled. To change the device lock passcode using MagTek security or to see its default value see **Property 1.2.5.2.1.3 Device Lock Passcode**. See **Device Lock Feature** for more information.

Table 6.9-16 - Request Data for Command 0xEF07 – Change Device Lock Passcode

Tag	Len	Value / Description	Typ	Req	Default
Beginning of any wrappers, at minimum including **Request Message** found on page **32**					
EF07 = Command **0xEF07 – Change Device Lock Passcode**					
81	4-63	Current Passcode The current passcode in the clear. This must match the value of the current passcode or the command will fail.	B	M	
82	04-63	New Passcode The new passcode in the clear. It can only contain any printable ASCII character or the command will fail.	B	M	
End of any wrappers, at minimum including **Request Message** found on page **32**					

Table 6.9-17 - Response Data for Command 0xEF07 – Change Device Lock Passcode

Tag	Len	Value / Description	Typ	Req	Default
Beginning of any wrappers, at minimum including **Response Message** found on page **33**					
EF07 = Command **0xEF07 – Change Device Lock Passcode**					
No parameters.					
End of any wrappers, at minimum including **Response Message** found on page **33**					

Table 6.9-18 - Request Example

Example (Hex)
AA00 81 04 0155EF07 84 0E EF07 810434333231 820434333231

Table 6.9-19 - Response Example

Example (Hex)
AA00 81048255EF07 820400000000 8402EF07

6.9.6 Command 0xEF11 - Get Key Info

The host uses this command to retrieve information about a key slot, including details about the key stored in that slot. It can be used for several purposes, including:

- Determine if a key exists / has been loaded
- Get key derivation data to derive a DUKPT key
- Get transport key information to retrieve the appropriate transport key

The sequence of events is as follows:

1) The host constructs the command request in the format below.
2) The host sends the command request to the device.
3) The device sends a response in the format below to the host.

Table 6.9-20 - Request Data for Command 0xEF11 - Get Key Info

Tag	Len	Value / Description	Typ	Req	Default
Beginning of any wrappers, at minimum including **Request Message** found on page **32**					
EF11 = **Command 0xEF11 - Get Key Info**					
81	02	Key Slot ID See **Table 4.18-6 - Key Slot ID** on page **87**.	B	R	
End of any wrappers, at minimum including **Request Message** found on page **32**					

Table 6.9-21 - Response Data for Command 0xEF11 - Get Key Info

Tag	Len	Value / Description	Typ	Req	Default
EF11 = **Command 0xEF11 - Get Key Info**					
81	04	Key Slot Information	B	R	
/null	(1)	Key Slot Status • 0x00 = Empty • 0x01 = Loaded (Key not assigned purpose) • 0x02 = Loaded & Active • 0x03 = Exhausted (End of DUKPT key sequence • 0x04 = Expired (Reserved, certificate status) • 0xFF = Not supported in this device	B	R	
/null	(1)	Key Slot Type First byte of the Key Slot ID in the host's request message.	B	R	

Tag	Len	Value / Description	Typ	Req	Default
/null	(2)	Transport Key Slot ID This specifies the key used to secure and load the key that the host is retrieving information about. See **Table 4.18-6 - Key Slot ID** on page **87**.	B	R	
82	06	Loaded Key Information	B	O	
/null	(1)	Key Environment • 'T' = Test • 'P' = Production	A	R	
/null	(4)	TR-31 Attributes See **Table 4.18-2 - TR-31 Key Type Table - Usage/Algorithm/Mode.**	B	R	
/null	(1)	Encoding of Algorithm & Length • 0x01 = DEA • 0x02 = 2TDEA • 0x03 = 3TDEA • 0x04 = AES128 • 0x05 = AES192 • 0x06 = AES256	B	R	
83	var	Key Check Value • For AES-CMAC, 5 bytes. • For TDES-CMAC or TDES-CBCMAC, 3 bytes	B	O	
84	var	Key Derivation Information This contains the derivation block, key serial number (KSN), or key label, as appropriate for the key type.	B	O	
A6	var	Restrictions Reserved. Do not include.	B	O	
81	02	DUKPT Restrictions These restrictions come from the TR-31 block.	B	O	
89	var	Timestamp This comes from the TR-31 block or from device's real-time clock.	B	O	

Table 6.9-22 - Request Example

Example (Hex)
AA 00 81 04 01 21 EF 11 84 06 EF 11 81 02 20 07

Table 6.9-23 - Response Example

Example (Hex)
AA 00 81 04 82 21 EF 11 82 04 00 00 00 00 84 34 A2 32 81 04 02 20 10 81 82 06 54 42 31 54 58 03 84 0A FF FF 98 76 54 32 10 30 00 00 A6 04 81 02 00 3F 89 10 32 30 32 30 30 39 30 32 54 31 35 35 38 30 32 5A

Note: For additional support, please contact MagTek Support.

6.10 Command Group 0xFnnn - Manufacturing

6.10.1 Command 0xF017 - Establish Ephemeral KBPK

The host uses this command to complete the ***ECDHE-ECDSA Key Exchange*** protocol, which enables the host and the device to generate the same TEMP KBPK key to use with **Command 0xEF01 - Load Key Using TR-31** to load the Master Transport Key (MTK).

The sequence of events is as follows:

1) The host uses **Command 0xE001 - Get Challenge** to establish a secure session with the device.
2) The host constructs **Command 0xF017 - Establish Ephemeral KBPK** per the request table below.
3) The host constructs **Command 0xEEEE - Send Secured Command to Device** using the previously constructed command request as the payload, and sends that command request to the device as a **Request Message**.
   a) Because this command is secured using a signature, read **Property 2.1.2.2.2.6 Key Type** to determine which fixed key to use to generate the signature.
   b) Build the **Security Parameters Type** portion of the wrapper with:
      i) **Security Operation Type** populated with the following values:
         (1) Operation Type = **Command Authorization Using Signature**
         (2) Operation Algorithm = **ECDSA (indeterministic)**
         (3) Operation Hash = **SHA-256**
         (4) Operation Curve = **P521**.
4) The device does the following:
   a) Validates the secure wrapper around the command, and terminates if the signature is invalid.
   b) Determine if the Master Transport Key (MTK) has already been loaded. If it has, the device rejects the command request.
   c) Generates a pair of keys, saves the Device Private Key for calculation.
   d) Generates and saves 8 bytes of Device Random Token for calculation.
   e) Calculates the TEMP KBPK based on Host Public Key and Host Random Token passed in with the command request, and the Device Private Key and Device Random Token the device generated.
5) The device sends a **Response Message** to the host to indicate the result. The response message includes Device Random Token and Device Public Key.
6) The host calculates a matching TEMP KBPK as defined in as defined in ***NIST SP800-56A***, using the Host Private Key, Host Random Token, Device Public Key, and Device Random Token. It can then use this key to perform encryption operations on secret data in the Master Transport Key (MTK).
7) The device uses its copy of the matching TEMP KBPK to decrypt the secret information encrypted by the host using the same key. On successful MTK load, the device erases the TEMP KBPK. It also erases the TEMP KBPK if the device is power cycled or reset, and the host would need to restart the process with a new TEMP KBPK.

Table 6.10-1 - Request Data for Command 0xF017 - Establish Ephemeral KBPK

Tag	Len	Value / Description	Typ	Req	Default
Beginning of any wrappers, at minimum including **Request Message** found on page **32**					

Tag	Len	Value / Description	Typ	Req	Default
F017 = **Command 0xF017 - Establish Ephemeral KBPK**					
A1	var	Security Parameters This contains a **Security Parameters Type** TLV data object with only the first parameter populated with: 01 = Key Agreement, 01 = ECDHE, 05 = Curve P521, 01 = SP800-56A	B	R	
83	var	Host Ephemeral Public Key This parameter is in ASN.1 format. The information of the cipher and key size are included in the ASN.1 Public Key file (PKCS#8).	B	R	
84	08	Host Random Token This contains an 8 byte random number generated by the host.	B	R	

Table 6.10-2 - Response Data for Command 0xF017 - Establish Ephemeral KBPK

Tag	Len	Value / Description	Typ	Req	Default
Beginning of any wrappers, at minimum including **Response Message** found on page **33**					
F017 = **Command 0xF017 - Establish Ephemeral KBPK**					
A1	var	Security Parameters This contains a **Security Parameters Type** TLV data object populated entirely with 0x00 padding to indicate that all values are the same as the corresponding values in the Request.	B	R	
83	var	Device Ephemeral Public Key This parameter is in ASN.1 format. The information of the cipher and key size are included in the ASN.1 key file.	B	R	
84	08	Device Random Token This contains an 8 byte random number generated by the device.	B	R	
End of any wrappers, at minimum including **Response Message** found on page **33**					

Table 6.10-3 - Request Example

Example (Hex)
AA 00 81 04 01 10 F0 17 84 82 01 8C EE EE // Secure Wrapper A1 24 81 04 02 01 04 05 84 00 85 00 A8 16 81 02 00 00 82 07 45 43 43 53 49 47 4E

Example (Hex)

```
86 05 45 43 44 53 41
88 00 A9 00
82 04 B5 03 3D A0
83 08 ED B0 79 E6 E3 F1 83 AE
84 81 C3                        // payload is 0xF017 command body
F0 17
A1 14                       // Security parameters
81 04 01 01 05 01
84 00
A8 0A
81 02 00 00
82 00
86 00
88 00
83 81 9E                        // TL of PKCS8 public key
30 81 9B                        // V of PKCS8 public key
30 10
06 07 2A 86 48 CE 3D 02 01
06 05 2B 81 04 00 23
03 81 86
00
04
01 64 1C DA 45 C5 56 B3 8B 31 29 8C 94 A1 E7 95
C9 D3 85 C0 4D F3 15 13 D9 91 43 84 58 15 CD 45
6B 67 F6 AC 7C 56 DF F8 0C 65 A7 CF 81 F1 13 2F
AA E5 22 10 78 23 C9 4F 1D CD 24 42 EC 1A 3F A4
75 58
00 97 59 96 9E 01 D0 62 47 B7 EF 5F 0B D0 8B E6
CA 12 F0 3C 13 43 AF 15 21 92 3D 6B FE 47 74 68
38 3F DD 1E 90 2B FD 0F D6 DA 7A A1 E9 A1 98 85
3A DA 93 6D EE 05 61 87 8B 81 BF 6A 78 2F 40 A5
E8 66
84 08 BD E3 77 88 83 0C F6 37       // TLV of 8-byte random # for TEMP-KBPK
                                // end of 0xF017 command body
9E 81 8B
30 81 88
02 42
00 C4 13 1D C2 13 7A F6 FD F0 F1 BB BD 14 C2 4A
FE D7 6F BC 80 91 84 26 43 85 40 B6 5D BE 1D 9C
74 90 77 B6 41 62 69 52 04 72 93 C0 9C 59 2A DB
03 31 0F 8A 28 C0 DB 1A B7 1B 51 B3 E6 BD FF 50
77 CA
02 42
01 EE D8 2D 9F A3 D1 98 4E 74 C8 85 11 52 93 15
FF 9D 7D 5A 03 FD 84 B8 B9 09 20 8B 15 98 7A 5E
56 A5 61 71 9A 0A B9 D1 DA 1C 96 1D 0C EF F0 D2
E3 A4 22 84 60 E2 AA 8C AA 2B 8B AE 02 50 D8 B3
CF 84
```

Table 6.10-4 - Response Example

Example (Hex)
AA 00 81 04 82 10 F0 17 82 04 00 00 00 00 84 81 B5 // Response Payload F0 17 A1 06 81 01 00 82 01 00 83 81 9E // TL of PKCS8 public key 30 81 9B // V of PKCS8 public key 30 10 06 07 2A 86 48 CE 3D 02 01 06 05 2B 81 04 00 23 03 81 86 00 04 01 77 CD 91 56 96 34 2B C6 5A 6C EC 5D 74 96 41 B3 F9 2B 12 85 19 90 F8 73 BF FF 3C 10 44 E3 CB 21 4E CA F6 CE FC F8 C8 80 52 44 13 FA B1 97 A1 8C 44 FE 95 A2 0A F3 3D A4 3A 8F 2E 39 41 23 22 B1 AB 01 29 26 4F CC 0E 86 11 16 92 FF BC E1 BF DA FC 21 BA B1 5A C4 DE 7B C1 6F A9 17 F8 4B 1E B2 1F 5F 21 7D 54 00 15 41 C3 21 75 0D 21 DC 95 13 A7 2C 8C 11 77 96 38 87 51 08 7A 1F 63 EC A8 8F C4 AB B3 84 08 4C 4A EC 0B 47 E4 53 EB // TLV of 8-byte random # for TEMP-KBPK

Note: For additional support, please contact MagTek Support.

7 Notifications

This section provides definitions and practical information about **Notification Message**s, which follow the format described in section **3.2.2.3**. To find documentation about a specific notification:

1) Use the **Notification Source** byte in the message to find the Notification Source subsection below. For example, Notification Source **0x01 Transaction** leads to subsection **7.1 Notification Source 0x01nn - Notifications from Transactions**.
2) Within that subsection, use the **Notification Source** byte plus the **Notification Type** byte in the message to find the Notification subsection. For example, Notification Source **0x01 Transaction** plus Notification Type **0x01 Information Update** leads to **Notification 0x0101 - Transaction Information Update**.
3) Within that subsection, use the four bytes of **Notification Detail Code** in the message to look up the notification in the **Notification Detail Code** table. For example, **0x07080302** leads to a row in **Table 7.1-1** that provides practical information about what event led to the notification, and how to interpret the Notification Payload.

7.1 Notification Source 0x01nn - Notifications from Transactions

7.1.1 Notification 0x0101 - Transaction Information Update

This notification reports information about progress and state changes that occur during a transaction.

For this notification, **Notification Detail** described in section **3.2.2.3 Notification Message** contains one byte each, shown in **Table 7.1-1**, to indicate:

- The **Payment Technology** (PT) involved
- The **Reason** (Rsn) for the notification
- **Detail** (Det) about the notification that has different meanings depending on the Reason, and
- An **Extra** field (Ext) that has different meanings depending on the Reason.

Table 7.1-1 - Notification Detail Codes

PT	Rsn	Det	Ext	Meaning
Payment Technology **0x10 EMV Contact** contains transaction notification detail codes involving contact chip cards (EMV Contact Only). • Reason 0x01 = Card Event • Reason 0x02 = Online PIN Event • Reason 0x08 = Data Update • Reason 0x10 = State Update				
10	01	02	00	EMV Contact, Card Event, Inserted, Reserved (EMV Contact Only)
10	01	03	00	EMV Contact, Card Event, Removed, Reserved (EMV Contact Only)
10	01	04	00	EMV Contact, Card Event, Detected, Reserved (EMV Contact Only)
10	02	06	01	EMV Contact, Online PIN Event, PIN Entry, PIN Pad Error (EMV Contact Only)
10	02	06	02	EMV Contact, Online PIN Event, PIN Entry, PIN Block Encryption Error (EMV Contact Only)

PT	Rsn	Det	Ext	Meaning
10	08	02	02	EMV Contact, Data Update, ARQC Update (Quick Chip), Data Attached In this case, the device includes additional data, defined in **Table 7.1-2**, in the **Notification Payload** portion of the **Notification Message** (EMV Contact Only).
10	08	03	02	EMV Contact, Data Update, Batch Data, Data Attached In this case, the device includes additional data, defined in **Table 7.1-2**, in the **Notification Payload** portion of the **Notification Message** (EMV Contact Only).
Payment Technology **0x20 EMV Contactless** contains transaction notification detail codes involving contactless cards and contactless payment devices. • Reason 0x01 = Card Event • Reason 0x02 = Online PIN Event • Reason 0x08 = Data Update • Reason 0x10 = State Update				
20	01	03	00	EMV Contactless, Card Event, Removed, Reserved (EMV Contactless Only)
20	01	04	00	EMV Contactless, Card Event, Detected, EMV (EMV Contactless Only)
20	01	04	01	EMV Contactless, Card Event, Detected, NFC/MIFARE Ultralight (EMV Contactless Only)
20	01	04	02	EMV Contactless, Card Event, Detected, MIFARE Classic 1K (EMV Contactless Only)
20	01	04	03	EMV Contactless, Card Event, Detected, MIFARE Classic 4K (EMV Contactless Only)
20	01	04	04	EMV Contactless, Card Event, Detected, MIFARE DESFire Light (EMV Contactless Only)
20	01	04	05	EMV Contactless, Card Event, Detected, MIFARE MINI® (EMV Contactless Only)
20	01	04	06	EMV Contactless, Card Event, Detected, MIFARE Plus EV1 (EMV Contactless Only)
20	01	04	07	EMV Contactless, Card Event, Detected, MIFARE Plus EV2 (EMV Contactless Only)
20	01	04	08	EMV Contactless, Card Event, Detected, MIFARE Plus SE (EMV Contactless Only)
20	01	04	09	EMV Contactless, Card Event, Detected, MIFARE Plus X (EMV Contactless Only)
20	01	04	0A	EMV Contactless, Card Event, Detected, MIFARE DESFire EV1 (EMV Contactless Only)
20	01	04	0B	EMV Contactless, Card Event, Detected, MIFARE DESFire EV2 (EMV Contactless Only)
20	01	04	0C	EMV Contactless, Card Event, Detected, MIFARE DESFire EV3 (EMV Contactless Only)
20	01	05	00	EMV Contactless, Card Event, Collision, Reserved (EMV Contactless Only)

PT	Rsn	Det	Ext	Meaning
20	01	10	02	EMV Contactless, Card Event, Apple VAS Error, Data Attached In this case, the device includes additional data, defined in **Table 7.1-5** – **VAS Error Report**, in the **Notification Payload** portion of the **Notification Message** (Apple VAS Only).
20	01	20	02	EMV Contactless, Card Event, Google Smart Tap Error, Data Attached In this case, the device includes additional data, defined in **Table 7.1-5** – **VAS Error Report**, in the **Notification Payload** portion of the **Notification Message** (Google Smart Tap Only).
20	02	06	01	EMV Contactless, Online PIN Event, PIN Entry, PIN Pad Error (EMV Contactless Only)
20	02	06	02	EMV Contactless, Online PIN Event, PIN Block Encryption Error (EMV Contactless Only)
20	08	02	02	EMV Contactless, Data Update, ARQC Update (Quick Chip), Data Attached In this case, the device includes additional data, defined in **Table 7.1-2**, in the **Notification Payload** portion of the **Notification Message** (EMV Contactless Only).
20	08	03	02	EMV Contactless, Data Update, Batch data, Data attached In this case, the device includes additional data, defined in **Table 7.1-2**, in the **Notification Payload** portion of the **Notification Message** (EMV Contactless Only)
20	08	04	02	EMV Contactless, Data Update, NFC UID data, Data attached In this case, the device includes additional data, defined in **Table 7.1-2**, in the **Notification Payload** portion of the **Notification Message** (EMV Contactless Only)
20	08	07	02	EMV Contactless, Data Update, GPO Response, Data Attached In this case, the device includes additional data, defined in **Table 7.1-2**, in the **Notification Payload** portion of the **Notification Message** (EMV Contactless Only)
Payment Technology **0x30 Barcode Reader** contains transaction notification detail codes involving barcodes (BCR Only). • Reason 0x01 = Barcode Event				
30	01	01	00	(BCR Only) Barcode Reader, Barcode Event, Read, Reserved

PT	Rsn	Det	Ext	Meaning
30	08	00	02	Barcode Reader, Barcode Update, Type, Data Attached • Type 0x00 = Unknown • Type 0x01 = MagTek • Type 0x02 = EMV The barcode reader has successfully read a barcode. In this case, the device includes barcode data in the **Notification Payload** portion of the **Notification Message**. If the data is encrypted, the data is in the format described in **Table 7.4-5**. Data that is not encrypted is in the format described in **Table 7.4-6** (BCR Only) Barcode Reader, Barcode Event, MagTek Blob Type, Data Attached In this case, the device includes additional data, defined in **Table 7.1-2**, in the **Notification Payload** portion of the **Notification Message**.

Table 7.1-2 - Notification Payload for Data Update, ARQC Update (Quick Chip), Data Attached

Tag	Len	Value / Description	Typ	Req	Default
Beginning of **Notification Message** found on page **38**					
0101 = DynaPro Format Transaction Data					
84	var	Transaction Data	B	R	
/null	(v)	**EMV ARQC Type** data blob, if the notification is an ARQC Update (Quick Chip) notification. **EMV Batch Data Type** data blob, if the notification is a Batch Data notification. Decoded raw barcode data, if the notification is a Barcode event notification. If the barcode data is Base64 encoded, the device sends the decoded version in binary format. **NFC UID Type** reports the unique ID from the NFC tag (EMV Contactless Only) **GPO Response Type reports the card's GPO Response (EMV Contactless Only)**	B	O	
End of **Notification Message** found on page **38**					

Table 7.1-3 - Notification Payload for Barcode Reader, Barcode Event, Type (Encrypted Data Attached)

Tag	Len	Value / Description	Typ	Req	Default
Beginning of **Notification Message** found on page **38**					
1805 = User Interface Operation Complete					
84	var	Notification Payload	B	R	
/DFDF59	var	Encrypted Data Primitive	B	R	

Tag	Len	Value / Description	Typ	Req	Default
		Decrypt the value of this TLV data object using the algorithm and variant specified in the **Encrypted Data KSN** parameter and the **Encrypted Data Encryption Type** parameter to read its contents. The format of the decrypted data is shown in **Table 7.1-4**.			
/DFDF50	var	Encrypted Data KSN	B	R	
/DFDF51	01	Encrypted Data Encryption Type See section **4.4 Encryption Type** for a list of valid values.	B	R	
End of **Notification Message** found on page **38**					

Table 7.1-4 - Notification Payload for Barcode Reader, Barcode Event, Type, Data Attached (Unencrypted Data)

Tag	Len	Value / Description	Typ	Req	Default
FC	var	Barcode Data Container	T	R	
/DF74	var	Barcode Data	B	O	

Table 7.1-5 – VAS Error Report

Tag	Len	Value / Description	Typ	Req	Default
FC	var	VAS Error Container	T	R	
/DFDF25	var	Device Serial Number (IFD Serial Number)	B	R	
/DF75	var	VAS Error Report Byte 1 = SW1 returned by the Apple or Android device Byte 2 = SW2 returned by the Apple or Android device Byte 3 = Slot Number Byte 4 = Where in the VAS command sequence the error occurred. • 0x00 = Select OSE Error • 0x01 = Get VAS Data Error (Apple only) • 0x02 = Select Smart Tap Error (Google only) • 0x03 = Negotiate Secure Session Error (Google only) • 0x04 = Get Select Smart Tap Error (Google only)	B	R	

Table 7.1-6 - Notification Example

Example (Hex)
AA 00 81 04 83 00 01 01 82 04 10 08 02 02 84 82 01 F8 01 01 84 82 01 F2 01 E9 F9 82 01 E5 DF DF 54 0A 00 00 00 00 00 00 00 00 00 00 DF DF 55 01 82 DF DF 25 0F 42 34 44 42 38 31 37 31 31 31 39 31 39 41 41 FA 82 01 BB 70 82 01 B7 DF DF 53 01 00 5F 20 12 47 55 53 54 49 4E 2F 53 54 45 50 48 41 4E 49 45 20 4D 5F 30 02 02 01 DF DF 4D 25 3B 35 33 32 32 30 30 30 30 36 30 30 30 31 36 37 34 3D 31 38 30 39 32 30 31 30 30 30 30 30 30 30 30 30 30 30 3F DF DF 52 01 05 F8 82 01 66 DF DF 59 82 01 48 36 69 98 57 4C 81 08 66 28 B6 1F DD 69 B1 C3 43 F7 BA 98 B2 5A 92 53 5F DA 63 6D DA 44 95 F1 15 2D 01 07 9A 4C EB 28 0B 30 5C 21 B2 39 ED E7 EE B6 1A 79 43 56 2E 26 1E C9 87 86 19 68 FA EB 4A 2B BD F5 5D 54 6F C4 67 97 FF 42 2C D6 CF 36 03 48 58 A4 23 9B 51 03 5C 32 0B DC 5E 4E E5 95 1B B5 C5 18 E7 33 0B D2 FE 8D E8 5C 47 4D 3C 16 79 42 48 1D CD 83 D5 58 64 48 23 17 F3 29 A7 F1 F1 75 F4 B9 C6 45 F3 02 28 1C 90 C7 83 B9 49 AF 56 BD 76 73 E7 45 7D 25 C5 77 3F C7 9A 1B ED 52 0A 05 54 15 B7 9A 2A 59 C1 67 6D E2 8C 02 8B 97 64 96 5D 4C F8 31 A2 20 75 12 8D 99 C1 A2 DF AB 55 0A 62 24 79 CF A8 51 3D AE 84 91 A5 80 19 9F BC 75 B0 F9 56 5E BA 57 A3 B1 61 AA 84 43 F0 D8 E6 44 C1 FA 51 0E 0A B0 F7 F2 61 57 5B 86 7E AB DC 49 00 87 A0 3B 69 5F C1 45 C4 10 9A A8 5F B6 30 59 2C 25 FA 15 A5 44 83 24 96 3D 5A 03 50 36 02 EC 6B 15 7C 8D CC 66 BF B7 F4 CF 4C 6D 67 75 87 B9 4E D4 08 76 25 F8 B8 EF BF A8 A0 72 F9 81 AB FF 49 84 E7 BC 8F C5 DD A0 86 B2 74 DD 59 8A B2 83 5D DD CC 0C 30 01 96 DF DF 56 0A 90 10 01 0B 4D B8 17 00 00 03 DF DF 57 01 80 DF DF 58 01 05 00 00 00 00 00 00 00

7.1.2 Notification 0x0105 - Transaction Operation Complete

This notification reports the final result of a transaction the host initiated using **Command 0x1001 - Start Transaction**.

For this notification, **Notification Detail** described in section **3.2.2.3 Notification Message** contains one byte each, shown in **Table 7.1-7**, to indicate:

- The **Payment Technology** (PT) involved
- The **Reason** (Rsn) for the notification
- **Detail** (Det) about the notification that has different meanings depending on the Reason, and
- An **Extra** field (Ext) that has different meanings depending on the Reason.

Table 7.1-7 - Notification Detail Codes

PT	Res	Det	Ext	Meaning
Payment Technology **0x00 None** contains transaction notification detail codes that are not specific to a particular payment technology. • 0x00 = Reserved • 0x01 = Timeout/No Cards Detected • 0x02 = Kernel Outcome • 0x03 = Transaction Terminated • 0x04 = Transaction Completed				
00	01	00	00	None, Timeout/No Cards Detected, Transaction Timed Out, Reserved
00	03	01	00	None, Transaction Terminated, Transaction Canceled by Host, Reserved

PT	Res	Det	Ext	Meaning
Payment Technology **0x10 EMV Contact** ("ICC") contains transaction notification detail codes involving contact chip cards (EMV Contact Only). • 0x00 = Reserved • 0x01 = Timeout, no cards detected • 0x02 = Kernel Outcome • 0x03 = Transaction Terminated • 0x04 = Transaction Completed				
10	02	08	03	ICC, Kernel Outcome, Failed, MSR Fallback (EMV Contact Only)
10	02	09	03	ICC, Kernel Outcome, Empty Candidate List, MSR Fallback (EMV Contact Only)
10	02	00	00	ICC, Kernel Outcome, None, Reserved (EMV Contact Only)
10	02	00	02	ICC, Kernel Outcome, None, Technical Fallback (EMV Contact Only)
10	02	0F	02	ICC, Kernel Outcome, End Application, Technical Fallback (EMV Contact Only)
10	02	01	00	ICC, Kernel Outcome, Try Another Interface, Reserved (EMV Contact Only)
10	02	02	00	ICC, Kernel Outcome, Offline Approved, Reserved (EMV Contact Only)
10	02	02	01	ICC, Kernel Outcome, Offline Approved, Signature Capture Requested (EMV Contact Only)
10	02	02	04	ICC, Kernel Outcome, Offline Approved, Signature Capture Available (EMV Contact Only)
10	02	02	05	ICC, Kernel Outcome, Offline Approved, Signature Capture Success (EMV Contact Only) The device includes signature data inside **EMV Batch Data Type** tag DFDF3E.
10	02	02	06	ICC, Kernel Outcome, Offline Approved, Signature Capture Fail (EMV Contact Only)
10	02	03	00	ICC, Kernel Outcome, Offline Declined, Reserved (EMV Contact Only)
10	02	04	00	ICC, Kernel Outcome, Offline Failed, Reserved (EMV Contact Only)
10	02	05	00	ICC, Kernel Outcome, Offline Not Accepted, Reserved (EMV Contact Only)
10	02	06	00	ICC, Kernel Outcome, Approved, Reserved (EMV Contact Only)
10	02	06	01	ICC, Kernel Outcome, Approved, Signature Capture Requested (EMV Contact Only)
10	02	06	05	ICC, Kernel Outcome, Approved, Signature Capture Success (EMV Contact Only) The device includes signature data inside **EMV Batch Data Type** tag DFDF3E.
10	02	06	06	ICC, Kernel Outcome, Approved, Signature Capture Fail (EMV Contact Only)
10	02	07	00	ICC, Kernel Outcome, Quick Chip Deferred, Reserved (EMV Contact Only)
10	02	07	01	ICC, Kernel Outcome, Quick Chip Deferred, Signature Capture Requested (EMV Contact Only)

PT	Res	Det	Ext	Meaning
10	02	07	05	ICC, Kernel Outcome, Quick Chip Deferred, Signature Capture Success (EMV Contact Only) The device includes signature data inside **EMV Batch Data Type** tag DFDF3E.
10	02	07	06	ICC, Kernel Outcome, Quick Chip Deferred, Signature Capture Failed (EMV Contact Only)
10	02	08	00	ICC, Kernel Outcome, Failed, Reserved (EMV Contact Only)
10	02	09	00	ICC, Kernel Outcome, Not Accepted, Reserved (EMV Contact Only)
10	02	0A	00	ICC, Kernel Outcome, Transaction Canceled, Reserved (EMV Contact Only)
10	02	0B	00	ICC, Kernel Outcome, Select Next Not Accepted, Reserved (EMV Contact Only)
10	02	0C	00	ICC, Kernel Outcome, Select Next Retry, Reserved (EMV Contact Only)
10	02	0D	00	ICC, Kernel Outcome, Try Again, Reserved (EMV Contact Only)
10	02	0E	00	ICC, Kernel Outcome, Online Request, Reserved (EMV Contact Only)
10	02	0F	00	ICC, Kernel Outcome, End Application, Reserved (EMV Contact Only)
10	02	10	00	ICC, Kernel Outcome, Not EMV Card Pooled, Reserved (EMV Contact Only)
10	02	80	00	ICC, Kernel Outcome, Declined, Reserved (EMV Contact Only)
10	02	80	01	ICC, Kernel Outcome, Declined, Signature Capture Requested (EMV Contact Only)
10	02	80	05	ICC, Kernel Outcome, Declined, Signature Capture Success (EMV Contact Only) The device includes signature data inside **EMV Batch Data Type** tag DFDF3E.
10	02	80	06	ICC, Kernel Outcome, Declined, Signature Capture Failed (EMV Contact Only)
10	04	07	00	ICC, Transaction Completed, Declined, Reserved (EMV Contact Only)
Payment Technology **0x20 EMV Contactless** (“PICC”) contains transaction notification detail codes involving contactless cards and contactless payment devices (EMV Contactless Only). • 0x00 = Reserved • 0x01 = Timeout, no cards detected • 0x02 = Kernel Outcome • 0x03 = Transaction Terminated • 0x04 = Transaction Completed				
20	02	00	00	PICC, Kernel Outcome, None, Reserved (EMV Contactless Only)
20	02	01	00	PICC, Kernel Outcome, Try Another Interface, Reserved (EMV Contactless Only)
20	02	02	00	PICC, Kernel Outcome, Offline Approved, Reserved (EMV Contactless Only)
20	02	02	01	PICC, Kernel Outcome, Offline Approved, Signature Capture Requested (EMV Contactless Only)

PT	Res	Det	Ext	Meaning
20	02	02	05	PICC, Kernel Outcome, Offline Approved, Signature Capture Success (EMV Contactless Only) The device includes signature data inside **EMV Batch Data Type** tag DFDF3E.
20	02	02	06	PICC, Kernel Outcome, Offline Approved, Signature Capture Fail (EMV Contactless Only)
20	02	03	00	PICC, Kernel Outcome, Offline Declined, Reserved (EMV Contactless Only)
20	02	04	00	PICC, Kernel Outcome, Offline Failed, Reserved (EMV Contactless Only)
20	02	05	00	PICC, Kernel Outcome, Offline Not Accepted, Reserved (EMV Contactless Only)
20	02	06	00	PICC, Kernel Outcome, Approved, Reserved (EMV Contactless Only)
20	02	06	01	PICC, Kernel Outcome, Approved, Signature Capture Requested (EMV Contactless Only)
20	02	06	05	PICC, Kernel Outcome, Approved, Signature Capture Success (EMV Contactless Only) The device includes signature data inside **EMV Batch Data Type** tag DFDF3E.
20	02	06	06	PICC, Kernel Outcome, Approved, Signature Capture Fail (EMV Contactless Only)
20	02	07	00	PICC, Kernel Outcome, Quick Chip Deferred, Reserved (EMV Contactless Only)
20	02	07	01	PICC, Kernel Outcome, Quick Chip Deferred, Signature Capture Requested (EMV Contactless Only)
20	02	07	05	PICC, Kernel Outcome, Quick Chip Deferred, Signature Capture Success (EMV Contactless Only) The device includes signature data inside **EMV Batch Data Type** tag DFDF3E.
20	02	07	06	PICC, Kernel Outcome, Quick Chip Deferred, Signature Capture Failed (EMV Contactless Only)
20	02	08	00	PICC, Kernel Outcome, Failed, Reserved (EMV Contactless Only)
20	02	09	00	PICC, Kernel Outcome, Not Accepted, Reserved (EMV Contactless Only)
20	02	0A	00	PICC, Kernel Outcome, Transaction Canceled, Reserved (EMV Contactless Only)
20	02	0B	00	PICC, Kernel Outcome, Select Next Not Accepted, Reserved (EMV Contactless Only)
20	02	0C	00	PICC, Kernel Outcome, Select Next Retry, Reserved (EMV Contactless Only)
20	02	0D	00	PICC, Kernel Outcome, Try Again, Reserved (EMV Contactless Only)
20	02	0E	00	PICC, Kernel Outcome, Online Request, Reserved (EMV Contactless Only)
20	02	0F	00	PICC, Kernel Outcome, End Application, Reserved (EMV Contactless Only)
20	02	10	00	PICC, Kernel Outcome, Not EMV Card Pooled, Reserved (EMV Contactless Only)

PT	Res	Det	Ext	Meaning
20	02	80	00	PICC, Kernel Outcome, Declined, Reserved (EMV Contactless Only)
20	02	80	01	PICC, Kernel Outcome, Declined, Signature Capture Requested (EMV Contactless Only)
20	02	80	05	PICC, Kernel Outcome, Declined, Signature Capture Success (EMV Contactless Only) The device includes signature data inside **EMV Batch Data Type** tag DFDF3E.
20	02	80	06	PICC, Kernel Outcome, Declined, Signature Capture Failed (EMV Contactless Only)
20	04	07	00	PICC, Transaction Completed, Declined, Reserved (EMV Contactless Only)
20	05	00	00	PICC, NFC TAG, Tag Removed, Reserved (EMV Contactless Only)

Table 7.1-8 - Notification Example

Example (Hex)
AA 00 81 04 83 00 01 05 82 04 20 00 00 00

7.2 Notification Source 0x09nn - Notifications from Firmware Updates

7.2.1 Notification 0x0905 - Firmware Update Successful

This notification reports the successful completion of firmware update operations the host initiated using **Command 0xD801 - Load Firmware File** and **Command 0xD901 - Commit Firmware from File**. The **Notification Detail** for those two commands are shown in the table below.

Table 7.2-1 - Notification Detail Codes

B1	B2	B3	B4	Meaning
08	01	0A	03	Commit Firmware from File Complete, Success
08	01	09	03	Load Firmware File Complete, Success

7.2.2 Notification 0x0906 - Firmware Update Failed

This notification reports the failure of a Firmware Update command the host initiated using **Command 0xD801 - Load Firmware File** and **Command 0xD901 - Commit Firmware from File**. The **Notification Detail** for those two commands are shown in the table below.

Table 7.2-2 - Notification Detail Codes

B1	B2	B3	B4	Meaning
08	01	0A	04	Commit Firmware from File Complete, Failed
08	01	09	04	Load Firmware File Complete, Failed

7.2.3 Notification 0x0907 - Firmware is Up to Date

This notification reports Firmware is up to date for host initiated **Command 0xD801 - Load Firmware File**. The **Notification Detail** for this command is shown in the table below.

Table 7.2-3 - Notification Detail Codes

B1	B2	B3	B4	Meaning
08	01	09	03	Load Firmware File Complete, Complete

7.3 Notification Source 0x10nn - Notifications from Device

7.3.1 Notification 0x1001 - Device Information Update

This notification reports information about general state changes that occur within the device, outside the context of specific operations like transactions (**Notification Source 0x01nn - Notifications from Transactions**), requests the host receives from the firmware's integrated display interface (**Notification Source 0x18nn - Notifications from User Interface**), and so on.

For this notification, **Notification Detail** described in section **3.2.2.3 Notification Message** contains one byte each, shown in **Table 7.3-1 below**, to indicate:

- The **Category** (Cat) of notification (for example, the subsystem the notification is originating from)
- The **Reason** (Rsn) for the notification (Device Event)
- **Detail** (Det) about the notification that has different meanings depending on the Reason, and
- An **Extra** field (Ext) that has different meanings depending on the Reason.

The other optional message parameters in the **Notification Message** depend on which combination of four bytes is included in **Notification Detail**, and are described in **Table 7.3-1**.

Table 7.3-1 - Notification Detail Codes

Cat	Rsn	Det	Ext	Meaning
Category **0x00 Power / Reset** contains notification detail codes involving the device's power and reset functionality. Each possible notification has a unique **Reason** value: • Reason 0x00 = Device Reset Occurred • Reason 0x01 = Device Reset Will Occur Soon • Reason 0x02 = Low Battery • Reason 0x03 = Key Management • Reason 0x04 = Temperature				
00	00	00	00	Power/Reset, Device Reset Occurred, Reserved, Reserved The device sends and repeats this notification after the device power cycles or resets, depending on the setting in **Property 1.2.7.1.1.1 Device Reset Occurred Notification Control**. If it is set to repeat, it does so until the host acknowledges it using **Property 1.2.7.1.1.2 Device Reset Occurred Notification Acknowledged**. These notifications always include the **Notification Payload** parameter in the **Notification Message**, as shown in **Table 7.4-1**.

Cat	Rsn	Det	Ext	Meaning
00	01	00	00	Power/Reset, Device Reset Will Occur Soon, Reserved, Reserved The device sends this notification before it automatically resets to conform to PCI's 24 hour Self-Test requirement, and behaves according to the setting in **Property 1.2.7.1.1.3 Device Reset Will Occur Soon Notification Control**. See **24 Hour Automatic Reset PCI Requirement** for more information. These notifications always include the **Notification Payload** parameter in the **Notification Message**, which directly contains one byte indicating the number of minutes (0x01 to 0xFF) until the device will perform the reset.
00	02	00	00	Power/Reset, Battery, Low Battery Warning, Percent The device sends this notification when the battery charge reaches 15 percent. If a device is powered on with a charge that is already 15 percent or below, this notification is sent shortly after power up and includes the current battery charge percentage. Percent indicates the percent of battery charge remaining.
00	02	01	00	Power/Reset, Battery, Power Down Imminent, Reserved The device sends this notification one minute before it automatically powers down the device because the battery charge has reached 0 percent.
00	02	02	00	Power/Reset, Battery, Battery Charge Complete, Reserved The device sends this notification when the battery charger detects that the battery is fully charged.
00	03	00	00	Power/Reset, Device reset per Command 0x1F01. The device sends this notification to notify the host when Device executes Reset Command 0x1F01.

Cat	Rsn	Det	Ext	Meaning
Category **0x01 User Event** contains notification detail codes involving events triggered by user actions. Each possible notification has a unique **Reason** value: • Reason 0x00 = Contactless Card Presented (EMV Contactless Only) • Reason 0x01 = Contactless Card Removed (EMV Contactless Only) • Reason 0x02 = Card Seated in Slot (EMV Contact Only) • Reason 0x03 = Card Unseated from Slot (EMV Contact Only • Reason 0x07 = Barcode Read (BCR Only) The host can use **Property 1.2.7.1.2.1 User Event Notification Controls** Enable to enable these notification reasons individually. The host may choose to use these notifications to determine when to send additional commands. For example, it may send **Command 0x1001 - Start Transaction**. The host should do this as quickly as possible to minimize the response time between when the cardholder presents a card and when the device provides feedback via a visible or audible state change or attempts to read a chip card. Note there are cases where the device may send a notification with **Reason** = **Contactless Card Presented** while the cardholder is inserting or swiping a card, so the host should start transactions with all supported card interfaces enabled to maximize the chances of a successful card read.				
01	00	00	00	User Event, Contactless Card Presented, EMV, Reserved (EMV Contactless Only)
01	01	00	00	User Event, Contactless Card Removed, EMV, Reserved (EMV Contactless Only)
01	00	01	00	User Event, Contactless Card Presented, NTag/MIFARE Ultralite, Reserved (EMV Contactless Only) The contactless reader has successfully read a NFC tag. In this case, the device includes UID data in the **Notification Payload** portion of the **Notification Message**. UID Data is in the format described in **Table 7.3-7 - Notification Payload for UID**. A sample notification is shown here: **Table 7.3-8 – Notification Payload for UID Example**
01	01	01	00	User Event, Contactless Card Removed, NTag/MIFARE Ultralite, Reserved (EMV Contactless Only)
01	00	02	00	User Event, Contactless Card Presented, MIFARE Classic, 1K (EMV Contactless Only) The contactless reader has successfully read a NFC tag. In this case, the device includes UID data in the **Notification Payload** portion of the **Notification Message**. UID Data is in the format described in **Table 7.3-7 - Notification Payload for UID**. A sample notification is shown here: **Table 7.3-8 – Notification Payload for UID Example**
01	01	02	00	User Event, Contactless Card Removed, MIFARE Classic, 1K (EMV Contactless Only)

Cat	Rsn	Det	Ext	Meaning
01	00	03	00	User Event, Contactless Card Presented, MIFARE Classic, 4K (EMV Contactless Only) The contactless reader has successfully read a NFC tag. In this case, the device includes UID data in the **Notification Payload** portion of the **Notification Message**. UID Data is in the format described in **Table 7.3-7 - Notification Payload for UID**. A sample notification is shown here: **Table 7.3-8 – Notification Payload for UID Example**
01	01	03	00	User Event, Contactless Card Removed, MIFARE Classic, 4K (EMV Contactless Only)
01	00	04	00	User Event, Contactless Card Presented, MIFARE DESFire Light (EMV Contactless Only) The contactless reader has successfully read a NFC tag. In this case, the device includes UID data in the **Notification Payload** portion of the **Notification Message**. UID Data is in the format described in **Table 7.3-7 - Notification Payload for UID**. A sample notification is shown here: **Table 7.3-8 – Notification Payload for UID Example**
01	01	04	00	User Event, Contactless Card Removed, MIFARE DESFire Light (EMV Contactless Only)
01	00	05	00	User Event, Contactless Card Presented, MIFARE MINI® (EMV Contactless Only) The contactless reader has successfully read a NFC tag. In this case, the device includes UID data in the **Notification Payload** portion of the **Notification Message**. UID Data is in the format described in **Table 7.3-7 - Notification Payload for UID**. A sample notification is shown here: **Table 7.3-8 – Notification Payload for UID Example**
01	01	05	00	User Event, Contactless Card Removed, MIFARE MINI® (EMV Contactless Only)
01	00	06	00	User Event, Contactless Card Presented, MIFARE Plus EV1 (EMV Contactless Only) The contactless reader has successfully read an NFC tag. In this case, the device includes UID data in the **Notification Payload** portion of the **Notification Message**. UID Data is in the format described in **Table 7.3-7 - Notification Payload for UID**. A sample notification is shown here: **Table 7.3-8 – Notification Payload for UID Example**

Cat	Rsn	Det	Ext	Meaning
01	01	06	00	User Event, Contactless Card Removed, MIFARE Plus EV1 (EMV Contactless Only)
01	00	07	00	User Event, Contactless Card Presented, MIFARE Plus EV2 (EMV Contactless Only) The contactless reader has successfully read an NFC tag. In this case, the device includes UID data in the **Notification Payload** portion of the **Notification Message**. UID Data is in the format described in **Table 7.3-7 - Notification Payload for UID.** A sample notification is shown here: **Table 7.3-8 – Notification Payload for UID Example**
01	01	07	00	User Event, Contactless Card Removed, MIFARE Plus EV2 (EMV Contactless Only)
01	00	08	00	User Event, Contactless Card Presented, MIFARE Plus SE (EMV Contactless Only) The contactless reader has successfully read an NFC tag. In this case, the device includes UID data in the **Notification Payload** portion of the **Notification Message**. UID Data is in the format described in **Table 7.3-7 - Notification Payload for UID**. A sample notification is shown here: **Table 7.3-8 – Notification Payload for UID Example**
01	01	08	00	User Event, Contactless Card Removed, MIFARE Plus SE (EMV Contactless Only)
01	00	09	00	User Event, Contactless Card Presented, MIFARE Plus X (EMV Contactless Only) The contactless reader has successfully read an NFC tag. In this case, the device includes UID data in the **Notification Payload** portion of the **Notification Message**. UID Data is in the format described in **Table 7.3-7 - Notification Payload for UID**. A sample notification is shown here: **Table 7.3-8 – Notification Payload for UID Example**
01	01	09	00	User Event, Contactless Card Removed, MIFARE Plus X (EMV Contactless Only)

Cat	Rsn	Det	Ext	Meaning
01	00	0A	00	User Event, Contactless Card Presented, MIFARE DESFire EV1 (EMV Contactless Only) The contactless reader has successfully read a NFC tag. In this case, the device includes UID data in the **Notification Payload** portion of the **Notification Message**. UID Data is in the format described in **Table 7.3-7 - Notification Payload for UID**. A sample notification is shown here: **Table 7.3-8 – Notification Payload for UID Example**
01	01	0A	00	User Event, Contactless Card Removed, MIFARE DESFire EV1 (EMV Contactless Only)
01	00	0B	00	User Event, Contactless Card Presented, MIFARE DESFire EV2 (EMV Contactless Only) The contactless reader has successfully read a NFC tag. In this case, the device includes UID data in the **Notification Payload** portion of the **Notification Message**. UID Data is in the format described in **Table 7.3-7 - Notification Payload for UID**. A sample notification is shown here: **Table 7.3-8 – Notification Payload for UID Example**
01	01	0B	00	User Event, Contactless Card Removed, MIFARE DESFire EV2 (EMV Contactless Only)
01	00	0C	00	User Event, Contactless Card Presented, MIFARE DESFire EV3 (EMV Contactless Only) The contactless reader has successfully read a NFC tag. In this case, the device includes UID data in the **Notification Payload** portion of the **Notification Message**. UID Data is in the format described in **Table 7.3-7 - Notification Payload for UID**. A sample notification is shown here: **Table 7.3-8 – Notification Payload for UID Example**
01	01	0C	00	User Event, Contactless Card Removed, MIFARE DESFire EV3 (EMV Contactless Only)
01	02	00	00	User Event, Card Seated in Slot, Reserved, Reserved (EMV Contact Only)
01	03	00	00	User Event, Card Unseated from Slot, Reserved, Reserved (EMV Contact Only)

Cat	Rsn	Det	Ext	Meaning
01	07	00	00	Barcode Reader, Read Barcode Result, Type, Reserved (BCR Only) • Type 0x00 = Unknown • Type 0x01 = MagTek • Type 0x02 = EMV The barcode reader has successfully read a barcode. In this case, the device includes barcode data in the **Notification Payload** portion of the **Notification Message**. If the data is encrypted, the data is in the format described in **Table 7.4-5**. Data that is not encrypted is in the format described in **Table 7.4-6**.
Category **0x04 Temperature** contains notification detail codes involving temperature reporting. Each possible notification has a unique **Reason** value: • Reason 0x00 = Temperature out of range				
04	00	00	00	Temperature, Out of Range, Low Warning, Temperature The device sends this notification when the device's temperature falls below the temperature set in Low Temperature Notification Level (1.2.7.1.4.1). The temperature reported is in Celsius.
04	00	01	00	Temperature, Out of Range, High Warning, Temperature The device sends this notification when the device's temperature rises above the temperature set in High Temperature Notification Level (1.2.7.1.4.2). The temperature reported is in Celsius.

7.3.1.1 Notification 0x1001 – Low Battery Warning

The device sends this notification when the battery charge reaches 15 percent. If a device is powered on with a charge that is already 15 percent or below, this notification is sent shortly after power up and includes the current battery charge percentage. Percent indicates the percent of battery charge remaining. It is recommended that the device is charged using a USB power source soon after receiving this notification. See **Table 3.2-5 - Notification Message Format** and **Table 7.3-2 Low Battery Notification Example**.

Table 7.3-2 Low Battery Notification Example

Example (Hex)
AA 00 81 04 83 00 10 01 82 00 02 00 0D

7.3.1.2 Notification 0x1001 - Low Battery Shutdown

The device sends this notification one minute before it automatically powers down the device. This occurs when the battery charge has reached 0 percent. Shutdown can be prevented by connecting the device to a USB power source. See **Table 3.2-5 - Notification Message Format** and **Table 7.3-3 Low Battery Shutdown Response Example**.

Table 7.3-3 Low Battery Shutdown Response Example

Example (Hex)
AA 00 81 04 83 00 10 01 82 00 02 01 00

7.3.1.3 Notification 0x1001 - Battery Charged

The device sends this notification when the battery charger detects that the battery is fully charged. This only occurs when the device is powered by a USB power source. See **Table 3.2-5 - Notification Message Format** and **Table 7.3-4 - Battery Charged Notification Example**.

Table 7.3-4 - Battery Charged Notification Example

Example (Hex)
AA 00 81 04 83 00 10 01 82 00 02 02 00

Table 7.3-5 - Notification Payload for Device Reset Occurred

Tag	Len	Value / Description	Typ	Req	Default
Beginning of **Notification Message** found on page **38**					
1001 = Device Information Update					
81	var	This parameter has the same length and value described by **Property 2.3.1.2.1.1 Device Operational Status**.	B	R	
82	var	This parameter has the same length and value described by **Property 2.3.1.2.1.2 Offline Status Detail**.	B	R	
xx	var	More data objects may be added in future firmware revisions.			
End of **Notification Message** found on page **38**					

Table 7.3-6 - Notification Example for Device Reset Occurred

Example (Hex)
AA00 81 04 83001001 82 04 00000000 84 08 1001 81 01 02 82 01 00

Table 7.3-7 - Notification Payload for UID

Tag	Len	Value / Description	Typ	Req	Default
Beginning of **Notification Message** found on page **38**					
1001 = Device Information Update					
84	var	NFC Payload	B	R	
/DF79	var	NFC UID	B	R	
End of **Notification Message** found on page **38**					

Table 7.3-8 – Notification Payload for UID Example

Example (Hex)
AA 00 81 04 83 00 10 01 82 04 01 00 01 00 84 0E 10 01 84 0A DF 79 07 04 37 1F 71 BF 61 80

7.4 Notification Source 0x18nn - Notifications from User Interface

7.4.1 Notification 0x1803 - User Interface Host Action Request

This notification requests that the host take action during operations involving the device's User Interface modules.

For this notification, **Notification Detail** described in section **3.2.2.3 Notification Message** contains one byte each, shown in **Table 7.4-1**, to indicate:

- The **User Interface Module** (UI) involved
- The **Reason** (Rsn) for the notification (UI Event)
- **Detail** (Det) about the notification that has different meanings depending on the Reason, and
- An **Extra** field (Ext) that has different meanings depending on the Reason.

Table 7.4-1 - Notification Detail Codes

UI	Rsn	Det	Ext	Meaning
Module **0x00 Miscellaneous** contains UI notification detail codes that are not specific to a particular UI module.				
Module **0x02 Display** contains UI notification detail codes that are specific to the display module. The device primarily uses these notifications to request that the host stand in as a display to interact with cardholders / operators when the device does not have a display of its own (No Display Only). • Reason 0x01 = Display Message Request • Reason 0x02 = Cardholder Selection Request				
02	01	00	00	Display, Display Message, No Data Attached, Clear Display. The host should clear any existing messages it is currently displaying (No Display Only).
02	01	01	00	Display, Display Message, Data Attached, No Cancel Function (No Display Only) In this case, the notification includes additional data and display control, defined in **Table 7.4-2**, in the **Notification Payload** portion of the **Notification Message**. This notification is for simple messages. The host should show the requested message until another notification arrives or until the host needs to use the display for its own purposes. For example, during a transaction, this notification will pass along the simple messages in the transaction sequence like **CARD READ OK - REMOVE CARD**, **THANK YOU**, and **WELCOME**.
02	01	02	00	Display, Display Message, Data Attached, Include Cancel Function (No Display Only) In this case, the notification includes additional data and display control, defined in **Table 7.4-2**, in the **Notification Payload** portion of the **Notification Message**. The host should show the requested message and should allow the request to be canceled by a cardholder or operator. For example, during a transaction, this notification will pass along the simple messages in the transaction sequence like "TAP, INSERT, or SWIPE CARD," where cancelation is appropriate. If the cancel button involves canceling the whole transaction, the host should send **Command 0x1008 - Cancel Transaction**.

UI	Rsn	Det	Ext	Meaning
02	02	00	00	Display, Cardholder Selection, Data Attached, Reserved (No Display Only) In this case, the notification includes additional data and display control, defined in **Table 7.4-3**, in the **Notification Payload** portion of the **Notification Message**. The host should show the requested selection, then send **Command 0x1802 - Report Cardholder Selection** to device to return the cardholder's selection to the device.

Table 7.4-2 - Notification Payload for Display Message Request Notifications (No Display Only)

Tag	Len	Value / Description	Typ	Req	Default
Beginning of **Notification Message** found on page **38**					
1803 = **Notification 0x1803 - User Interface Host Action Request**					
/81	01	Reserved for time functions. 0x00	B	R	
/82	01	Append or Clear Display • 0x00 = Append the message(s) on the next available row. • 0x01 = Clear the display first and display the requested message on the first row.	B	R	
/83	var	Message This parameter is a set of one or more ASCII strings the host should show, with each string terminated by any one of the following characters. MagTek recommends proceeding to the next row on receiving any of these characters: • Null (0x00) • LF (0x0A) • The end of the Message parameter.	B	R	
/85	var	Enhanced Application Select Message For Contactless, this parameter will return the PPSE response starting from tag BF0C BF0C <LEN><VALUE> For Contact, this parameter will return all available AID returned from the PSE or List Of AIDs, from tag 70 70 <len> 61 <len> 4F<len><value of AID> 50<len><value of Application Label> 87<len><value of Application Priority Indicator> 61 <len> 4F<len><value of AID> 50<len><value of Application Label> 87<len><value of Application Priority Indicator, optional>	B	O	

Tag	Len	Value / Description	Typ	Req	Default
		If the GPO Response returns a **Notification 0x0101 - Transaction Information Update of**'69 85', the response will be sent first by the device and this Enhanced Application Select Message will be sent again.			
End of **Notification Message** found on page **38.**					

Table 7.4-3 - Notification Payload for Cardholder Selection Notifications (No Display Only)

Tag	Len	Value / Description	Typ	Req	Default
Beginning of **Notification Message** found on page **38**					
1803 = **Notification 0x1803 - User Interface Host Action Request**					
/81	01	Time in seconds the device will wait for host to respond with **Command 0x1802 - Report Cardholder Selection**.	B	R	
/82	01	Append or Clear Display • 0x00 = Append, display the messages on the next line. • 0x01 = Clear, clear the display first and display the requested message on the first line.	B	R	
/83	var	Message This parameter is a set of one or more ASCII strings the host should show, with each string terminated by any one of the following characters. MagTek recommends proceeding to the next row on receiving any of these characters: • Null (0x00) • LF (0x0A) • The end of the Message parameter.	B	R	
End of **Notification Message** found on page **38**					

7.4.2 Notification 0x1805 - User Interface Operation Complete

This notification reports information about progress and state changes that occur during interaction with the User Interface modules.

For this notification, **Notification Detail** described in section **3.2.2.3 Notification Message** contains one byte each, shown in **Table 7.4-4**, to indicate:

- The **User Interface Module** (UI) involved
- The **Reason** (Rsn) for the notification (UI Event)
- **Detail** (Det) about the notification that has different meanings depending on the Reason, and
- An **Extra** field (Ext) that has different meanings depending on the Reason.

Table 7.4-4 - Notification Detail Codes

UI	Rsn	Det	Ext	Meaning
Module **0x04 Barcode Reader** contains notification detail codes that are specific to the barcode reader module. • Reason 0x03 = Read Barcode Result				
04	03	01	00	Barcode Reader, Read Barcode Result, Success, Unidentified Type The barcode reader has successfully read a barcode. In this case, the device includes barcode data in the **Notification Payload** portion of the **Notification Message**. If the data is encrypted, the data is in the format described in **Table 7.4-5**. Data that is not encrypted is in the format described in **Table 7.4-6**.
04	03	01	01	Barcode Reader, Read Barcode Result, Success, MagTek Blob Type The barcode reader has successfully read a barcode. In this case, the device includes barcode data in the **Notification Payload** portion of the **Notification Message**. MagTek Blob data is always encrypted. The data is in the format described in **Table 7.4-5**.
04	03	01	02	Barcode Reader, Read Barcode Result, Success, EMV Type The barcode reader has successfully read a barcode. In this case, the device includes barcode data in the **Notification Payload** portion of the **Notification Message**. EMV barcode data is always encrypted. The data is in the format described in **Table 7.4-5.**
04	03	02	01	Barcode Reader, Read Barcode Result, Operation Failed, Timeout
Module **0x05 Buzzer** contains notification detail codes that are specific to the buzzer module. • Reason 0x04 = Buzzer Result				
05	04	01	00	Buzzer, Buzzer Result, Operation Success, Reserve
05	04	02	00	Buzzer, Buzzer Result, Operation Fail, Reserve
Module **0x06 Card Emulation** contains notification detail codes that are specific to card emulation. • Reason 0x05 = Card Emulation Result				
06	05	01	00	Card Emulation, Card Emulation Result, Operation Success, Reserve
06	05	02	01	Card Emulation, Card Emulation Result, Operation Fail, Timeout
06	05	02	03	Card Emulation, Card Emulation Result, Operation Fail, Cancel
06	05	02	04	Card Emulation, Card Emulation Result, Operation Fail, Error

Table 7.4-5 - Notification Payload for Barcode Reader, Read Barcode Result, Success (Encrypted Data Attached)

Tag	Len	Value / Description	Typ	Req	Default
Beginning of **Notification Message** found on page **38**					
1805 = User Interface Operation Complete					
84	var	Notification Payload	B	R	
/DFDF59	var	Encrypted Data Primitive	B	R	

Tag	Len	Value / Description	Typ	Req	Default
		Decrypt the value of this TLV data object using the algorithm and variant specified in the **Encrypted Data KSN** parameter and the **Encrypted Data Encryption Type** parameter to read its contents. The format of the decrypted data is shown in **Table 7.4-6**.			
/DFDF50	var	Encrypted Data KSN	B	R	
/DFDF51	01	Encrypted Data Encryption Type See section **4.4 Encryption Type** for a list of valid values.	B	R	
End of **Notification Message** found on page **38**					

Table 7.4-6 - Notification Payload for Barcode Reader, Read Barcode Result, Success (Unencrypted Data)

Tag	Len	Value / Description	Typ	Req	Default
FC	var	Barcode Data Container	T	R	
/DF74	var	Barcode Data	B	O	

Table 7.4-7 – Default User Interface String IDs and Strings

Display String ID	Display String
0x0000	" "
0x0001	"Thank You"
0x0002	"Hello"
0x0003	"Please Select"
0x0004	"Touch Screen To Continue"
0x0005	"Cancel"
0x0006	"Select a Transaction"
0x0007	"Deposit"
0x0008	"Withdrawal"
0x0009	"Choose PIN"
0x000A	"Change PIN"
0x000B	"Balance"
0x000C	"Cash Advance"
0x000D	"Authenticate Me"
0x000E	"Transfer"
0x000F	"Make Payment"
0x0010	"On Sale"
0x0011	"Discount"

Display String ID	Display String
0x0012	“Clearance”
0x0013	“$5.00 /5 Mins.”
0x0014	“$6.00 /7 Mins.”
0x0015	“$7.00 /9 Mins.”
0x0016	“$8.00 /15 Mins.”
0x0017	“Call Attendant”
0x0018	“Print Receipt”
0x0019	“Email Receipt”
0x001A	“Text Receipt”
0x001B	“Please Select An Amount”
0x001C	“Please Select A Tip Amount”
0x001D	“Menu Selection 1”
0x001E	“Menu Selection 2”
0x001F	“Menu Selection 3”
0x0020	“Menu Selection 4”
0x0021	“Service Request”
0x0022	“Show More”
0x0023	“Options”
0x0024	“What is the issue?”
0x0025	“Exit”
0x0026	“No Hot Water”
0x0027	“Doesn't Spin”
0x0028	“Water Leakage”
0x0029	“No Heat”
0x002A	“Please select a Machine”
0x002B	“Pump 1”
0x002C	“Pump 2”
0x002D	“Pump 3”
0x002E	“Pump 4”
0x002F	“A1”
0x0030	“A2”
0x0031	“A3”
0x0032	“A4”

Display String ID	Display String
0x0033	“B1”
0x0034	“B2”
0x0035	“B3”
0x0036	“B4”
0x0037	“C1”
0x0038	“C2”
0x0039	“C3”
0x003A	“C4”
0x003B	“$5.50 – Cake”
0x003C	“$5.00 – Muffin”
0x003D	“$6.50 – Croissant”
0x003E	“$4.50 – Danish”
0x003F	“Req. QR Code”
0x0040	“Scan QR Code”
0x0041	“Scan NFC”
0x0042	“Call Server”
0x0043	“Request Bill”
0x0044	“Approve”
0x0045	“Reject”
0x0046	“Verify SMS”
0x0047	“Verify Email”
0x0048	“English”
0x0049	“Spanish”
0x0080	“Hola”
0x0081	“Por Favor Seleccione”
0x0082	“Toca Para Continuar”
0x0083	“Cancelar”
0x0084	“Seleccione Una Transaccion”
0x0085	“Deposito”
0x0086	“Retiro”
0x0087	“Elija PIN”
0x0088	“Cambiar PIN”
0x0089	“Adelanto En Efectivo”

Display String ID	Display String
0x008A	“Autenticarme”
0x008B	“Verificar SMS”
0x008C	“Confirmar”
0x008D	“Mostrar Mas”
0x008E	“Opciones”
0x008F	“Salida”
0x0090	“Transferir”
0x0091	“Realizar Pago”
0x0092	“En Venta”
0x0093	“Descuento”
0x0094	“Autorizacion”
0x0095	“Operador De Llamada”
0x0096	“Imprimir Recibo”
0x0097	“Correo Electronico”
0x0098	“Recibo De Texto”
0x0099	“Seleccione Una Cantidad”
0x009A	“Anadir Un Consejo”
0x009B	“Requerido Codigo QR”
0x009C	“Escanear Codigo QR”
0x009D	“Escanear NFC”
0x009E	“Servidor De Llamadas”
0x009F	“Solicitar Factura”
0x00A0	“Aprobar”
0x00A1	“Rechazar”
0x00A2	“Gracias”
0x00A3	“Ingles”
0x00A4	“Espanol”

String ID value can be from 0x0000 to 0x012B, and String length maximum is 30 characters. This default configured Strings can be changed and uploaded to the device.

8 Configuration

8.1 About Configuration

The device provides two mechanisms for the host to retrieve information about the device and to configure / customize device behavior:

- **Properties** are smaller configuration values (generally up to 255 bytes and often significantly smaller). The device stores properties in a hierarchical tree, where each property has its own unique ID (sometimes referred to as an Object Identifier or OID) that identifies its position in the tree. The host can set and get properties using the commands in Command Group 0xD1nn - Settings and Information. For further information about OIDs, see **Appendix A Object IDs (OIDs) and ITU-T X660**.
- **Files** are larger configuration values, which include custom graphics files and large configuration data blobs such as EMV terminal and application settings. Each file has a unique identifier more akin to a file path in a file system. The host can set and get files using the commands in **Command Group 0xD8nn - File Operations**.

This section focuses on **properties**, which the host can either read from the device to retrieve device information (such as its PCI hardware ID, serial number, firmware revision numbers, etc.), or write to the device to change device behavior, or to store values for future reference.

Property unique IDs represent a tree structure, where each property is at the "leaf" level of the tree. Some operations allow the host to work with entire groups of properties (branches or even the trunk containing all its leaves) by using a property number representing a higher level of the tree.

Each property (a "leaf" in the tree) is described in a subsection below, including whether it can be written to, how it is secured, how to construct or interpret its value(s), and how it affects device behavior.

To reflect this hierarchy, this section of the document groups settings by major categories that correspond to the tree structure, which roughly breaks down into overall property **type**, **module**, and **submodule**.

8.2 Property Group 1.1.nnnn Financial Settings

8.2.1 Property Subgroup 1.1.1.nnn EMV Settings

8.2.1.1 Property 1.1.1.1.1.1 EMV Terminal Identification

Table 8.2-1 - Property 1.1.1.1.1.1 EMV Terminal Identification

Property Description	
Property OID	1.1.1.1.1.1 / 0x010101010101
Name	EMV Terminal Identification
Description	The device uses this property to report its Terminal ID / Terminal Identification in tag **9F1C** in all EMV-related communication. Merchants usually configure each device with a different terminal ID per their own proprietary system standards to help identify the source of a transaction.
Securing Key	None
Min. Len (b)	8

Property Description	
Max. Len (b)	8
Data Type	Alphanumeric
Valid Values	Any string
Default	“00000000” (0x3030303030303030)

Table 8.2-2 - Get Request Example

Example (Hex)
AA 00 81 04 01 04 D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 01 89 0A E1 08 E1 06 E1 04 E1 02 C1 00

Table 8.2-3 - Get Response Example

Example (Hex)
AA 00 81 04 82 04 D1 01 82 04 00 00 00 00 84 82 00 22 D1 01 81 07 2B 06 01 04 01 F6 09 85 01 01 89 12 E1 10 E1 0E E1 0C E1 0A C1 08 30 30 30 30 30 30 30 30

Table 8.2-4 - Set Request Example

Example (Hex)
AA 00 81 04 01 3E D1 11 84 22 D1 11 81 07 2B 06 01 04 01 F6 09 85 01 01 89 12 E1 10 E1 0E E1 0C E1 0A C1 08 31 31 31 31 31 31 31 31

Table 8.2-5 - Set Response Example

Example (Hex)
AA 00 81 04 82 3E D1 11 82 04 00 00 00 00 84 82 00 22 D1 11 81 07 2B 06 01 04 01 F6 09 85 01 01 89 12 E1 10 E1 0E E1 0C E1 0A C1 08 31 31 31 31 31 31 31 31

8.2.1.2 Property 1.1.1.1.1.2 EMV ARQC Message Tag List

Table 8.2-6 - Property 1.1.1.1.1.2 EMV ARQC Message Tag List

Property Description	
Property OID	1.1.1.1.1.2 / 0x010101010102
Name	EMV ARQC Message Tag List
Description	The device uses this property when it generates an **EMV ARQC Type** data object, to determine what tags to include in the data object. It serves the same purpose that MagTek custom tag DFDF02 serves on some other MagTek devices.
Securing Key	None
Min. Len (b)	0
Max. Len (b)	255
Data Type	Binary
Valid Values	List of any valid standard EMV message tags and device custom tags

Property Description	
Default	50 5A 57 82 84 95 9A 9B 9C F4 5F 1A 5F 20 5F 24 5F 2A 5F 30 5F 34 9F 02 9F 03 9F 06 9F 09 9F 10 9F 15 9F 1A 9F 1D 9F 1E 9F 24 9F 26 9F 27 9F 33 9F 34 9F 35 9F 36 9F 37 9F 39 9F 41 9F 45 9F 4C 9F 4E 9F 5D 9F 66 9F 6E 9F 7C

Table 8.2-7 - Get Request Example

Example (Hex)
AA 00 81 04 01 08 D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 01 89 0A E1 08 E1 06 E1 04 E1 02 C2 00

Table 8.2-8 - Get Response Example

Example (Hex)
AA 00 81 04 82 08 D1 01 82 04 00 00 00 00 84 82 00 64 D1 01 81 07 2B 06 01 04 01 F6 09 85 01 01 89 54 E1 52 E1 50 E1 4E E1 4C C2 4A 50 5A 57 82 84 95 9A 9B 9C F4 5F 1A 5F 20 5F 24 5F 2A 5F 30 5F 34 9F 02 9F 03 9F 06 9F 09 9F 10 9F 15 9F 1A 9F 1D 9F 1E 9F 24 9F 26 9F 27 9F 33 9F 34 9F 35 9F 36 9F 37 9F 39 9F 41 9F 45 9F 4C 9F 4E 9F 5D 9F 66 9F 6E 9F 7C

Table 8.2-9 - Set Request Example

Example (Hex)
AA 00 81 04 01 09 D1 11 84 64 D1 11 81 07 2B 06 01 04 01 F6 09 85 01 01 89 54 E1 52 E1 50 E1 4E E1 4C C2 4A 50 5A 57 82 84 95 9A 9B 9C F4 5F 1A 5F 20 5F 24 5F 2A 5F 30 5F 34 9F 02 9F 03 9F 06 9F 09 9F 10 9F 15 9F 1A 9F 1D 9F 1E 9F 24 9F 26 9F 27 9F 33 9F 34 9F 35 9F 36 9F 37 9F 39 9F 41 9F 45 9F 4C 9F 4E 9F 5D 9F 66 9F 6E 9F 7C

Table 8.2-10 - Set Response Example

Example (Hex)
AA 00 81 04 82 09 D1 11 82 04 00 00 00 00 84 82 00 64 D1 11 81 07 2B 06 01 04 01 F6 09 85 01 01 89 54 E1 52 E1 50 E1 4E E1 4C C2 4A 50 5A 57 82 84 95 9A 9B 9C F4 5F 1A 5F 20 5F 24 5F 2A 5F 30 5F 34 9F 02 9F 03 9F 06 9F 09 9F 10 9F 15 9F 1A 9F 1D 9F 1E 9F 24 9F 26 9F 27 9F 33 9F 34 9F 35 9F 36 9F 37 9F 39 9F 41 9F 45 9F 4C 9F 4E 9F 5D 9F 66 9F 6E 9F 7C

8.2.1.3 Property 1.1.1.1.1.3 EMV Batch Data Tag List

Table 8.2-11 - Property 1.1.1.1.1.3 EMV Batch Data Tag List

Property Description	
Property OID	1.1.1.1.1.3 / 0x010101010103
Name	EMV Batch Data Tag List
Description	The device uses this property when it generates an **EMV Batch Data Type** data object to contain **transaction result data**, to determine what tags to include in the data object. It serves the same purpose that MagTek custom tag DFDF17 serves on some other MagTek devices.
Securing Key	None
Min. Len (b)	0

Property Description	
Max. Len (b)	255
Data Type	Binary
Valid Values	List of any valid standard EMV message tags and device custom tags
Default	50 57 5A 82 84 8E 95 9A 9C F4 5F 20 5F 24 5F 25 5F 28 5F 2A 5F 2D 5F 34 9F 02 9F 03 9F 07 9F 09 9F 0B 9F 0D 9F 0E 9F 0F 9F 10 9F 11 9F 12 9F 15 9F 16 9F 1A 9F 1C 9F 1E 9F 21 9F 24 9F 26 9F 27 9F 33 9F 34 9F 35 9F 36 9F 37 DF DF 25

Table 8.2-12 - Get Request Example

Example (Hex)
AA 00 81 04 01 15 D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 01 89 0A E1 08 E1 06 E1 04 E1 02 C3 00

Table 8.2-13 - Get Response Example

Example (Hex)
AA 00 81 04 82 15 D1 01 82 04 00 00 00 00 84 82 00 66 D1 01 81 07 2B 06 01 04 01 F6 09 85 01 01 89 56 E1 54 E1 52 E1 50 E1 4E C3 4C 50 57 5A 82 84 8E 95 9A 9C 5F 20 5F 24 5F 25 5F 28 5F 2A 5F 2D 5F 34 9F 02 9F 03 9F 07 9F 09 9F 0B 9F 0D 9F 0E 9F 0F 9F 10 9F 11 9F 12 9F 15 9F 16 9F 1A 9F 1C 9F 1E 9F 21 9F 24 9F 26 9F 27 9F 33 9F 34 9F 35 9F 36 9F 37 DF DF 25

Table 8.2-14 - Set Request Example

Example (Hex)
AA 00 81 04 01 0B D1 11 84 66 D1 11 81 07 2B 06 01 04 01 F6 09 85 01 01 89 56 E1 54 E1 52 E1 50 E1 4E C3 4C 50 57 5A 82 84 8E 95 9A 9C 5F 20 5F 24 5F 25 5F 28 5F 2A 5F 2D 5F 34 9F 02 9F 03 9F 07 9F 09 9F 0B 9F 0D 9F 0E 9F 0F 9F 10 9F 11 9F 12 9F 15 9F 16 9F 1A 9F 1C 9F 1E 9F 21 9F 24 9F 26 9F 27 9F 33 9F 34 9F 35 9F 36 9F 37 DF DF 25

Table 8.2-15 - Set Response Example

Example (Hex)
AA 00 81 04 82 0B D1 11 82 04 00 00 00 00 84 82 00 66 D1 11 81 07 2B 06 01 04 01 F6 09 85 01 01 89 56 E1 54 E1 52 E1 50 E1 4E C3 4C 50 57 5A 82 84 8E 95 9A 9C 5F 20 5F 24 5F 25 5F 28 5F 2A 5F 2D 5F 34 9F 02 9F 03 9F 07 9F 09 9F 0B 9F 0D 9F 0E 9F 0F 9F 10 9F 11 9F 12 9F 15 9F 16 9F 1A 9F 1C 9F 1E 9F 21 9F 24 9F 26 9F 27 9F 33 9F 34 9F 35 9F 36 9F 37 DF DF 25

8.2.1.4 Property 1.1.1.1.1.4 EMV Reversal Data Tag List (EMV Contact Only)

Table 8.2-16 - Property 1.1.1.1.1.4 EMV Reversal Data Tag List (EMV Contact Only)

Property Description	
Property OID	1.1.1.1.1.4 / 0x010101010104
Name	EMV Reversal Data Tag List

Property Description	
Description	The device uses this property when it generates an **EMV Batch Data Type** data object to contain **reversal data**, to determine what tags to include in the data object. It serves the same purpose that MagTek custom tag DFDF05 serves on some other MagTek devices.
Securing Key	None
Min. Len (b)	0
Max. Len (b)	255
Data Type	Binary
Valid Values	List of any valid standard EMV message tags and device custom tags
Default	57 5A 82 8A 95 9A 9C 5F 24 5F 2A 5F 34 9F 01 9F 02 9F 10 9F 15 9F 16 9F 1A 9F 1C 9F 21 9F 33 9F 35 9F 36 9F 39 9F 5B DF DF 25

Table 8.2-17 - Get Request Example

Example (Hex)
AA 00 81 04 01 16 D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 01 89 0A E1 08 E1 06 E1 04 E1 02 C4 00

Table 8.2-18 - Get Response Example

Example (Hex)
AA 00 81 04 82 16 D1 01 82 04 00 00 00 00 84 82 00 44 D1 01 81 07 2B 06 01 04 01 F6 09 85 01 01 89 34 E1 32 E1 30 E1 2E E1 2C C4 2A 57 5A 82 8A 95 9A 9C 5F 24 5F 2A 5F 34 9F 01 9F 02 9F 10 9F 15 9F 16 9F 1A 9F 1C 9F 21 9F 33 9F 35 9F 36 9F 39 9F 5B DF DF 25

Table 8.2-19 - Set Request Example

Example (Hex)
AA 00 81 04 01 0D D1 11 84 44 D1 11 81 07 2B 06 01 04 01 F6 09 85 01 01 89 34 E1 32 E1 30 E1 2E E1 2C C4 2A 57 5A 82 8A 95 9A 9C 5F 24 5F 2A 5F 34 9F 01 9F 02 9F 10 9F 15 9F 16 9F 1A 9F 1C 9F 21 9F 33 9F 35 9F 36 9F 39 9F 5B DF DF 25

Table 8.2-20 - Set Response Example

Example (Hex)
AA 00 81 04 82 0D D1 11 82 04 00 00 00 00 84 82 00 44 D1 11 81 07 2B 06 01 04 01 F6 09 85 01 01 89 34 E1 32 E1 30 E1 2E E1 2C C4 2A 57 5A 82 8A 95 9A 9C 5F 24 5F 2A 5F 34 9F 01 9F 02 9F 10 9F 15 9F 16 9F 1A 9F 1C 9F 21 9F 33 9F 35 9F 36 9F 39 9F 5B DF DF 25

8.2.1.5 Property 1.1.1.1.1.5 ARPC Receive Timeout

Table 8.2-21 - Property 1.1.1.1.1.5 ARPC Receive Timeout

Property Description	
Property OID	1.1.1.1.1.5 / 0x010101010105

Property Description	
Name	ARPC Receive Timeout
Description	The device uses this property to determine how long to wait for the host to send **Command 0x1004 - Resume Transaction** with ARPC data before it times out and continues a transaction the host started with **Command 0x1001 - Start Transaction**.
Securing Key	None
Min. Len (b)	1
Max. Len (b)	1
Data Type	Binary
Valid Values	0x00 = Infinite timeout 0x01..0xFE = Timeout in seconds 0xFF = Use Transaction Timeout parameter in **Command 0x1001 - Start Transaction**
Default	0xFF

Table 8.2-22 - Get Request Example

Example (Hex)
AA00 8104 0155D101 840F D101 8501 01 8704 01010101 8902 C500

Table 8.2-23 - Get Response Example

Example (Hex)
AA 00 81 04 82 55 D1 01 82 04 00 00 00 00 84 82 00 10 D1 01 85 01 01 87 04 01 01 01 01 89 03 C5 01 FF

Table 8.2-24 - Set Request Example

Example (Hex)
AA00 81040155D111 8410 D111 850101 870401010101 8903 C501FF

Table 8.2-25 - Set Response Example

Example (Hex)
AA 00 81 04 82 55 D1 11 82 04 00 00 00 00 84 82 00 10 D1 11 85 01 01 87 04 01 01 01 01 89 03 C5 01 FF

8.2.1.6 Property 1.1.1.1.1.6 ARPC Retry Attempts

Table 8.2-26 - Property 1.1.1.1.1.6 ARPC Retry Attempts

Property Description	
Property OID	1.1.1.1.1.6 / 0x010101010106
Name	ARPC Retry Attempts

Property Description	
Description	The device uses this property to control its ARPC Retry Attempts. If a timeout occurs when the device is waiting for the host to send **Command 0x1004 - Resume Transaction** with ARPC data, the device re-sends **Notification 0x0101 - Transaction Information Update** to report **ARQC Update** and waits for ARPC data again, up to the number of retries specified by this property. After the final retry, the device continues with the transaction.
Securing Key	None
Min. Len (b)	1
Max. Len (b)	1
Data Type	Binary
Valid Values	0x00..0xFF
Default	0x00

Table 8.2-27 - Get Request Example

Example (Hex)
AA00 8104 0155D101 840F D101 8501 01 8704 01010101 8902 C600

Table 8.2-28 - Get Response Example

Example (Hex)
AA 00 81 04 82 55 D1 01 82 04 00 00 00 00 84 82 00 10 D1 01 85 01 01 87 04 01 01 01 01 89 03 C6 01 00

Table 8.2-29 - Set Request Example

Example (Hex)
AA00 81040155D111 8410 D111 850101 870401010101 8903 C60100

Table 8.2-30 - Set Response Example

Example (Hex)
AA 00 81 04 82 55 D1 11 82 04 00 00 00 00 84 82 00 10 D1 11 85 01 01 87 04 01 01 01 01 89 03 C6 01 00

8.2.1.7 Property 1.1.1.1.1.7 Apple VAS Merchant ID and URL Slot 1

Table 8.2-31 - Property 1.1.1.1.1.7 Apple VAS Merchant ID and URL Slot 1

Property Description	
Property OID	1.1.1.1.1.7 / 0x010101010107
Name	Apple VAS Merchant ID and URL Slot 1
Description	The device uses this property for Apple VAS processing to configure the device's supported Merchant ID and URL.

Property Description	
Securing Key	None
Min. Len (b)	1
Max. Len (b)	102
Data Type	Binary
Valid Values	00 or see **Table 8.2-32 - VAS Merchant ID and URL Type**
Default	0x00

Table 8.2-32 - VAS Merchant ID and URL Type

Tag	Len	Value / Description	Typ	Req	Default
9F25	0x20	Merchant ID	B	O	
9F29	Var up to 0x40	Merchant URL	B	O	

Table 8.2-33 - Get Request Example

Example (Hex)
AA00 8104 0155D101 840F D101 8501 01 8704 01010101 8902 C700

Table 8.2-34 - Get Response Example

Example (Hex)
AA 00 81 04 82 55 D1 01 82 04 00 00 00 00 84 82 00 49 D1 01 85 01 01 87 04 01 01 01 01 89 3C C7 3A 9F 25 20 3E 22 54 3A AF 0A C5 E4 AC FC 25 67 1A 6E BF 6E DE 5A C1 96 74 6C 55 F3 D4 37 08 19 FF F9 22 C3 9F 29 14 77 77 77 2E 65 78 61 6D 70 6C 65 2D 75 72 6C 2E 63 6F 6D

Table 8.2-35 - Set Request Example

Example (Hex)
AA00 81040155D111 8410 D111 850101 870401010101 8903 C70100

Table 8.2-36 - Set Response Example

Example (Hex)
AA 00 81 04 82 55 D1 11 82 04 00 00 00 00 84 82 00 10 D1 11 85 01 01 87 04 01 01 01 01 89 03 C7 01 00

8.2.1.8 Property 1.1.1.1.1.8 Apple VAS Merchant ID and URL Slot 2

Table 8.2-37 - Property 1.1.1.1.1.8 Apple VAS Merchant ID and URL Slot 2

Property Description	
Property OID	1.1.1.1.1.8 / 0x010101010108

Property Description	
Name	Apple VAS Merchant ID and URL Slot 2
Description	The device uses this property for Apple VAS processing to configure the device's supported Merchant ID and URL.
Securing Key	None
Min. Len (b)	1
Max. Len (b)	102
Data Type	Binary
Valid Values	00 or see **Table 8.2-32 - VAS Merchant ID and URL Type**
Default	0x00

Table 8.2-38 - Get Request Example

Example (Hex)
AA00 8104 0155D101 840F D101 8501 01 8704 01010101 8902 C800

Table 8.2-39 - Get Response Example

Example (Hex)
AA 00 81 04 82 55 D1 01 82 04 00 00 00 00 84 82 00 49 D1 01 85 01 01 87 04 01 01 01 01 89 3C C8 3A 9F 25 20 3E 22 54 3A AF 0A C5 E4 AC FC 25 67 1A 6E BF 6E DE 5A C1 96 74 6C 55 F3 D4 37 08 19 FF F9 22 C3 9F 29 14 77 77 77 2E 65 78 61 6D 70 6C 65 2D 75 72 6C 2E 63 6F 6D

Table 8.2-40 - Set Request Example

Example (Hex)
AA00 81040155D111 8410 D111 850101 870401010101 8903 C80100

Table 8.2-41 - Set Response Example

Example (Hex)
AA 00 81 04 82 55 D1 11 82 04 00 00 00 00 84 82 00 10 D1 11 85 01 01 87 04 01 01 01 01 89 03 C8 01 00

8.2.1.9 Property 1.1.1.1.1.9 Apple VAS Merchant ID and URL Slot 3

Table 8.2-42 - Property 1.1.1.1.1.9 Apple VAS Merchant ID and URL Slot 3

Property Description	
Property OID	1.1.1.1.1.9 / 0x010101010109
Name	Apple VAS Merchant ID and URL Slot 3
Description	The device uses this property for Apple VAS processing to configure the device's supported Merchant ID and URL.
Securing Key	None

Property Description	
Min. Len (b)	1
Max. Len (b)	102
Data Type	Binary
Valid Values	00 or see **Table 8.2-32 - VAS Merchant ID and URL Type**
Default	0x00

Table 8.2-43 - Get Request Example

Example (Hex)
AA00 8104 0155D101 840F D101 8501 01 8704 01010101 8902 C900

Table 8.2-44 - Get Response Example

Example (Hex)
AA 00 81 04 82 55 D1 01 82 04 00 00 00 00 84 82 00 49 D1 01 85 01 01 87 04 01 01 01 01 89 3C C9 3A 9F 25 20 3E 22 54 3A AF 0A C5 E4 AC FC 25 67 1A 6E BF 6E DE 5A C1 96 74 6C 55 F3 D4 37 08 19 FF F9 22 C3 9F 29 14 77 77 77 2E 65 78 61 6D 70 6C 65 2D 75 72 6C 2E 63 6F 6D

Table 8.2-45 - Set Request Example

Example (Hex)
AA00 81040155D111 8410 D111 850101 870401010101 8903 C90100

Table 8.2-46 - Set Response Example

Example (Hex)
AA 00 81 04 82 55 D1 11 82 04 00 00 00 00 84 82 00 10 D1 11 85 01 01 87 04 01 01 01 01 89 03 C9 01 00

8.2.1.10 Property 1.1.1.1.1.A Apple VAS Merchant ID and URL Slot 4

Table 8.2-47 - Property 1.1.1.1.1.A Apple VAS Merchant ID and URL Slot 4

Property Description	
Property OID	1.1.1.1.1.A / 0x01010101010A
Name	Apple VAS Merchant ID and URL Slot 4
Description	The device uses this property for Apple VAS processing to configure the device's supported Merchant ID and URL.
Securing Key	None
Min. Len (b)	1
Max. Len (b)	102
Data Type	Binary
Valid Values	00 or see **Table 8.2-32 - VAS Merchant ID and URL Type**

Property Description	
Default	0x00

Table 8.2-48 - Get Request Example

Example (Hex)
AA00 8104 0155D101 840F D101 8501 01 8704 01010101 8902 CA00

Table 8.2-49 - Get Response Example

Example (Hex)
AA 00 81 04 82 55 D1 01 82 04 00 00 00 00 84 82 00 49 D1 01 85 01 01 87 04 01 01 01 01 89 3C CA 3A 9F 25 20 3E 22 54 3A AF 0A C5 E4 AC FC 25 67 1A 6E BF 6E DE 5A C1 96 74 6C 55 F3 D4 37 08 19 FF F9 22 C3 9F 29 14 77 77 77 2E 65 78 61 6D 70 6C 65 2D 75 72 6C 2E 63 6F 6D

Table 8.2-50 - Set Request Example

Example (Hex)
AA00 81040155D111 8410 D111 850101 870401010101 8903 CA0100

Table 8.2-51 - Set Response Example

Example (Hex)
AA 00 81 04 82 55 D1 11 82 04 00 00 00 00 84 82 00 10 D1 11 85 01 01 87 04 01 01 01 01 89 03 CA 01 00

8.2.1.11 Property 1.1.1.1.1.B Apple VAS Merchant ID and URL Slot 5

Table 8.2-52 - Property 1.1.1.1.1.B Apple VAS Merchant ID and URL Slot 5

Property Description	
Property OID	1.1.1.1.1.B / 0x01010101010B
Name	Apple VAS Merchant ID and URL Slot 5
Description	The device uses this property for Apple VAS processing to configure the device's supported Merchant ID and URL.
Securing Key	None
Min. Len (b)	1
Max. Len (b)	102
Data Type	Binary
Valid Values	00 or see **Table 8.2-32 - VAS Merchant ID and URL Type**
Default	0x00

Table 8.2-53 - Get Request Example

Example (Hex)
AA00 8104 0155D101 840F D101 8501 01 8704 01010101 8902 CB00

Table 8.2-54 - Get Response Example

Example (Hex)
AA 00 81 04 82 55 D1 01 82 04 00 00 00 00 84 82 00 49 D1 01 85 01 01 87 04 01 01 01 01 89 3C CB 3A 9F 25 20 3E 22 54 3A AF 0A C5 E4 AC FC 25 67 1A 6E BF 6E DE 5A C1 96 74 6C 55 F3 D4 37 08 19 FF F9 22 C3 9F 29 14 77 77 77 2E 65 78 61 6D 70 6C 65 2D 75 72 6C 2E 63 6F 6D

Table 8.2-55 - Set Request Example

Example (Hex)
AA00 81040155D111 8410 D111 850101 870401010101 8903 CB0100

Table 8.2-56 - Set Response Example

Example (Hex)
AA 00 81 04 82 55 D1 11 82 04 00 00 00 00 84 82 00 10 D1 11 85 01 01 87 04 01 01 01 01 89 03 CB 01 00

8.2.1.12 Property 1.1.1.1.1.C Apple VAS Merchant ID and URL Slot 6

Table 8.2-57 - Property 1.1.1.1.1.C Apple VAS Merchant ID and URL Slot 6

Property Description	
Property OID	1.1.1.1.1.C / 0x01010101010C
Name	Apple VAS Merchant ID and URL Slot 6
Description	The device uses this property for Apple VAS processing to configure the device's supported Merchant ID and URL.
Securing Key	None
Min. Len (b)	1
Max. Len (b)	102
Data Type	Binary
Valid Values	00 or see **Table 8.2-32 - VAS Merchant ID and URL Type**
Default	0x00

Table 8.2-58 - Get Request Example

Example (Hex)
AA00 8104 0155D101 840F D101 8501 01 8704 01010101 8902 CC00

Table 8.2-59 - Get Response Example

Example (Hex)
AA 00 81 04 82 55 D1 01 82 04 00 00 00 00 84 82 00 49 D1 01 85 01 01 87 04 01 01 01 01 89 3C CC 3A 9F 25 20 3E 22 54 3A AF 0A C5 E4 AC FC 25 67 1A 6E BF 6E DE 5A C1 96 74 6C 55 F3 D4 37 08 19 FF F9 22 C3 9F 29 14 77 77 77 2E 65 78 61 6D 70 6C 65 2D 75 72 6C 2E 63 6F 6D

Table 8.2-60 - Set Request Example

Example (Hex)
AA00 81040155D111 8410 D111 850101 870401010101 8903 CC0100

Table 8.2-61 - Set Response Example

Example (Hex)
AA 00 81 04 82 55 D1 11 82 04 00 00 00 00 84 82 00 10 D1 11 85 01 01 87 04 01 01 01 01 89 03 CC 01 00

8.2.1.13 Property 1.1.1.1.1.D Apple iAP2 AppBundleID

Table 8.2-62 - Property 1.1.1.1.1.D Apple iAP2 AppBundleID

Property Description	
Property OID	1.1.1.1.1.D / 0x01010101010D
Name	Apple iAP2 AppBundleID
Description	This setting allows the iOS device to recommend installation of a specific app or set of apps to use with the iOS device when connects to an accessory.
Securing Key	None
Min. Len (b)	A valid reverse-DNS similar to the Default below
Max. Len (b)	50
Data Type	ASCII
Valid Values	ASCII
Default	com.magtek.dynaflex2go

Table 8.2-63 - Get Request Example

Example (Hex)
AA 00 81 04 01 06 D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 01 89 0A E1 08 E1 06 E1 04 E1 02 CD 00

Table 8.2-64 - Get Response Example

Example (Hex)
AA 00 81 04 82 06 D1 01 82 04 00 00 00 00 84 82 00 4C D1 01 81 07 2B 06 01 04 01 F6 09 85 01 01 89 3C E1 3A E1 38 E1 36 E1 34 CD 32 63 6F 6D 2E 6D 61 67 74 65 6B 2E 64 79 6E 61 66 6C 65 78 32 67 6F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Table 8.2-65 - Set Request Example

Example (Hex)
AA 00 81 04 01 07 D1 11 84 4C D1 11 81 07 2B 06 01 04 01 F6 09 85 01 01 89 3C E1 3A E1 38 E1 36 E1 34 CD 32 63 6F 6D 2E 6D 61 67 74 65 6B 2E 64 79 6E 61 66 6C 65 78 32 67 6F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Table 8.2-66 - Set Response Example

Example (Hex)
AA 00 81 04 82 07 D1 11 82 04 00 00 00 00 84 82 00 4C D1 11 81 07 2B 06 01 04 01 F6 09 85 01 01 89 3C E1 3A E1 38 E1 36 E1 34 CD 32 63 6F 6D 2E 6D 61 67 74 65 6B 2E 64 79 6E 61 66 6C 65 78 32 67 6F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

8.2.1.14 Property 1.1.1.1.1.10 Google Smart Tap Collector ID Slot 1

Table 8.2-62 - Property 1.1.1.1.1.10 Google Smart Tap Collector ID Slot 1

Property Description	
Property OID	1.1.1.1.1.10 / 0x010101010110
Name	Google Smart Tap Collector ID Slot 1
Description	The device uses this property for Google Smart Tap processing to configure the device's supported Collector ID and Service Types.
Securing Key	None
Min. Len (b)	1
Max. Len (b)	25
Data Type	Binary
Valid Values	00 or see **Table 8.2-32 - VAS Merchant ID and URL Types**
Default	0x00

Table 8.2-63 – Google Smart Tap Collector ID and Service Types

Tag	Len	Value / Description	Typ	Req	Default
DF7C	0x08	GWST Collector ID	B	R	
DF7D	Var up to 0x0E	GWST Service Types 0x00 = All services 0x01 = All services except PPSE 0x02 = PPSE 0x03 = Loyalty 0x04 = Offer 0x05 = Gift card 0x06 = Private label card 0x07 = Event ticket 0x08 = Flight 0x09 = Transit 0x10 = Cloud-based wallet 0x11 = Mobil marketing platform 0x12 = Generic 0x13 = Generic private pass 0x40 = Wallet customer	B	R	

Table 8.2-64 - Get Request Example

Example (Hex)
AA00 8104 0155D101 840F D101 8501 01 8704 01010101 8902 D000

Table 8.2-68.2-67 - Get Response Example

Example (Hex)
AA 00 81 04 82 55 D1 01 82 04 00 00 00 00 84 82 00 1E D1 01 85 01 01 87 04 01 01 01 01 89 11 D0 0F DF 7C 08 32 30 31 38 30 36 30 38 DF 7D 01 00

Table 8.2-686 - Set Request Example

Example (Hex)
AA00 81048255D111 8427 D111 8107 2B06010401F609 8501 01 8704 01010101 8911 D00F DF7C083230313830363038DF7D0100

Table 8.2-69 - Set Response Example

Example (Hex)
AA 00 81 04 82 55 D1 11 82 04 00 00 00 00 84 82 00 2 7D 11 18 10 7 2B 06 01 04 01 F6 09 85 01 01 87 04 01 01 01 01 89 11 D0 0F DF 7C 08 32 30 31 38 30 36 30 38 DF 7D 01 00

8.2.1.15 Property 1.1.1.1.1.11 Google Smart Tap Collector ID Slot 2

Table 8.2-68 - Property 1.1.1.1.1.11 Google Smart Tap Collector ID Slot 2

Property Description	
Property OID	1.1.1.1.1.11 / 0x010101010111
Name	Google Smart Tap Collector ID Slot 2
Description	The device uses this property for Google Smart Tap processing to configure the device's supported Collector ID and Service Types.
Securing Key	None
Min. Len (b)	1
Max. Len (b)	25
Data Type	Binary
Valid Values	00 or see **Table 8.2-32 - VAS Merchant ID and URL Types**
Default	0x00

Table 8.2-69 - Get Request Example

Example (Hex)
AA00 8104 0155D101 840F D101 8501 01 8704 01010101 8902 D100

Table 8.2-70 - Get Response Example

Example (Hex)
AA 00 81 04 82 55 D1 01 82 04 00 00 00 00 84 82 00 1E D1 01 85 01 01 87 04 01 01 01 01 89 11 D1 0F DF 7C 08 32 30 31 38 30 36 30 38 DF 7D 01 00

Table 8.2-71 - Set Request Example

Example (Hex)
AA00 81048255D111 8427 D111 8107 2B06010401F609 8501 01 8704 01010101 8911 D10F DF7C083230313830363038DF7D0100

Table 8.2-702 - Set Response Example

Example (Hex)
AA 00 81 04 82 55 D1 11 82 04 00 00 00 00 84 82 00 2 7D 11 18 10 7 2B 06 01 04 01 F6 09 85 01 01 87 04 01 01 01 01 89 11 D1 0F DF 7C 08 32 30 31 38 30 36 30 38 DF 7D 01 00

8.2.1.16 Property 1.1.1.1.1.12 Google Smart Tap Collector ID Slot 3

Table 8.2-73 - Property 1.1.1.1.1.12 Google Smart Tap Collector ID Slot 3

Property Description	
Property OID	1.1.1.1.1.12 / 0x010101010112
Name	Google Smart Tap Collector ID Slot 3
Description	The device uses this property for Google Smart Tap processing to configure the device's supported Collector ID and Service Types.
Securing Key	None
Min. Len (b)	1
Max. Len (b)	25
Data Type	Binary
Valid Values	00 or see **Table 8.2-32 - VAS Merchant ID and URL Types**
Default	0x00

Table 8.2-74 - Get Request Example

Example (Hex)
AA00 8104 0155D101 840F D101 8501 01 8704 01010101 8902 D200

Table 8.2-75 - Get Response Example

Example (Hex)
AA 00 81 04 82 55 D1 01 82 04 00 00 00 00 84 82 00 1E D1 01 85 01 01 87 04 01 01 01 01 89 11 D2 0F DF 7C 08 32 30 31 38 30 36 30 38 DF 7D 01 00

Table 8.2-76 - Set Request Example

Example (Hex)
AA00 81048255D111 8427 D111 8107 2B06010401F609 8501 01 8704 01010101 8911 D20F DF7C083230313830363038DF7D0100

Table 8.2-717 - Set Response Example

Example (Hex)
AA 00 81 04 82 55 D1 11 82 04 00 00 00 00 84 82 00 2 7D 11 18 10 7 2B 06 01 04 01 F6 09 85 01 01 87 04 01 01 01 01 89 11 D2 0F DF 7C 08 32 30 31 38 30 36 30 38 DF 7D 01 00

8.2.1.17 Property 1.1.1.1.1.13 Google Smart Tap Collector ID Slot 4

Table 8.2-78 - Property 1.1.1.1.1.13 Google Smart Tap Collector ID Slot 4

Property Description	
Property OID	1.1.1.1.1.13 / 0x010101010113
Name	Google Smart Tap Collector ID Slot 4
Description	The device uses this property for Google Smart Tap processing to configure the device's supported Collector ID and Service Types.
Securing Key	None
Min. Len (b)	1
Max. Len (b)	25
Data Type	Binary
Valid Values	00 or see **Table 8.2-32 - VAS Merchant ID and URL Types**
Default	0x00

Table 8.2-79 - Get Request Example

Example (Hex)
AA00 8104 0155D101 840F D101 8501 01 8704 01010101 8902 D300

Table 8.2-80 - Get Response Example

Example (Hex)
AA 00 81 04 82 55 D1 01 82 04 00 00 00 00 84 82 00 1E D1 01 85 01 01 87 04 01 01 01 01 89 11 D3 0F DF 7C 08 32 30 31 38 30 36 30 38 DF 7D 01 00

Table 8.2-81 - Set Request Example

Example (Hex)
AA00 81048255D111 8427 D111 8107 2B06010401F609 8501 01 8704 01010101 8911 D30F DF7C083230313830363038DF7D0100

Table 8.2-82 - Set Response Example

Example (Hex)
AA 00 81 04 82 55 D1 11 82 04 00 00 00 00 84 82 00 2 7D 11 18 10 7 2B 06 01 04 01 F6 09 85 01 01 87 04 01 01 01 01 89 11 D3 0F DF 7C 08 32 30 31 38 30 36 30 38 DF 7D 01 00

8.2.1.18 Property 1.1.1.1.1.14 Google Smart Tap Collector ID Slot 5

Table 8.2-83 - Property 1.1.1.1.1.14 Google Smart Tap Collector ID Slot 5

Property Description	
Property OID	1.1.1.1.1.14 / 0x010101010114
Name	Google Smart Tap Collector ID Slot 5
Description	The device uses this property for Google Smart Tap processing to configure the device's supported Collector ID and Service Types.
Securing Key	None
Min. Len (b)	1
Max. Len (b)	25
Data Type	Binary
Valid Values	00 or see **Table 8.2-32 - VAS Merchant ID and URL Types**
Default	0x00

Table 8.2-84 - Get Request Example

Example (Hex)
AA00 8104 0155D101 840F D101 8501 01 8704 01010101 8902 D400

Table 8.2-85 - Get Response Example

Example (Hex)
AA 00 81 04 82 55 D1 01 82 04 00 00 00 00 84 82 00 1E D1 01 85 01 01 87 04 01 01 01 01 89 11 D4 0F DF 7C 08 32 30 31 38 30 36 30 38 DF 7D 01 00

Table 8.2-86 - Set Request Example

Example (Hex)
AA00 81048255D111 8427 D111 8107 2B06010401F609 8501 01 8704 01010101 8911 D40F DF7C083230313830363038DF7D0100

Table 8.2-87 - Set Response Example

Example (Hex)
AA 00 81 04 82 55 D1 11 82 04 00 00 00 00 84 82 00 2 7D 11 18 10 7 2B 06 01 04 01 F6 09 85 01 01 87 04 01 01 01 01 89 11 D4 0F DF 7C 08 32 30 31 38 30 36 30 38 DF 7D 01 00

8.2.1.19 Property 1.1.1.1.1.15 Google Smart Tap Collector ID Slot 6

Table 8.2-88 - Property 1.1.1.1.1.15 Google Smart Tap Collector ID Slot 6

Property Description	
Property OID	1.1.1.1.1.15 / 0x010101010115
Name	Google Smart Tap Collector ID Slot 6
Description	The device uses this property for Google Smart Tap processing to configure the device's supported Collector ID and Service Types.
Securing Key	None
Min. Len (b)	1
Max. Len (b)	25
Data Type	Binary
Valid Values	00 or see **Table 8.2-32 - VAS Merchant ID and URL Types**
Default	0x00

Table 8.2-89 - Get Request Example

Example (Hex)
AA00 8104 0155D101 840F D101 8501 01 8704 01010101 8902 D500

Table 8.2-90 - Get Response Example

Example (Hex)
AA 00 81 04 82 55 D1 01 82 04 00 00 00 00 84 82 00 1E D1 01 85 01 01 87 04 01 01 01 01 89 11 D5 0F DF 7C 08 32 30 31 38 30 36 30 38 DF 7D 01 00

Table 8.2-91 - Set Request Example

Example (Hex)
AA00 81048255D111 8427 D111 8107 2B06010401F609 8501 01 8704 01010101 8911 D50F DF7C083230313830363038DF7D0100

Table 8.2-92 - Set Response Example

Example (Hex)
AA 00 81 04 82 55 D1 11 82 04 00 00 00 00 84 82 00 2 7D 11 18 10 7 2B 06 01 04 01 F6 09 85 01 01 87 04 01 01 01 01 89 11 D5 0F DF 7C 08 32 30 31 38 30 36 30 38 DF 7D 01 00

8.2.1.20 Property 1.1.1.1.1.1A Google Smart Tap POS Capabilities

Table 8.2-83 - Property 1.1.1.1.1.1A Google Smart Tap POS CapabilitiesPOS Capabilities

Property Description	
Property OID	1.1.1.1.1.1A / 0x01010101011A
Name	Google Smart Tap POS Capabilities

Property Description	
Description	The device uses this property for Google Smart Tap processing to configure the device's supported POS Capabilities.
Securing Key	None
Min. Len (b)	4
Max. Len (b)	4
Data Type	Binary
Valid Values	Byte value where each bit of the byte indicates a particular POS capability. Bit 0 is the least significant bit. The Tap byte is generated by the device. Byte 1 – System: Bit 0: Standalone Bit 1: Semi-integrated Bit 2: Unattended Bit 3: Online Bit 4: Offline Bit 5: MMP Bit 6: zlib support Bit 7: RFU Byte 2 – UI: Bit 0: Printer Bit 1: Printer graphics Bit 2: Display Bit 3: Images Bit 4: Audio Bit 5: Animation Bit 6: Video Bit 7: RFU Byte 3 – Checkout: Bit 0: Support payment Bit 1: Support digital receipt Bit 2: Support service issuance Bit 3: Support OTA POS data Bit 4: RFU Bit 5: RFU Bit 6: RFU Bit 7: RFU Byte 4 – CVM Bit 0: Online PIN Bit 1: CD PIN Bit 2: Signature

Property Description	
	Bit 3: No CVM Bit 4: Device-generated code Bit 5: SP-generated code Bit 6: ID capture Bit 7: Biometric
Default	DynaProx, DynaFlex II Go and DynaFlex II: 0x08, 0x10, 0x03, 0x00 All Other devices: 0x08, 0x14, 0x03, 0x05

Table 8.2-84 - Get Request Example

Example (Hex)
AA00 8104 0155D101 840F D101 8501 01 8704 01010101 8902 D400

Table 8.2-85 - Get Response Example

Example (Hex)
AA 00 81 04 82 55 D1 01 82 04 00 00 00 00 84 82 00 1E D1 01 85 01 01 87 04 01 01 01 01 89 11 D4 0F DF 7C 08 32 30 31 38 30 36 30 38 DF 7D 01 00

Table 8.2-86 - Set Request Example

Example (Hex)
AA00 81048255D111 8427 D111 8107 2B06010401F609 8501 01 8704 01010101 8911 D40F DF7C083230313830363038DF7D0100

Table 8.2-87 - Set Response Example

Example (Hex)
AA 00 81 04 82 55 D1 11 82 04 00 00 00 00 84 82 00 2 7D 11 18 10 7 2B 06 01 04 01 F6 09 85 01 01 87 04 01 01 01 01 89 11 D4 0F DF 7C 08 32 30 31 38 30 36 30 38 DF 7D 01 00

8.2.1.21 Property 1.1.1.1.2.6 Tip Mode Enable Submit on Amount Button Press

Table 8.2-72 - Property 1.1.1.1.2.6 Tip Mode Enable Submit on Amount Button Press

Property Description	
Property OID	1.1.1.1.2.6 / 0x010101010206
Name	Tip Mode Enable Submit on Amount Button Press
Description	When enabled, pressing an amount, percent, or 'no tip' button on the tip entry page will proceed immediately without requiring a subsequent tap of the 'Submit' button.
Securing Key	None
Min. Len (b)	1
Max. Len (b)	1

Property Description	
Data Type	Binary
Valid Values	0x00 – Disabled: Require submit button tap 0x01 – Enabled: Don't require submit button tap
Default	0x00 - Disabled

Table 8.2-73 - Get Request Example

Example (Hex)
AA0081040155D101840FD10185010187040101010289 02C600

Table 8.2-74 - Get Response Example

Example (Hex)
AA0081048255D10182040000000084820010D10185010187040101010289 03C60101

Table 8.2-75 - Set Request Example

Example (Hex)
AA0081040155D1118410D11185010187040101010289 03C60101

Table 8.2-76 - Set Response Example

Example (Hex)
AA0081048255D11182040000000084820010D11185010187040101010289 03C60101

8.2.1.22 Property 1.1.1.1.4.1 EMV Configuration Filename

Table 8.2-77 - Property 1.1.1.1.4.1 EMV Configuration Filename

Property Description	
Property OID	1.1.1.1.4.1 / 0x010101010401
Name	EMV Configuration Filename
Description	This string contains the part number and the revision of the EMV Configuration File.
Securing Key	None
Min. Len (b)	14
Max. Len (b)	14
Data Type	ASCII
Valid Values	CFG000xxxx-xxx
Default	None

Table 8.2-78 - Get Request Example

Example (Hex)
AA0081040106D101841AD10181072B06010401F609850101890AE108E106E104E402C100

Table 8.2-79 - Get Response Example

Example (Hex)
AA0081048206D10182040000000084820028D10181072B06010401F609850101 8918E116E114E112E410C10E434647303030363831322D323030

Table 8.2-80 - Set Request Example

Example (Hex)
AA0081040106D1118428D11181072B06010401F6098501018918E116E114E112E410C10E 434647303030363831322D323030

Table 8.2-81 - Set Response Example

Example (Hex)
AA0081048206D11182040000000084820028D11181072B06010401F609850101 8918E116E114E112E410C10E434647303030363831322D323030

8.2.1.23 Property 1.1.1.1.4.2 CA Public Key Configuration Filename

Table 8.2-82 - Property 1.1.1.1.4.2 CA Public Key Configuration Filename

Property Description	
Property OID	1.1.1.1.4.2 / 0x010101010402
Name	CA Public Key Configuration Filename
Description	This string contains the part number and the revision of the CA Public Key Configuration File.
Securing Key	None
Min. Len (b)	14
Max. Len (b)	14
Data Type	ASCII
Valid Values	CFG000xxxx-xxx
Default	None

Table 8.2-83 - Get Request Example

Example (Hex)
AA0081040106D101841AD10181072B06010401F609850101890AE108E106E104E402C200

Table 8.2-84 - Get Response Example

Example (Hex)
AA0081048206D10182040000000084820028D10181072B06010401F609850101 8918E116E114E112E410C20E43464730303036383 1322D323030

Table 8.2-85 - Set Request Example

Example (Hex)
AA0081040106D1118428D11181072B06010401F6098501018918E116E114E112E410C20E 43464730303036383 1322D323030

Table 8.2-86 - Set Response Example

Example (Hex)
AA0081048206D11182040000000084820028D11181072B06010401F609850101 8918E116E114E112E410C20E43464730303036383 1322D323030

8.2.2 Property Subgroup 1.1.2.nnn SRED Settings

8.2.2.1 Property 1.1.2.2.1.2 ISO/ABA Masking Character

Table 8.2-87 - Property 1.1.2.2.1.2 ISO/ABA Masking Character

Property Description	
Property OID	1.1.2.2.1.2 / 0x010102020102
Name	ISO/ABA Masking Character
Description	The device uses this property to determine what character it should use when sending masked ISO/ABA track data. If the **Property 1.1.2.2.1.5 PAN MOD 10 Check Digit Correction** is enabled, the masked portions of the track will always be masked with '0'. For more information about masking, see the information at the bottom of this section.
Securing Key	None
Min. Len (b)	0x00
Max. Len (b)	0x01
Data Type	Character
Valid Values	Any ASCII character
Default	'0'

Table 8.2-88 - Get Request Example

Example (Hex)
AA 00 81 04 01 17 D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 01 89 0A E1 08 E2 06 E2 04 E1 02 C2 00

Table 8.2-89 - Get Response Example

Example (Hex)
AA 00 81 04 82 17 D1 01 82 04 00 00 00 00 84 82 00 1B D1 01 81 07 2B 06 01 04 01 F6 09 85 01 01 89 0B E1 09 E2 07 E2 05 E1 03 C2 01 2A

Table 8.2-90 - Set Request Example

Example (Hex)
AA 00 81 04 01 0F D1 11 84 1B D1 11 81 07 2B 06 01 04 01 F6 09 85 01 01 89 0B E1 09 E2 07 E2 05 E1 03 C2 01 30

Table 8.2-91 - Set Response Example

Example (Hex)
AA 00 81 04 82 0F D1 11 82 04 00 00 00 00 84 82 00 1B D1 11 81 07 2B 06 01 04 01 F6 09 85 01 01 89 0B E1 09 E2 07 E2 05 E1 03 C2 01 30

When the device sends masked track data, it replaces some characters in the track data with a specified mask character. The specific characters the device masks depends on the card encode type. The device only selectively masks data that came from ISO/ABA cards (Financial Cards with ***ISO/IEC 7813*** Format code B); it sends data from all other card types entirely unmasked.

The device masks all fields from ISO/ABA cards except a number of configurable leading and trailing characters of the PAN. See **Property 1.1.2.2.1.3 PAN Number of Leading Unmasked Characters** and **Property 1.1.2.2.1.4 PAN Number of Trailing Unmasked Characters**.

For chip card transactions, masked track 2 data is contained in TLV data object DFDF4D. For magnetic stripe transactions, masked track 1, 2 and 3 data is contained in TLV data objects DFDF31, DFDF33 and DFDF35, respectively.

Table 8.2-92 below provides an example of data from tracks 1, 2, and 3 of an ISO/ABA card after it has been decrypted or if the device has sent it in the clear. **Table 8.2-93** shows the same data as it might appear with a specific set of masking rules applied.

Table 8.2-92 - Sample ISO/ABA Track Data, Clear Text / Decrypted

Sample ISO/ABA Track Data, Clear Text / Decrypted	
Track 1	%B6011000995500000^ TEST CARD ^15121015432112345678?
Track 2	;6011000995500000=15121015432112345678?
Track 3	;6011000995500000=151210154321123456783333333333333333333333?

Table 8.2-93 - Sample ISO/ABA Track Data, Masked

Sample ISO/ABA Track Data, Masked	
Track 1	%B6011000020000000^0000000000^00000000000000000000?
Track 2	;6011000020000000=00000000000000000000?
Track 3	;6011000020000000=000000000000000000000000000000000000000000?

8.2.2.2 Property 1.1.2.2.1.3 PAN Number of Leading Unmasked Characters

Table 8.2-94 - Property 1.1.2.2.1.3 PAN Number of Leading Unmasked Characters

Property Description	
Property OID	1.1.2.2.1.3 / 0x010102020103
Name	PAN Number of Leading Unmasked Characters
Description	The device uses this property to determine how many of the leading characters of the PAN the device sends unmasked in **Masked Track x Data** in ISO/ABA account information For details about ISO/ABA track masking, see **Property 1.1.2.2.1.2 ISO/ABA** Masking Character.
Securing Key	None
Min. Len (b)	0x01
Max. Len (b)	0x01
Data Type	Binary
Valid Values	Min 0x00 Max 0x08, if PAN length is less than 16, the number of unmasked characters will be limited to 6.
Default	0x04

Table 8.2-95 - Get Request Example

Example (Hex)
AA 00 81 04 01 18 D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 01 89 0A E1 08 E2 06 E2 04 E1 02 C3 00

Table 8.2-96 - Get Response Example

Example (Hex)
AA 00 81 04 82 18 D1 01 82 04 00 00 00 00 84 82 00 1B D1 01 81 07 2B 06 01 04 01 F6 09 85 01 01 89 0B E1 09 E2 07 E2 05 E1 03 C3 01 04

Table 8.2-97 - Set Request Example

Example (Hex)
AA 00 81 04 01 10 D1 11 84 1B D1 11 81 07 2B 06 01 04 01 F6 09 85 01 01 89 0B E1 09 E2 07 E2 05 E1 03 C3 01 04

Table 8.2-98 - Set Response Example

Example (Hex)
AA 00 81 04 82 10 D1 11 82 04 00 00 00 00 84 82 00 1B D1 11 81 07 2B 06 01 04 01 F6 09 85 01 01 89 0B E1 09 E2 07 E2 05 E1 03 C3 01 04

8.2.2.3 Property 1.1.2.2.1.4 PAN Number of Trailing Unmasked Characters

Table 8.2-99 - Property 1.1.2.2.1.4 PAN Number of Trailing Unmasked Characters

Property Description	
Property OID	1.1.2.2.1.4 / 0x010102020104
Name	PAN Number of Trailing Unmasked Characters
Description	The device uses this property to determine how many of the trailing characters of the PAN the device sends unmasked in **Masked Track x Data** in ISO/ABA account information For details about ISO/ABA track masking, see **Property 1.1.2.2.1.2 ISO/ABA Masking Character**.
Securing Key	None
Min. Len (b)	0x01
Max. Len (b)	0x01
Data Type	Binary
Valid Values	Min 0x00 Max 0x04
Default	0x04

Table 8.2-100 - Get Request Example

Example (Hex)
AA 00 81 04 01 19 D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 01 89 0A E1 08 E2 06 E2 04 E1 02 C4 00

Table 8.2-101 - Get Response Example

Example (Hex)
AA 00 81 04 82 19 D1 01 82 04 00 00 00 00 84 82 00 1B D1 01 81 07 2B 06 01 04 01 F6 09 85 01 01 89 0B E1 09 E2 07 E2 05 E1 03 C4 01 04

Table 8.2-102 - Set Request Example

Example (Hex)
AA 00 81 04 01 11 D1 11 84 1B D1 11 81 07 2B 06 01 04 01 F6 09 85 01 01 89 0B E1 09 E2 07 E2 05 E1 03 C4 01 04

Table 8.2-103 - Set Response Example

Example (Hex)
AA 00 81 04 82 11 D1 11 82 04 00 00 00 00 84 82 00 1B D1 11 81 07 2B 06 01 04 01 F6 09 85 01 01 89 0B E1 09 E2 07 E2 05 E1 03 C4 01 04

8.2.2.4 Property 1.1.2.2.1.5 PAN MOD 10 Check Digit Correction

Table 8.2-104 - Property 1.1.2.2.1.5 PAN MOD 10 Check Digit Correction

Property Description	
Property OID	1.1.2.2.1.5 / 0x010102020105
Name	PAN MOD 10 Check Digit Correction
Description	The device uses this property to determine whether it should modify one of the masked PAN digits such that the masked PAN passes the Luhn MOD-10 algorithm check. If this property is enabled, the device uses masking character '0' to mask the PAN, regardless of the setting in **Property 1.1.2.2.1.2 ISO/ABA Masking Character**, and masks the remainder of the track data with the configured masking character.
Securing Key	None
Min. Len (b)	0x01
Max. Len (b)	0x01
Data Type	Binary
Valid Values	0x00 = Disabled 0x01 = Enabled
Default	0x01

Table 8.2-105 - Get Request Example

Example (Hex)
AA 00 81 04 01 1A D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 01 89 0A E1 08 E2 06 E2 04 E1 02 C5 00

Table 8.2-106 - Get Response Example

Example (Hex)
AA 00 81 04 82 1A D1 01 82 04 00 00 00 00 84 82 00 1B D1 01 81 07 2B 06 01 04 01 F6 09 85 01 01 89 0B E1 09 E2 07 E2 05 E1 03 C5 01 01

Table 8.2-107 - Set Request Example

Example (Hex)
AA 00 81 04 01 12 D1 11 84 1B D1 11 81 07 2B 06 01 04 01 F6 09 85 01 01 89 0B E1 09 E2 07 E2 05 E1 03 C5 01 01

Table 8.2-108 - Set Response Example

Example (Hex)
AA 00 81 04 82 12 D1 11 82 04 00 00 00 00 84 82 00 1B D1 11 81 07 2B 06 01 04 01 F6 09 85 01 01 89 0B E1 09 E2 07 E2 05 E1 03 C5 01 01

8.2.2.5 Property 1.1.2.4.1.2 Key Mapping of Account Data

Table 8.2-109 - Property 1.1.2.4.1.2 Key Mapping of Account DataKey Mapping of Account Data

Property Description	
Property OID	1.1.2.4.1.2 / 0x010102040102
Name	Key Mapping of Account Data
Description	The device uses this property to determine which DUKPT Key Set and Variant/Usage shall be used for Account Data.
Securing Key	None
Min. Len (b)	0x03
Max. Len (b)	0x03
Data Type	Binary
Valid Values	See DUKPT Key Mapping for detail information.
Default	0x20, 0x07, 0x04

Table 8.2-110 - Get Request Example

Example (Hex)
AA00 81040159D101 841AD101 81072B06010401F609 850101 890AE108E206E404E102C200

Table 8.2-111 - Get Response Example

Example (Hex)
AA00 81048259D101 820400000000 8482001DD101 81072B06010401F609 850101 890DE10BE209E407E105C2 03200704

Table 8.2-112 - Set Request Example

Example (Hex)
AA00 8104015AD111 841DD111 81072B06010401F609 850101 890DE10BE209E407E105C2 03200704

Table 8.2-113 - Set Response Example

Example (Hex)
AA00 8104825AD111 820400000000 8482001DD111 81072B06010401F609 850101 890DE10BE209E407E105C2 03200704

8.2.2.6 Property 1.1.2.4.1.3 Key Mapping of MAC

Table 8.2-114 - Property 1.1.2.4.1.3 Key Mapping of MACKey Mapping of MAC

Property Description	
Property OID	1.1.2.4.1.3 / 0x010102040103
Name	Key Mapping of MAC

Property Description	
Description	The device uses this property to determine which DUKPT Key Set and Variant/Usage shall be used for MAC.
Securing Key	None
Min. Len (b)	0x03
Max. Len (b)	0x03
Data Type	Binary
Valid Values	See DUKPT Key Mapping for detail information.
Default	0x20, 0x07, 0x02

Table 8.2-115 - Get Request Example

Example (Hex)
AA00 81040159D101 841AD101 81072B06010401F609 850101 890AE108E206E404E102C300

Table 8.2-116 - Get Response Example

Example (Hex)
AA00 81048259D101 820400000000 8482001DD101 81072B06010401F609 850101 890DE10BE209E407E105C3 03200702

Table 8.2-117 - Set Request Example

Example (Hex)
AA00 8104015AD111 841DD111 81072B06010401F609 850101 890DE10BE209E407E105C3 03200702

Table 8.2-118 - Set Response Example

Example (Hex)
AA00 8104825AD111 820400000000 8482001DD111 81072B06010401F609 850101 890DE10BE209E407E105C3 03200702

8.2.2.7 Property 1.1.2.4.1.5 Key Mapping of MagTek Token

This OID is reserved for Mey Mapping of MagTek Token.

8.2.2.8 Property 1.1.2.4.1.6 Key Mapping of User Data 1

This OID is reserved for Mey Mapping of User Data 1.

8.3 Property Group 1.2.nnnn Device Settings

8.3.1 Property Subgroup 1.2.1.nnn Transaction Settings

8.3.1.1 Property 1.2.1.1.1.2 Application Selection Behavior (Contactless Only)

Table 8.3-1 - Property 1.2.1.1.1.2 Application Selection Behavior (Contactless Only)Application Selection Behavior

Property Description	
Property OID	1.2.1.1.1.2 / 0x010201010102
Name	Application Selection Behavior
Description	The device uses this property to set the device's application selection behavior for payment brand selection.
Securing Key	None
Min. Len (b)	1
Max. Len (b)	1
Data Type	Binary
Valid Values	0x00 –Card Preference. The device automatically chooses the application that is mutually supported by the card and the terminal, based on the priority order specified by the card. This is the default and is standard EMV transaction flow behavior. 0x01 – Reserved 0x02 – Reserved 0x03 – Enhanced Prompt Cardholder. The device sends the host **Notification 0x1803 – 02 02 00 00** (Display Cardholder, Data Attached, Reserved) to request that the cardholder select from a list of available applications. Each application may include additional tags such as - priority indicator, Issuer Country Code and Issuer Identification Number. After the cardholder selects an application, the host passes the selection to the device using **Command 0x1802 - Report Cardholder Selection** to device to return the cardholder's selection to the device. Note that if the device is configured to use this option, the cardholder must hold the card or contactless payment device in contact with the device for until cardholder application selection is complete, otherwise the device will report the transaction failed. For this reason, MagTek does not recommend using this setting.
Default	0x00 – Card Preference

Table 8.3-2 - Get Request Example

Example (Hex)
AA00 8104 0155D101 840F D101 8501 01 8704 02010101 8902 C200

Table 8.3-3 - Get Response Example

Example (Hex)
AA 00 81 04 82 55 D1 01 82 04 00 00 00 00 84 82 00 10 D1 01 85 01 01 87 04 02 01 01 01 89 03 C2 01 00

Table 8.3-4 - Set Request Example

Example (Hex)
AA00 81040155D111 8410 D111 850101 870402010101 8903 C20103

Table 8.3-5 - Set Response Example

Example (Hex)
AA 00 81 04 82 55 D1 11 82 04 00 00 00 00 84 82 00 10 D1 11 85 01 01 87 04 02 01 01 01 89 03 C2 01 03

8.3.1.2 Property 1.2.1.1.2.1 Signature Capture Control

Table 8.3-6 - Property 1.2.1.1.2.1 Signature Capture Control

Property Description	
Property OID	1.2.1.1.2.1 / 0x010201010201
Name	Signature Capture Control
Description	The host can use this property to change whether the device automatically prompts the cardholder for a signature when running **Command 0x1001 - Start Transaction**, or leaves the triggering of signature capture prompts to the host.
Securing Key	None
Min. Len (b)	1
Max. Len (b)	1
Data Type	Binary
Valid Values	0x00 = Device-driven Signature Capture 0x01 = Host-driven Signature Capture
Default	0x00

Table 8.3-7 - Get Request Example

Example (Hex)
AA 00 81 04 01 04 D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 01 89 0A E2 08 E1 06 E1 04 E2 02 C1 00

Table 8.3-8 - Get Response Example

Example (Hex)
AA 00 81 04 82 04 D1 01 82 04 00 00 00 00 84 82 00 1B D1 01 81 07 2B 06 01 04 01 F6 09 85 01 01 89 0B E2 09 E1 07 E1 05 E2 03 C1 01 01

Table 8.3-9 - Set Request Example

Example (Hex)
AA 00 81 04 01 02 D1 11 84 1B D1 11 81 07 2B 06 01 04 01 F6 09 85 01 01 89 0B E2 09 E1 07 E1 05 E2 03 C1 01 01

Table 8.3-10 - Set Response Example

Example (Hex)
AA 00 81 04 82 02 D1 11 82 04 00 00 00 00 84 82 00 1B D1 11 81 07 2B 06 01 04 01 F6 09 85 01 01 89 0B E2 09 E1 07 E1 05 E2 03 C1 01 01

8.3.1.3 Property 1.2.1.1.3.1 Contactless Low Power Card Detect (Contactless Only)

Table 8.3-11 - Property 1.2.1.1.3.1 Contactless Low Power Card Detect (Contactless Only)

Property Description	
Property OID	1.2.1.1.3.1 / 0x0102010100301
Name	Contactless Low Power Card Detect
Description	The device uses this property to enable/disable contactless low power card detect (Proximity Detection Mode). When disabled, device will read a contactless card as soon as the card is detected. If enabled, device will track and delay reading a contactless card, allowing user to use Contact or MSR card slots.
Securing Key	None
Min. Len (b)	1
Max. Len (b)	1
Data Type	Binary
Valid Values	0x00 = Disable 0x01 = Enable
Default	0x00 - Disabled

Table 8.3-12 - Get Request Example

Example (Hex)
AA 00 81 04 01 55 D1 01 84 0F D1 01 85 01 01 87 04 02 01 01 03 89 02 C1 00

Table 8.3-13 - Get Response Example

Example (Hex)
AA 00 81 04 82 55 D1 01 82 04 00 00 00 00 84 82 00 10 D1 01 85 01 01 87 04 02 01 01 03 89 03 C1 01 00

Table 8.3-14 - Set Request Example

Example (Hex)
AA 00 81 04 01 55 D1 11 84 10 D1 11 85 01 01 87 04 02 01 01 03 89 03 C1 01 00

Table 8.3-15 - Set Response Example

Example (Hex)
AA 00 81 04 82 55 D1 11 82 04 00 00 00 00 84 82 00 10 D1 11 85 01 01 87 04 02 01 01 03 89 03 C1 01 00

8.3.1.4 Property 1.2.1.1.4.1 MIFARE Ultralight C 2keys3DES

Table 8.3-16 - Property 1.2.1.1.4.1 MIFARE Ultralight C 2keys3DES

Property Description	
Property OID	1.2.1.1.4.1 / 0x0102010100401
Name	MIFARE Ultralight C 2keys3DES
Description	The device uses this property to set 16-byte of the 2keys 3DES for MIFARE Ultralight C authentication. On an example of Key1 = 0001020304050607h and Key2 = 08090A0B0C0D0E0Fh, the setting should be 000102030405060708090A0B0C0D0E0F For security, the Get request for this property will always return 3-MSB of KCV. This key will be encrypted and stored in KPM.
Securing Key	None
Min. Len (b)	16
Max. Len (b)	16
Data Type	Binary
Valid Values	
Default	49454D4B 41455242 214E4143 554F5946

Table 8.3-17 - Get Request Example

Example (Hex)
AA 00 81 04 01 06 D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 01 89 0A E2 08 E1 06 E1 04 E4 02 C1 00

Table 8.3-18 - Get Response Example

Example (Hex)
AA 00 81 04 82 06 D1 01 82 04 00 00 00 00 84 82 00 1D D1 01 81 07 2B 06 01 04 01 F6 09 85 01 01 89 0D E2 0B E1 09 E1 07 E4 05 C1 03 29 07 96

Table 8.3-19 - Set Request Example

Example (Hex)
AA 00 81 04 01 55 D1 11 84 1F D1 11 85 01 01 87 04 02 01 01 04 89 12 C1 10 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

Table 8.3-20 - Set Response Example

Example (Hex)
AA 00 81 04 82 55 D1 11 82 04 00 00 00 00 84 82 00 1F D1 11 85 01 01 87 04 02 01 01 04 89 12 C1 10 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

8.3.1.5 Property 1.2.1.1.4.2 MIFARE Ultralight AES DataProtKey.

Table 8.3-21 - Property 1.2.1.1.4.2 MIFARE Ultralight AES DataProtKey.

Property Description	
Property OID	1.2.1.1.4.2 / 0x0102010100402
Name	MIFARE Ultralight AES DataProtKey.
Description	The device uses this property to set 16-byte of the AES_DataProtKey for MIFARE Ultralight AES authentication. On an example of ASE Key = 000102030405060708090A0B0C0D0E0Fh, the setting should be 000102030405060708090A0B0C0D0E0F For security, the Get request for this property will always return 4-MSB of KCV. This key will be encrypted and stored in KPM.
Securing Key	None
Min. Len (b)	16
Max. Len (b)	16
Data Type	Binary
Valid Values	
Default	00000000 00000000 00000000 00000000

Table 8.3-22 - Get Request Example

Example (Hex)
AA 00 81 04 01 06 D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 01 89 0A E2 08 E1 06 E1 04 E4 02 C2 00

Table 8.3-23 - Get Response Example

Example (Hex)
AA00810482D3D10182040000000008482001ED10181072B06010401F609850101890EE20CE10AE108E406C204763CBCDE

Table 8.3-24 - Set Request Example

Example (Hex)
AA 00 81 04 01 55 D1 11 84 1F D1 11 85 01 01 87 04 02 01 01 04 89 12 C2 10 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

Table 8.3-25 - Set Response Example

Example (Hex)
AA00810482E5D11182040000000008482002AD11181072B06010401F609850101891AE218E116E114E412C2100001020304050607080 90A0B0C0D0E0F

8.3.1.6 Property 1.2.1.1.4.3 MIFARE Ultralight AES UIDRetrKey.

Table 8.3-26 - Property 1.2.1.1.4.3 MIFARE Ultralight AES UIDRetrKey.

Property Description	
Property OID	1.2.1.1.4.3 / 0x0102010100403
Name	MIFARE Ultralight AES UIDRetrKey.
Description	The device uses this property to set 16-byte of the AES UIDRetrKey for MIFARE Ultralight AES authentication. On an example of the ASE Key = 000102030405060708090A0B0C0D0E0Fh, the setting should be 000102030405060708090A0B0C0D0E0F For security, the Get request for this property will always return 4-MSB of KCV. This key will be encrypted and stored in KPM.
Securing Key	None
Min. Len (b)	16
Max. Len (b)	16
Data Type	Binary
Valid Values	
Default	00000000 00000000 00000000 00000000

Table 8.3-27 - Get Request Example

Example (Hex)
AA 00 81 04 01 06 D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 01 89 0A E2 08 E1 06 E1 04 E4 02 C3 00

Table 8.3-28 - Get Response Example

Example (Hex)
AA00810482D3D10182040000000008482001ED10181072B06010401F609850101890EE20CE10AE108E406C304B548CFB4

Table 8.3-29 - Set Request Example

Example (Hex)
AA 00 81 04 01 55 D1 11 84 1F D1 11 85 01 01 87 04 02 01 01 04 89 12 C3 10 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

Table 8.3-30 - Set Response Example

Example (Hex)
AA00810482E5D11182040000000008482002AD11181072B06010401F609850101891AE218E116E114E412C3100001020304050607080 90A0B0C0D0E0F

8.3.1.7 Property 1.2.1.1.4.4 MIFARE Ultralight AES OriginalityKey.

Table 8.3-31 - Property 1.2.1.1.4.4 MIFARE Ultralight AES OriginalityKey.

Property Description	
Property OID	1.2.1.1.4.4 / 0x0102010100404
Name	MIFARE Ultralight AES OriginalityKey.
Description	The device uses this property to set 16-byte of the AES OriginalityKey for MIFARE Ultralight AES. On an example of the ASE Key = 000102030405060708090A0B0C0D0E0Fh, the setting should be 000102030405060708090A0B0C0D0E0F For security, the Get request for this property will always return 4-MSB of KCV. This key will be encrypted and stored in KPM.
Securing Key	None
Min. Len (b)	16
Max. Len (b)	16
Data Type	Binary
Valid Values	
Default	00000000 00000000 00000000 00000000

Table 8.3-32 - Get Request Example

Example (Hex)
AA 00 81 04 01 06 D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 01 89 0A E2 08 E1 06 E1 04 E4 02 C4 00

Table 8.3-33 - Get Response Example

Example (Hex)
AA00810482D3D10182040000000008482001ED10181072B06010401F609850101890EE20CE10AE108E406C404763CBCDE

Table 8.3-34 - Set Request Example

Example (Hex)
AA 00 81 04 01 55 D1 11 84 1F D1 11 85 01 01 87 04 02 01 01 04 89 12 C4 10 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

Table 8.3-35 - Set Response Example

Example (Hex)
AA00810482E5D11182040000000008482002AD11181072B06010401F609850101891AE218E116E114E412C4100001020304050607080 90A0B0C0D0E0F

8.3.1.8 Property 1.2.1.1.4.5 MIFARE Plus AES_Key1.

Table 8.3-36 - Property 1.2.1.1.4.5 MIFARE Plus AES_Key1.

Property Description	
Property OID	1.2.1.1.4.5 / 0x0102010100405
Name	MIFARE Plus AES_Key1.
Description	The device uses this property to set 16-byte of the AES_Key1 for MIFARE Plus. On example of the ASE Key = 000102030405060708090A0B0C0D0E0Fh, the setting should be 000102030405060708090A0B0C0D0E0F For security, the Get request for this property will always return 4-MSB of KCV. This key will be encrypted and stored in KPM.
Securing Key	None
Min. Len (b)	16
Max. Len (b)	16
Data Type	Binary
Valid Values	
Default	00000000 00000000 00000000 00000000

Table 8.3-37 - Get Request Example

Example (Hex)
AA 00 81 04 01 06 D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 01 89 0A E2 08 E1 06 E1 04 E4 02 C5 00

Table 8.3-38 - Get Response Example

Example (Hex)
AA00810482D3D10182040000000008482001ED10181072B06010401F609850101890EE20CE10AE108E406C504763CBCDE

Table 8.3-39 - Set Request Example

Example (Hex)
AA 00 81 04 01 55 D1 11 84 1F D1 11 85 01 01 87 04 02 01 01 04 89 12 C5 10 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

Table 8.3-40 - Set Response Example

Example (Hex)
AA00810482E5D11182040000000008482002AD11181072B06010401F609850101891AE218E116E114E412C5100001020304050607080 90A0B0C0D0E0F

8.3.1.9 Property 1.2.1.1.4.6 MIFARE Plus AES_Key2.

Table 8.3-41 - Property 1.2.1.1.4.6 MIFARE Plus AES_Key2.

Property Description	
Property OID	1.2.1.1.4.6 / 0x0102010100406
Name	MIFARE Plus AES_Key2.
Description	The device uses this property to set 16-byte of the AES_Key2 for MIFARE Plus. On example of the ASE Key = 000102030405060708090A0B0C0D0E0Fh, the setting should be 000102030405060708090A0B0C0D0E0F For security, the Get request for this property will always return 4-MSB of KCV. This key will be encrypted and stored in KPM.
Securing Key	None
Min. Len (b)	16
Max. Len (b)	16
Data Type	Binary
Valid Values	
Default	00000000 00000000 00000000 00000000

Table 8.3-42 - Get Request Example

Example (Hex)
AA 00 81 04 01 06 D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 01 89 0A E2 08 E1 06 E1 04 E4 02 C6 00

Table 8.3-43 - Get Response Example

Example (Hex)
AA00810482D3D10182040000000008482001ED10181072B06010401F609850101890EE20CE10AE108E406C604763CBCDE

Table 8.3-44 - Set Request Example

Example (Hex)
AA 00 81 04 01 55 D1 11 84 1F D1 11 85 01 01 87 04 02 01 01 04 89 12 C6 10 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

Table 8.3-45 - Set Response Example

Example (Hex)
AA00810482E5D11182040000000008482002AD11181072B06010401F609850101891AE218E116E114E412C6100001020304050607080 90A0B0C0D0E0F

8.3.1.10 Property 1.2.1.1.4.7 MIFARE Plus AES_Key3.

Table 8.3-46 - Property 1.2.1.1.4.7 MIFARE Plus AES_Key3.

Property Description	
Property OID	1.2.1.1.4.7 / 0x0102010100407
Name	MIFARE Plus AES_Key3.
Description	The device uses this property to set 16-byte of the AES_Key3 for MIFARE Plus. On example of the ASE Key = 000102030405060708090A0B0C0D0E0Fh, the setting should be 000102030405060708090A0B0C0D0E0F For security, the Get request for this property will always return 4-MSB of KCV. This key will be encrypted and stored in KPM.
Securing Key	None
Min. Len (b)	16
Max. Len (b)	16
Data Type	Binary
Valid Values	
Default	00000000 00000000 00000000 00000000

Table 8.3-47 - Get Request Example

Example (Hex)
AA 00 81 04 01 06 D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 01 89 0A E2 08 E1 06 E1 04 E4 02 C7 00

Table 8.3-48 - Get Response Example

Example (Hex)
AA00810482D3D10182040000000008482001ED10181072B06010401F609850101890EE20CE10AE108E406C704763CBCDE

Table 8.3-49 - Set Request Example

Example (Hex)
AA 00 81 04 01 55 D1 11 84 1F D1 11 85 01 01 87 04 02 01 01 04 89 12 C7 10 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

Table 8.3-50 - Set Response Example

Example (Hex)
AA00810482E5D11182040000000008482002AD11181072B06010401F609850101891AE218E116E114E412C710000102030405060708090A0B0C0D0E0F

8.3.1.11 Property 1.2.1.1.4.8 MIFARE Plus AES_Key4.

Table 8.3-51 - Property 1.2.1.1.4.8 MIFARE Plus AES_Key4.

Property Description	
Property OID	1.2.1.1.4.8 / 0x0102010100408
Name	MIFARE Plus AES_Key4.
Description	The device uses this property to set 16-byte of the AES_Key4 for MIFARE Plus. On example of the ASE Key = 000102030405060708090A0B0C0D0E0Fh, the setting should be 000102030405060708090A0B0C0D0E0F For security, the Get request for this property will always return 4-MSB of KCV. This key will be encrypted and stored in KPM.
Securing Key	None
Min. Len (b)	16
Max. Len (b)	16
Data Type	Binary
Valid Values	
Default	00000000 00000000 00000000 00000000

Table 8.3-52 - Get Request Example

Example (Hex)
AA 00 81 04 01 06 D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 01 89 0A E2 08 E1 06 E1 04 E4 02 C8 00

Table 8.3-53 - Get Response Example

Example (Hex)
AA00810482D3D10182040000000008482001ED10181072B06010401F609850101890EE20CE10AE108E406C804763CBCDE

Table 8.3-54 - Set Request Example

Example (Hex)
AA 00 81 04 01 55 D1 11 84 1F D1 11 85 01 01 87 04 02 01 01 04 89 12 C8 10 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

Table 8.3-55 - Set Response Example

Example (Hex)
AA00810482E5D11182040000000008482002AD11181072B06010401F609850101891AE218E116E114E412C810000102030405060708090A0B0C0D0E0F

8.3.1.12 Property 1.2.1.1.4.9 MIFARE Plus AES_Key5.

Table 8.3-56 - Property 1.2.1.1.4.9 MIFARE Plus AES_Key5.

Property Description	
Property OID	1.2.1.1.4.9 / 0x0102010100409
Name	MIFARE Plus AES_Key5.
Description	The device uses this property to set 16-byte of the AES_Key5 for MIFARE Plus. On example of the ASE Key = 000102030405060708090A0B0C0D0E0Fh, the setting should be 000102030405060708090A0B0C0D0E0F For security, the Get request for this property will always return 4-MSB of KCV. This key will be encrypted and stored in KPM.
Securing Key	None
Min. Len (b)	16
Max. Len (b)	16
Data Type	Binary
Valid Values	
Default	00000000 00000000 00000000 00000000

Table 8.3-57 - Get Request Example

Example (Hex)
AA 00 81 04 01 06 D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 01 89 0A E2 08 E1 06 E1 04 E4 02 C9 00

Table 8.3-58 - Get Response Example

Example (Hex)
AA00810482D3D10182040000000008482001ED10181072B06010401F609850101890EE20CE10AE108E406C904763CBCDE

Table 8.3-59 - Set Request Example

Example (Hex)
AA 00 81 04 01 55 D1 11 84 1F D1 11 85 01 01 87 04 02 01 01 04 89 12 C9 10 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

Table 8.3-60 - Set Response Example

Example (Hex)
AA00810482E5D11182040000000008482002AD11181072B06010401F609850101891AE218E116E114E412C9100001020304050607080 90A0B0C0D0E0F

8.3.1.13 Property 1.2.1.1.4.A MIFARE Plus AES_Key6.

Table 8.3-61 - Property 1.2.1.1.4.A MIFARE Plus AES_Key6.

Property Description	
Property OID	1.2.1.1.4.A / 0x010201010040A
Name	MIFARE Plus AES_Key6.
Description	The device uses this property to set 16-byte of the AES_Key6 for MIFARE Plus. On example of the ASE Key = 000102030405060708090A0B0C0D0E0Fh, the setting should be 000102030405060708090A0B0C0D0E0F For security, the Get request for this property will always return 4-MSB of KCV. This key will be encrypted and stored in KPM.
Securing Key	None
Min. Len (b)	16
Max. Len (b)	16
Data Type	Binary
Valid Values	
Default	00000000 00000000 00000000 00000000

Table 8.3-62 - Get Request Example

Example (Hex)
AA 00 81 04 01 06 D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 01 89 0A E2 08 E1 06 E1 04 E4 02 CA 00

Table 8.3-63 - Get Response Example

Example (Hex)
AA00810482D3D10182040000000008482001ED10181072B06010401F609850101890EE20CE10AE108E406CA04763CBCDE

Table 8.3-64 - Set Request Example

Example (Hex)
AA 00 81 04 01 55 D1 11 84 1F D1 11 85 01 01 87 04 02 01 01 04 89 12 CA 10 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

Table 8.3-65 - Set Response Example

Example (Hex)
AA00810482E5D11182040000000008482002AD11181072B06010401F609850101891AE218E116E114E412CA1000010203040506070809 0A0B0C0D0E0F

8.3.2 Property Subgroup 1.2.2.2.nn USB Settings (USB Only)

8.3.2.1 Property 1.2.2.2.1.1 Reduce Power During USB Suspend

Table 8.3-66 - Property 1.2.2.2.1.1 Reduce Power During USB Suspend

Property Description	
Property OID	1.2.2.2.1.1 / 0x010202020101
Name	Reduce Power During USB Suspend
Description	The device uses this property to determine whether it should reduce its power consumption when a USB host directs it to suspend. If this property is set to Disabled, the device will not have a USB compliant current draw when suspended, which, for example, allows it to continue to turn on LEDs and the display if present. To activate changes made to this property, the device must be reset or power cycled.
Securing Key	None
Min. Len (b)	1
Max. Len (b)	1
Data Type	Binary
Valid Values	0x00 = Disabled 0x01 = Enabled
Default	0x01

Table 8.3-67 - Get Request Example

Example (Hex)
AA00 8104 0155D101 840F D101 8501 01 8704 02020201 8902 C100

Table 8.3-68 - Get Response Example

Example (Hex)
AA0081048255D10182040000000084820010D10185010187040202020189 03C10101

Table 8.3-69 - Set Request Example

Example (Hex)
AA00 81040155D111 8410 D111 850101 8704 02020201 8903 C10101

Table 8.3-70 - Set Response Example

Example (Hex)
AA0081048255D11182040000000084820010D11185010187040202020189 03C10101

8.3.2.2 Property 1.2.2.2.1.2 USB Configuration Type

Table 8.3-71 - Property 1.2.2.2.1.2 USB Configuration Type

Property Description	
Property OID	1.2.2.2.1.2 / 0x010202020102
Name	USB Configuration Type
Description	The device uses this property to determine if it should behave as a USB HID device or an USB iAP2 device. USB HID devices can communicate with most hosts except for Apple hosts. USB iAP2 devices can communicate to Apple hosts like iPads and iPhones. The behavior affects USB enumeration and communications. To activate changes made to this property, the device must be reset or power cycled.
Securing Key	None
Min. Len (b)	1
Max. Len (b)	1
Data Type	Binary
Valid Values	0x00 = HID only 0x01 = iAP2 only 0x02 = autodetect (iAP2 with HID fallback) The autodetect option can allow a device to work with iAP2 hosts and HID hosts. With this option, every time the device is attached to a USB host it will first enumerate as an iAP2 device, if the device does not receive an iAP2 initialization sequence response from the host within 2 seconds the device will perform a soft USB detach from the host and then a soft USB attach and next enumerate as a USB HID device.
Default	0x02

Table 8.3-72 - Get Request Example

Example (Hex)
AA00 8104 0155D101 840F D101 8501 01 8704 02020201 8902 C200

Table 8.3-73 - Get Response Example (HID only)

Example (Hex)
AA0081048255D10182040000000084820010D10185010187040202020189 03C20100

Table 8.3-74 - Set Request Example (HID only)

Example (Hex)
AA00 81040155D111 8410 D111 850101 8704 02020201 8903 C20100

Table 8.3-75 - Set Response Example (HID only)

Example (Hex)
AA0081048255D11182040000000084820010D11185010187040202020189 03C20100

Table 8.3-76 - Set Request Example (iAP2 only)

Example (Hex)
AA00 81040155D111 8410 D111 850101 8704 02020201 8903 C20101

Table 8.3-77 - Set Response Example (iAP2 only)

Example (Hex)
AA0081048255D11182040000000084820010D11185010187040202020189 03C20101

Table 8.3-78 - Set Request Example (autodetect (iAP2 with HID fallback))

Example (Hex)
AA00 81040155D111 8410 D111 850101 8704 02020201 8903 C20102

Table 8.3-79 - Set Response Example (autodetect (iAP2 with HID fallback))

Example (Hex)
AA0081048255D11182040000000084820010D11185010187040202020189 03C20102

8.3.3 Property Subgroup 1.2.3.nnn User Interface Settings

8.3.3.1 Property 1.2.3.1.3.1 Enable Card Logos Page at Startup

Table 8.3-80- Property 1.2.3.1.3.1 Enable Card Logos Page at Startup

Property Description	
Property OID	1.2.3.1.3.1 / 0x010203010301
Name	Enable Card Logos Page at Startup
Description	Enables a page at startup that displays supported payment type brand logos.
Securing Key	None
Min. Len (b)	1
Max. Len (b)	1

Property Description	
Data Type	Binary
Valid Values	0x00 = Disable Logo Page at Startup 0x01 = Enable Logo Page at Startup
Default	0x00

Table 8.3-81 - Get Request Example

Example (Hex)
AA 00 81 04 01 5E D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 01 89 0A E2 08 E3 06 E1 04 E3 02 C1 00

Table 8.3-82 - Get Response Example

Example (Hex)
AA 00 81 04 82 5E D1 01 82 04 00 00 00 00 84 82 00 1B D1 01 81 07 2B 06 01 04 01 F6 09 85 01 01 89 0B E2 09 E3 07 E1 05 E3 03 C1 01 00

Table 8.3-83 - Set Request Example

Example (Hex)
AA 00 81 04 01 5D D1 11 84 1B D1 11 81 07 2B 06 01 04 01 F6 09 85 01 01 89 0B E2 09 E3 07 E1 05 E3 03 C1 01 01

Table 8.3-84 - Set Response Example

Example (Hex)
AA 00 81 04 82 5D D1 11 82 04 00 00 00 00 84 82 00 1B D1 11 81 07 2B 06 01 04 01 F6 09 85 01 01 89 0B E2 09 E3 07 E1 05 E3 03 C1 01 01

8.3.3.2 Property 1.2.3.1.3.2 Display Card Logo While Authorizing

Table 8.3-85- Property 1.2.3.1.3.2 Display Card Logo While Authorizing

Property Description	
Property OID	1.2.3.1.3.2 / 0x010203010302
Name	Display Card Logo While Authorizing
Description	Enable display of logo for card brand used in transaction
Securing Key	None
Min. Len (b)	1
Max. Len (b)	1
Data Type	Binary
Valid Values	0x00 = Disable Logo 0x01 = Enable Logo

Property Description	
Default	0x00

Table 8.3-86 - Get Request Example

Example (Hex)
AA 00 81 04 01 5E D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 01 89 0A E2 08 E3 06 E1 04 E3 02 C2 00

Table 8.3-87 - Get Response Example

Example (Hex)
AA 00 81 04 82 5E D1 01 82 04 00 00 00 00 84 82 00 1B D1 01 81 07 2B 06 01 04 01 F6 09 85 01 01 89 0B E2 09 E3 07 E1 05 E3 03 C2 01 00

Table 8.3-88 - Set Request Example

Example (Hex)
AA 00 81 04 01 5D D1 11 84 1B D1 11 81 07 2B 06 01 04 01 F6 09 85 01 01 89 0B E2 09 E3 07 E1 05 E3 03 C2 01 01

Table 8.3-89 - Set Response Example

Example (Hex)
AA 00 81 04 82 5D D1 11 82 04 00 00 00 00 84 82 00 1B D1 11 81 07 2B 06 01 04 01 F6 09 85 01 01 89 0B E2 09 E3 07 E1 05 E3 03 C2 01 01

8.3.3.3 Property 1.2.3.2.1.1 System Volume Control

Table 8.3-90- Property 1.2.3.2.1.1 System Volume Control

Property Description	
Property OID	1.2.3.2.1.1 / 0x010203020101
Name	System Volume Control
Description	This property sets the output volume for all sounds the device produces after startup, expressed as a percentage range from 0% (no sound) to 100%. The volume setting takes effect immediately after the host changes it. Note that devices with limited volume control hardware allocate ranges of numbers to represent the same physical volume level (for example, 1..49 = Low, 50..100 = High).
Securing Key	None
Min. Len (b)	1
Max. Len (b)	1
Data Type	Binary
Valid Values	0x00..0x64

Property Description	
Default	0x32 (50%)

Table 8.3-91 - Get Request Example

Example (Hex)
AA 00 81 04 01 5E D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 01 89 0A E2 08 E3 06 E2 04 E1 02 C1 00

Table 8.3-92 - Get Response Example

Example (Hex)
AA 00 81 04 82 5E D1 01 82 04 00 00 00 00 84 82 00 1B D1 01 81 07 2B 06 01 04 01 F6 09 85 01 01 89 0B E2 09 E3 07 E2 05 E1 03 C1 01 32

Table 8.3-93 - Set Request Example

Example (Hex)
AA 00 81 04 01 5D D1 11 84 1B D1 11 81 07 2B 06 01 04 01 F6 09 85 01 01 89 0B E2 09 E3 07 E2 05 E1 03 C1 01 01

Table 8.3-94 - Set Response Example

Example (Hex)
AA 00 81 04 82 5D D1 11 82 04 00 00 00 00 84 82 00 1B D1 11 81 07 2B 06 01 04 01 F6 09 85 01 01 89 0B E2 09 E3 07 E2 05 E1 03 C1 01 01

8.3.4 Property Subgroup 1.2.5.nnn Security Settings

8.3.4.1 Property 1.2.5.2.1.1 Device Lock State

Table 8.3-95- Property 1.2.5.2.1.1 Device Lock State

Property Description	
Property OID	1.2.5.2.1.1 / 0x010205020101
Name	Device Lock State
Description	This property can be used to get or set the device lock state. Getting this property does not require security, however setting it does. Setting it requires the use of **Command 0xD112 - Set Property (Secured)** which requires MagTek's involvement. To set the device lock state without involving MagTek use Command 0xEF04 – Load LTPK Protection Key. The value of the device lock state will revert to the value of **Property 1.2.5.2.1.2 Device Lock State After Reset** after a reset or a power cycle. See **Device Lock Feature** for more information.
Securing Key	See **Command 0xD112 - Set Property (Secured).**
Min. Len (b)	1
Max. Len (b)	1
Data Type	Binary

Property Description	
Valid Values	0x00 = Unlocked, 0x01 = Locked
Default	See **Property 1.2.5.2.1.2 Device Lock State After Reset.**

Table 8.3-96 - Get Request Example

Example (Hex)
AA00 8104 0155D101 840F D101 8501 01 8704 02050201 8902 C100

Table 8.3-97 - Get Response Example

Example (Hex)
AA00 81048255D101 820400000000 84820010 D101 850101 870402050201 8903 C101 00

Table 8.3-98 - Set Request Example

Example (Hex)
TBD

Table 8.3-99 - Set Response Example

Example (Hex)
TBD

8.3.4.2 Property 1.2.5.2.1.2 Device Lock State After Reset

Table 8.3-100- Property 1.2.5.2.1.2 Device Lock State After Reset

Property Description	
Property OID	1.2.5.2.1.2 / 0x010205020102
Name	Device Lock State After Reset
Description	This property can be used to get or set what **Property 1.2.5.2.1.1 Device Lock State** will be set to after the device is reset or power cycled. Systems that use the optional device lock feature should set this property to locked, otherwise all someone would need to do to unlock the device would be to power cycle or reset it. The value of the device lock state after reset property is stored in non-volatile memory so changes made to it will persist after the device is reset or power cycled. See **Device Lock Feature** for more information.
Securing Key	None
Min. Len (b)	1
Max. Len (b)	1
Data Type	Binary
Valid Values	0x00 = Unlocked, 0x01 = Locked
Default	0x00

Table 8.3-101 - Get Request Example

Example (Hex)
AA00 8104 0155D101 840F D101 8501 01 8704 02050201 8902 C200

Table 8.3-102 - Get Response Example

Example (Hex)
AA00 81048255D101 820400000000 84820010 D101 850101 870402050201 8903 C201 00

Table 8.3-103 - Set Request Example

Example (Hex)
AA00 81040155D111 8410 D111 8501 01 8704 02050201 8903 C201 00

Table 8.3-104 - Set Response Example

Example (Hex)
AA00 81048255D111 820400000000 84820010 D111 850101 870402050201 8903 C201 00

8.3.4.3 Property 1.2.5.2.1.3 Device Lock Passcode

Table 8.3-105- Property 1.2.5.2.1.3 Device Lock Passcode

Property Description	
Property OID	1.2.5.2.1.3 / 0x010205020103
Name	Device Lock Passcode
Description	This property can be used to set the device lock passcode. The value of the device lock passcode is stored in non-volatile memory so changes made to it will persist after the device is reset or power cycled. For security, the host can't get the value of the device lock passcode from the device. If the host gets this property, the device will always return a length of 0 and no value. Setting this property requires security. Setting it requires the use of **Command 0xD112 - Set Property (Secured)** which requires MagTek's involvement. To set the device lock passcode without involving MagTek use **Command 0xEF07 – Change Device Lock Passcode**. See **Device Lock Feature** for more information.
Securing Key	See **Command 0xD112 - Set Property (Secured).**
Min. Len (b)	4
Max. Len (b)	63
Data Type	Binary
Valid Values	It can only contain any printable ASCII character.
Default	0x34 0x33 0x32 0x31 ("4321")

Table 8.3-106 - Get Request Example

Example (Hex)
AA00 8104 0155D101 840F D101 8501 01 8704 02050201 8902 C300

Table 8.3-107 - Get Response Example

Example (Hex)
AA00 81048255D101 820400000000 8482000F D101 850101 870402050201 8902 C300

Table 8.3-108 - Set Request Example

Example (Hex)
TBD

Table 8.3-109 - Set Response Example

Example (Hex)
TBD

8.3.5 Property Subgroup 1.2.7.nnn System Settings

8.3.5.1 Property 1.2.7.1.1.1 Device Reset Occurred Notification Control

Table 8.3-110 - Property 1.2.7.1.1.1 Device Reset Occurred Notification Control

Property Description	
Property OID	1.2.7.1.1.1 / 0x010207010101
Name	Device Reset Occurred Notification Control
Description	The device uses this property to control behavior of the **Device Reset Occurred** notification in **Notification 0x1001 - Device Information Update**. Changes to this property do not take effect until the device is power cycled or reset.
Securing Key	None
Min. Len (b)	1
Max. Len (b)	1
Data Type	Binary
Valid Values	0x00 = Never send this notification. 0x01..0xFE = The frequency in seconds the device should repeat sending the notification until the host requests that it stop with **Property 1.2.7.1.1.2 Device Reset Occurred Notification Acknowledged**. 0xFF = Only send this notification once.
Default	0x00

Table 8.3-111 - Get Request Example

Example (Hex)
AA 00 81 04 01 55 D1 01 84 0F D1 01 85 01 01 87 04 02 07 01 01 89 02 C1 00

Table 8.3-112 - Get Response Example

Example (Hex)
AA 00 81 04 82 55 D1 01 82 04 00 00 00 00 84 82 00 10 D1 01 85 01 01 87 04 02 07 01 01 89 03 C1 01 00

Table 8.3-113 - Set Request Example

Example (Hex)
AA 00 81 04 01 55 D1 11 84 10 D1 11 85 01 01 87 04 02 07 01 01 89 03 C1 01 05

Table 8.3-114 - Set Response Example

Example (Hex)
AA 00 81 04 82 55 D1 11 82 04 00 00 00 00 84 82 00 10 D1 11 85 01 01 87 04 02 07 01 01 89 03 C1 01 05

8.3.5.2 Property 1.2.7.1.1.2 Device Reset Occurred Notification Acknowledged

Table 8.3-115 - Property 1.2.7.1.1.2 Device Reset Occurred Notification Acknowledged

Property Description	
Property OID	1.2.7.1.1.2 / 0x010207010102
Name	Device Reset Occurred Notification Acknowledged
Description	The host can use this property to acknowledge it has received the **Device Reset Occurred** notification in **Notification 0x1001 - Device Information Update** and to request that the device stop sending it. Alternatively, because the device automatically sets the value of this property to 0x00 on boot, the host can use this property to detect power cycles or resets using a polling method. To implement this, the host should read the value on a set schedule (for example, every 2 seconds). If the host finds a value of 0x00, a power cycle or reset has occurred, and the host should set the value back to 0x01. From that point until the next power cycle or reset, the value will remain 0x01. Changes made to this property will not persist after a power cycle or reset. After a power cycle or reset, this property has a value of 0x00 until the host changes it to 0x01.
Securing Key	None
Min. Len (b)	1
Max. Len (b)	1
Data Type	Binary

Property Description	
Valid Values	0x00 = Not acknowledged. Only the device can set to this value. 0x01 = Acknowledged. Do not continue to send the notification.
Default	0x00

Table 8.3-116 - Get Request Example

Example (Hex)
AA00 8104 0155D101 840F D101 8501 01 8704 02070101 8902 C200

Table 8.3-117 - Get Response Example

Example (Hex)
AA00 8104 8255D101 8204 00000000 84820010 D101 8501 01 8704 02070101 8903 C201 00

Table 8.3-118 - Set Request Example

Example (Hex)
AA00 81040155D111 8410 D111 8501 01 8704 02070101 8903 C20101

Table 8.3-119 - Set Response Example

Example (Hex)
AA00 8104 8255D111 8204 00000000 84820010 D111 8501 01 8704 02070101 8903 C201 01

8.3.5.3 Property 1.2.7.1.1.3 Device Reset Will Occur Soon Notification Control

Table 8.3-120 - Property 1.2.7.1.1.3 Device Reset Will Occur Soon Notification Control

Property Description	
Property OID	1.2.7.1.1.3 / 0x010207010103
Name	Device Reset Will Occur Soon Notification Control
Description	The host can use this property to control behavior of the **Device Reset Will Occur Soon** notification in **Notification 0x1001 - Device Information Update**. See **24 Hour Automatic Reset PCI Requirement** for more information. Changes to this property do not take effect until the device is power cycled or reset.
Securing Key	None
Min. Len (b)	1
Max. Len (b)	1
Data Type	Binary
Valid Values	0x00 = Never send this notification. 0x01..0xFF = Number of minutes before a reset to send the notification message.
Default	0x03

Table 8.3-121 - Get Request Example

Example (Hex)
AA 00 81 04 01 55 D1 01 84 0F D1 01 85 01 01 87 04 02 07 01 01 89 02 C3 00

Table 8.3-122 - Get Response Example

Example (Hex)
AA 00 81 04 82 55 D1 01 82 04 00 00 00 00 84 82 00 10 D1 01 85 01 01 87 04 02 07 01 01 89 03 C3 01 05

Table 8.3-123 - Set Request Example

Example (Hex)
AA 00 81 04 01 55 D1 11 84 10 D1 11 85 01 01 87 04 02 07 01 01 89 03 C1 01 05

Table 8-115 - Set Response Example

Example (Hex)
AA 00 81 04 82 55 D1 11 82 04 00 00 00 00 84 82 00 10 D1 11 85 01 01 87 04 02 07 01 01 89 03 C1 01 05

8.3.5.4 Property 1.2.7.1.1.4 Auto Reset Configuration

Table 8.3-124 - Property 1.2.7.1.1.4 Auto Reset Configuration

Property Description	
Property OID	1.2.7.1.1.4 / 0x010207010104
Name	Auto Reset Configuration
Description	This property controls the device's auto reset feature. The auto reset feature can be configured such that the device automatically resets 23 hours after booting up or at a specific time of day. The auto-reset feature cannot be disabled. See **24 Hour Automatic Reset PCI Requirement** for more information. Changes to this property do not take effect until the device is power cycled or reset.
Securing Key	None
Min. Len (b)	2
Max. Len (b)	2
Data Type	Binary
Valid Values	First byte 0 - 23 hours and second byte 0 - 59 minutes = Reset at the time of day specified. The time of day specified should be 24-hour Universal Time Coordinated (UTC). For example, values of 0 0 would be 12:00am UTC and values of 23 59 (0x17 0x3B) would be 11:59pm UTC. First byte 0xFF and second byte 0xFF = Auto reset 23 hours after booting up.
Default	0xFF 0xFF

Table 8.3-125 - Get Request Example

Example (Hex)
AA00 8104 0155D101 840F D101 8501 01 8704 02070101 8902 C400

Table 8.3-126 - Get Response Example

Example (Hex)
AA0081048255D10182040000000084820011D10185010187040207010189 04C402FFFF

Table 8.3-127 - Set Request Example

Example (Hex)
AA00 81040155D111 8411 D111 8501 01 8704 02070101 8904 C402 FFFF

Table 8.3-128 - Set Response Example

Example (Hex)
AA0081048255D11182040000000084820011D11185010187040207010189 04C402FFFF

8.3.5.5 Property 1.2.7.1.2.1 User Event Notification Controls Enable

Table 8.3-129 - Property 1.2.7.1.2.1 User Event Notification Controls Enable

Bit Property Description	
Property OID	1.2.7.1.2.1 / 0x010207010201
Name	User Event Notifications Enable
Description	The host can use this property to enable **notification reasons** defined in **Notification 0x1001 - Device Information Update** in the **User Events** Category. Each bit enables a specific **Reason** by setting that bit to 1. Byte 0 is the first byte, bit 0 is the LSB of each byte. The device only detects these user events and sends these notifications when it is idle. While processing a command (not idle), such as when processing a transaction with **Command 0x1001 - Start Transaction**, the device only sends notifications related to the command it is currently processing. Notification reasons may consume power when idle. For example, enabling Contactless reasons requires the device to continuously consume some radio frequency power to detect the presence of a contactless card when idle. To conserve power, only enable the notification reasons that are required by the solution design, if any. The readers and their associated notifications will be disabled if the battery charge is 5 percent or lower. Changes to this property do not take effect until the device is power cycled or reset.
Securing Key	None
Min. Len (b)	4
Max. Len (b)	4
Data Type	Binary

Bit Property Description	
Valid Values	Byte 0 • bit 0 = Reason: Contactless Card Presented • bit 1 = Reason: Contactless Card Removed • bit 2 = Reason: Card Seated in Slot • bit 3 = Reason: Card Unseated from Slot • bit 4 = Reason: Card Swiped • bit 5 = Reason: Touch Sensor Press Detected • bit 6 = Reason: Touch Sensor Release Detected • bit 7 = Reason: Barcode Read Detected Byte 1 • bit 0 = Reason: Encrypt Barcode Reader Event Data • bit 1 = Reserved (set to 0x00) • etc. Byte 2 • bit 0 = Reserved (set to 0x00) • etc. Byte 3 • bit 0 = Reserved (set to 0x00) • etc. Any byte / bit not listed here is reserved for future use. The host should set those values to 0, so if a future revision of firmware starts using those bits, the related notifications will be disabled and will not affect the existing host software.
Default	0x00, 0x00, 0x00, 0x00 (all User Event notifications disabled)

Table 8.3-130 - Get Request Example

Example (Hex)
AA00 8104 0155D101 840F D101 8501 01 8704 02070102 8902 C100

Table 8.3-131 - Get Response Example

Example (Hex)
AA00 8104 8255D101 8204 00000000 84820013 D101 8501 01 8704 02070102 8906 C1 04 00 00 00 00

Table 8.3-132 - Set Request Example

Example (Hex)
AA00 8104 0155D111 8413 D111 8501 01 8704 02070102 8906 C1 04 03 00 00 00

Table 8.3-133 - Set Response Example

Example (Hex)
AA00 8104 8255D111 8204 00000000 84820013 D111 8501 01 8704 02070102 8906 C1 04 03 00 00 00

8.3.5.6 Property 1.2.7.1.3.1 Maximum Battery Charge Level (Deprecated)

Table 8.3-134 - Property 1.2.7.1.3.1 Maximum Battery Charge Level (Deprecated)

Property Description	
Property OID	1.2.7.1.3.1 / 0x010207010301
Name	Maximum Battery Charge Level
Description	This OID has been deprecated. The OID can be written or read but it will have no effect of the device's behavior. The host can use this property to control the maximum charge level for the battery. All charge percentages reported or displayed will use this percentage as 100% charged. Setting this value to lower than 0x64 (100%) will increase the life of the battery but reduce the run time when running from the battery. The Maximum Battery Charge Level takes effect immediately after the host changes it.
Securing Key	None
Min. Len (b)	1
Max. Len (b)	1
Data Type	Binary
Valid Values	0x0A..0x64
Default	0x64 (100%) if WLAN device 0x50 (80%) for all other devices

Table 8.3-135 - Get Request Example

Example (Hex)
AA00 8104 0155D101 840F D101 8501 01 8704 02070103 8902 C100

Table 8.3-136 - Get Response Example

Example (Hex)
AA00 8104 8255D101 8204 00000000 84820010 D101 8501 01 8704 02070103 8903 C101 50

Table 8.3-137 - Set Request Example

Example (Hex)
AA00 81040155D111 8410 D111 8501 01 8704 02070103 8903 C101 50

Table 8.3-222 - Set Response Example

Example (Hex)
AA00 8104 8255D111 8204 00000000 84820010 D111 8501 01 8704 02070103 8903 C101 50

8.3.5.7 Property 1.2.7.1.4.1 Device Low Temperature Notification Level

Table 8.3-138 - Property 1.2.7.1.4.1 Device Low Temperature Notification Level

Property Description	
Property OID	1.2.7.1.4.1 / 0x010207010401
Name	Device Low Temperature Notification Level
Description	The device will send a low temperature notification when the device's temperature falls below this temperature. The temperature is in degrees Celsius. The Device Low Temperature Notification Level takes effect immediately after the host changes it.
Securing Key	None
Min. Len (b)	1
Max. Len (b)	1
Data Type	Signed Binary
Valid Values	Temperature in Celsius: DynaProx: 0xE2 .. 0x55 (-30 .. 85) DynaProx BCR: 0xEC .. 0x37 (-20 .. 55) DynaFlex: 0x00 .. 0x2D (0 .. 45) This value must be less than that set in 1.2.7.1.4.2.
Default	Temperature in Celsius: DynaProx: 0xE2 (-30 DynaProx BCR: 0xEC (-20) DynaFlex: 0x00 (0)

Table 8.3-139 - Get Request Example

Example (Hex)
AA00 8104 0155D101 840F D101 8501 01 8704 02070104 8902 C100

Table 8.3-140 - Get Response Example

Example (Hex)
AA00 8104 8255D101 8204 00000000 84820010 D101 8501 01 8704 02070104 8903 C101 00

Table 8.3-141 - Set Request Example

Example (Hex)
AA00 81040155D111 8410 D111 8501 01 8704 02070104 8903 C101 05

Table 8.3-227 - Set Response Example

Example (Hex)
AA00 8104 8255D111 8204 00000000 84820010 D111 8501 01 870402070104 8903 C101 05

8.3.5.8 Property 1.2.7.1.4.2 Device High Temperature Notification Level

Table 8.3-142 - Property 1.2.7.1.4.2 Device High Temperature Notification Level

Property Description	
Property OID	1.2.7.1.4.2 / 0x010207010402
Name	Device High Temperature Notification Level
Description	The device will send a high temperature notification when the device's temperature rises above this temperature. The temperature is in degrees Celsius. The Device High Temperature Notification Level takes effect immediately after the host changes it.
Securing Key	None
Min. Len (b)	1
Max. Len (b)	1
Data Type	Signed Binary
Valid Values	Temperature in Celsius: DynaProx: 0xE2 .. 0x55 (-30 .. 85) DynaProx BCR: 0xEC .. 0x37 (-20 .. 55) DynaFlex: 0x00 .. 0x2D (0 .. 45) This value must be greater than that set in 1.2.7.1.4.1.
Default	Temperature in Celsius: DynaProx: 0x55 (85) DynaProx BCR: 0x37 (55) DynaFlex: 0x2D (45)

Table 8.3-143 - Get Request Example

Example (Hex)
AA00 8104 0155D101 840F D101 8501 01 8704 02070104 8902 C200

Table 8.3-230 - Get Response Example

Example (Hex)
AA00 8104 8255D101 8204 00000000 84820010 D101 8501 01 8704 02070104 8903 C201 2D

Table 8.3-144 - Set Request Example

Example (Hex)
AA00 81040155D111 8410 D111 8501 01 8704 02070104 8903 C201 2D

Table 8.3-232 - Set Response Example

Example (Hex)
AA00 8104 8255D111 8204 00000000 84820010 D111 8501 01 870402070104 8903 C201 2D

8.3.5.9 Property 1.2.7.1.4.3 Device Temperature Notification Repeat Interval

Table 8.3-145 - Property 1.2.7.1.4.3 Device Temperature Notification Repeat Interval

Property Description	
Property OID	1.2.7.1.4.3 / 0x010207010403
Name	Device Temperature Notification Repeat Interval
Description	While the device's temperature is outside of the range defined by Device Low Temperature Notification Level (1.2.7.1.4.1) and Device High Temperature Notification Level (1.2.7.1.4.2), notifications will be sent to the host. This property sets the period between notifications. The Device Temperature Notification Repeat Interval takes effect immediately after the host changes it.
Securing Key	None
Min. Len (b)	1
Max. Len (b)	1
Data Type	Binary
Valid Values	0x00 Send once. Do not repeat. 0x0F .. 0xFF Repeat period in seconds. Must be multiple of 0x0F (15).
Default	0x1E (30 seconds)

Table 8.3-146 - Get Request Example

Example (Hex)
AA00 8104 0155D101 840F D101 8501 01 8704 02070104 8902 C300

Table 8.3-235 - Get Response Example

Example (Hex)
AA00 8104 8255D101 8204 00000000 84820010 D101 8501 01 8704 02070104 8903 C301 1E

Table 8.36 - Set Request Example

Example (Hex)
AA00 81040155D111 8410 D111 8501 01 8704 02070104 8903 C301 1E

Table 8.3-237 - Set Response Example

Example (Hex)
AA00 8104 8255D111 8204 00000000 84820010 D111 8501 01 870402070104 8903 C301 1E

8.4 Property Group 2.1.2.nnn Device Firmware Identification Information

8.4.1 Property Subgroup 2.1.2.1.nn Boot Firmware Information

8.4.1.1 Property 2.1.2.1.1.2 Boot1 Firmware Version

Table 8.4-1 - Property 2.1.2.1.1.2 Boot1 Firmware Version

Property Description	
Property OID	2.1.2.1.1.2 / 0x020102010102
Name	Boot 1 Firmware Version
Description	The device uses this property to report its bootloader firmware version number in the form **PartNumber-Version-PCI**, padded with null characters. Example: 1000007536-A0-PCI
Securing Key	None
Min. Len (b)	19
Max. Len (b)	19
Data Type	Alphanumeric
Valid Values	Any string
Default	

Table 8.4-2 - Get Request Example

Example (Hex)
AA 00 81 04 01 05 D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 0A E1 08 E2 06 E1 04 E1 02 C2 00

Table 8.4-3 - Get Response Example

Example (Hex)
AA 00 81 04 82 05 D1 01 82 04 00 00 00 00 84 82 00 2D D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 1D E1 1B E2 19 E1 17 E1 15 C2 13 31 30 30 30 30 30 37 35 33 36 2D 41 30 2D 50 43 49 00 00

8.4.1.2 Property 2.1.2.1.1.4 Boot1 Firmware Part Number

Table 8.4-4 - Property 2.1.2.1.1.4 Boot1 Firmware Part Number

Property Description	
Property OID	2.1.2.1.1.4 / 0x020102010104
Name	Boot 1 Firmware Part Number

Property Description	
Description	The device uses this property to report its bootloader firmware part number as a string, padded with null characters.
Securing Key	None
Min. Len (b)	11
Max. Len (b)	11
Data Type	Alphanumeric
Valid Values	Any string
Default	

Table 8.4-5 - Get Request Example

Example (Hex)
AA 00 81 04 01 03 D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 0A E1 08 E2 06 E1 04 E1 02 C4 00

Table 8.4-6 - Get Response Example

Example (Hex)
AA 00 81 04 82 03 D1 01 82 04 00 00 00 00 84 82 00 25 D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 15 E1 13 E2 11 E1 0F E1 0D C4 0B 31 30 30 30 30 30 37 35 33 36 00

8.4.1.3 Property 2.1.2.1.2.2 Boot0 Firmware Version

Table 8.4-7 - Property 2.1.2.1.2.2 Boot0 Firmware Version

Property Description	
Property OID	2.1.2.1.2.2 / 0x020102010202
Name	Boot 0 Firmware Version
Description	The device uses this property to report its bootloader firmware version number in the form **PartNumber-Version-PCI**, padded with null characters. Example: 1000007535-A0-PCI
Securing Key	None
Min. Len (b)	19
Max. Len (b)	19
Data Type	Alphanumeric
Valid Values	Any string
Default	

Table 8.4-8 - Get Request Example

Example (Hex)
AA 00 81 04 01 05 D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 0A E1 08 E2 06 E1 04 E2 02 C2 00

Table 8.4-9 - Get Response Example

Example (Hex)
AA 00 81 04 82 05 D1 01 82 04 00 00 00 00 84 82 00 2D D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 1D E1 1B E2 19 E1 17 E1 15 C2 13 31 30 30 30 30 30 37 35 33 35 2D 41 30 2D 50 43 49 00 00

8.4.1.4 Property 2.1.2.1.2.4 Boot0 Firmware Part Number

Table 8.4-10 - Property 2.1.2.1.2.4 Boot0 Firmware Part Number

Property Description	
Property OID	2.1.2.1.2.4 / 0x020102010204
Name	Boot 0 Firmware Part Number
Description	The device uses this property to report its bootloader firmware part number as a string, padded with null characters.
Securing Key	None
Min. Len (b)	11
Max. Len (b)	11
Data Type	Alphanumeric
Valid Values	Any string
Default	

Table 8.4-11 - Get Request Example

Example (Hex)
AA 00 81 04 01 0A D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 0A E1 08 E2 06 E1 04 E2 02 C4 00

Table 8.4-12 - Get Response Example

Example (Hex)
AA 00 81 04 82 0A D1 01 82 04 00 00 00 00 84 82 00 25 D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 15 E1 13 E2 11 E1 0F E2 0D C4 0B 31 30 30 30 30 30 37 35 33 35 00

8.4.2 Property Subgroup 2.1.2.2.nn Core Firmware Information

8.4.2.1 Property 2.1.2.2.2.1 Device Model Name

Table 8.4-13 - Property 2.1.2.2.2.1 Device Model Name

Property Description	
Property OID	2.1.2.2.2.1 / 0x020102020201
Name	Device Model Name
Description	The device uses this property to report its model name as string, padded with null characters.
Securing Key	None
Min. Len (b)	10
Max. Len (b)	20
Data Type	Alphanumeric
Valid Values	Any string
Default	

Table 8.4-14 - Get Request Example

Example (Hex)
AA 00 81 04 01 06 D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 0A E1 08 E2 06 E2 04 E2 02 C1 00

Table 8.4-15 - Get Response Example

Example (Hex)
AA 00 81 04 82 06 D1 01 82 04 00 00 00 00 84 82 00 27 D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 17 E1 15 E2 13 E2 11 E2 0F C1 0D 44 79 6E 61 46 6C 65 78 20 50 72 6F 00

8.4.2.2 Property 2.1.2.2.2.2 Main Firmware Version

Table 8.4-16 - Property 2.1.2.2.2.2 Main Firmware Version

Property Description	
Property OID	2.1.2.2.2.2 / 0x020102020202
Name	Main Firmware Version
Description	The device uses this property to report its main firmware version number in the form **PartNumber-Version-PCI**, padded with null characters. Example: 1000007183-A0-PCI
Securing Key	None
Min. Len (b)	19
Max. Len (b)	19

Property Description	
Data Type	Alphanumeric
Valid Values	Any string
Default	

Table 8.4-17 - Get Request Example

Example (Hex)
AA 00 81 04 01 08 D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 0A E1 08 E2 06 E2 04 E2 02 C2 00

Table 8.4-18 - Get Response Example

Example (Hex)
AA 00 81 04 82 08 D1 01 82 04 00 00 00 00 84 82 00 2D D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 1D E1 1B E2 19 E2 17 E2 15 C2 13 31 30 30 30 30 30 37 31 38 33 2D 41 31 2D 50 43 49 00 00

8.4.2.3 Property 2.1.2.2.2.4 Main Firmware Part Number

Table 8.4-19 - Property 2.1.2.2.2.4 Main Firmware Part Number

Property Description	
Property OID	2.1.2.2.2.4 / 0x020102020204
Name	Main Firmware Part Number
Description	The device uses this property to report its main firmware part number as a string, padded with null characters.
Securing Key	None
Min. Len (b)	11
Max. Len (b)	11
Data Type	Alphanumeric
Valid Values	Any string
Default	

Table 8.4-20 - Get Request Example

Example (Hex)
AA 00 81 04 01 0A D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 0A E1 08 E2 06 E2 04 E2 02 C4 00

Table 8.4-21 - Get Response Example

Example (Hex)
AA 00 81 04 82 0A D1 01 82 04 00 00 00 00 84 82 00 25 D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 15 E1 13 E2 11 E2 0F E2 0D C4 0B 31 30 30 30 30 30 37 31 38 33 00

8.4.2.4 Property 2.1.2.2.2.6 Key Type

Table 8.4-22 - Property 2.1.2.2.2.6 Key Type

Property Description	
Property OID	2.1.2.2.2.6 / 0x020102020206
Name	Key Type
Description	The device's firmware can be compiled to use one of two available ECDSA P-521 key sets when signing and verifying the signature in **Command 0xEEEE - Send Secured Command to Device**. This property indicates which of the two keys the firmware is configured to expect the host and device to use: • 0xFF00 = Development Key • Any other value = Production Key
Securing Key	None
Min. Len (b)	2
Max. Len (b)	2
Data Type	Alphanumeric
Valid Values	Any string
Default	

Table 8.4-23 - Get Request Example

Example (Hex)
AA 00 81 04 01 31 D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 0A E1 08 E2 06 E2 04 E2 02 C6 00

Table 8.4-24 - Get Response Example

Example (Hex)
AA 00 81 04 82 31 D1 01 82 04 00 00 00 00 84 82 00 1C D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 0C E1 0A E2 08 E2 06 E2 04 C6 02 FF 00

8.4.3 Property Subgroup 2.1.2.3.nn EMV Firmware Information

8.4.3.1 Property 2.1.2.3.2.1 EMV Contact L1 Kernel ID (Contact Only)

Table 8.4-25 - Property 2.1.2.3.2.1 EMV Contact L1 Kernel ID (Contact Only)

Property Description	
Property OID	2.1.2.3.2.1 / 0x020102030201
Name	EMV Contact L1 Kernel ID

Property Description	
Description	The device uses this property to report the EMV Contact L1 Kernel ID as string, padded with null characters. Example: CT L1 EMVCO 4.3C
Securing Key	None
Min. Len (b)	17
Max. Len (b)	17
Data Type	Alphanumeric
Valid Values	Any string
Default	

Table 8.4-26 - Get Request Example

Example (Hex)
AA 00 81 04 01 12 D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 0A E1 08 E2 06 E3 04 E2 02 C1 00

Table 8.4-27 - Get Response Example

Example (Hex)
AA 00 81 04 82 12 D1 01 82 04 00 00 00 00 84 82 00 2B D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 1B E1 19 E2 17 E3 15 E2 13 C1 11 43 54 20 4C 31 20 45 4D 56 43 4F 20 34 2E 33 43 00

8.4.3.2 Property 2.1.2.3.2.2 EMV Contact L1 Kernel Firmware Part Number (Contact Only)

Table 8.4-28 - Property 2.1.2.3.2.2 EMV Contact L1 Kernel Firmware Part Number (Contact Only)

Property Description	
Property OID	2.1.2.3.2.2 / 0x020102030202
Name	EMV Contact L1 Kernel Firmware Part Number
Description	The device uses this property to report the EMV Contact L1 Kernel Part Number as string, padded with null characters. Example: 1000007176 Ver A
Securing Key	None
Min. Len (b)	17
Max. Len (b)	17
Data Type	Alphanumeric
Valid Values	Any string
Default	

Table 8.4-29 - Get Request Example

Example (Hex)
AA 00 81 04 01 13 D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 0A E1 08 E2 06 E3 04 E2 02 C2 00

Table 8.4-30 - Get Response Example

Example (Hex)
AA 00 81 04 82 13 D1 01 82 04 00 00 00 00 84 82 00 2B D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 1B E1 19 E2 17 E3 15 E2 13 C2 11 31 30 30 30 30 30 37 31 37 36 20 56 65 72 20 41 00

8.4.3.3 Property 2.1.2.3.3.1 EMV Contact L2 Kernel ID (Contact Only)

Table 8.4-31 - Property 2.1.2.3.3.1 EMV Contact L2 Kernel ID (Contact Only)

Property Description	
Property OID	2.1.2.3.3.1 / 0x020102030301
Name	EMV Contact L2 Kernel ID
Description	The device uses this property to report the EMV Contact L2 Kernel ID as string, padded with null characters. Example: CT L2 4.3K
Securing Key	None
Min. Len (b)	11
Max. Len (b)	11
Data Type	Alphanumeric
Valid Values	Any string
Default	

Table 8.4-32 - Get Request Example

Example (Hex)
AA 00 81 04 01 15 D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 0A E1 08 E2 06 E3 04 E3 02 C1 00

Table 8.4-33 - Get Response Example

Example (Hex)
AA 00 81 04 82 16 D1 01 82 04 00 00 00 00 84 82 00 25 D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 15 E1 13 E2 11 E3 0F E3 0D C1 0B 43 54 20 4C 32 20 34 2E 33 4B 00

8.4.3.4 Property 2.1.2.3.3.2 EMV Contact L2 Kernel Firmware Part Number(Contact Only)

Table 8.4-34 - Property 2.1.2.3.3.2 EMV Contact L2 Kernel Firmware Part Number(Contact Only)

Property Description	
Property OID	2.1.2.3.3.2 / 0x020102030302
Name	EMV Contact L2 Kernel Firmware Part Number
Description	The device uses this property to report the EMV Contact L2 Kernel Part Number as string, padded with null characters. Example: 1000008878 Ver A DynaFlex PED L2 Kernel
Securing Key	None
Min. Len (b)	40
Max. Len (b)	40
Data Type	Alphanumeric
Valid Values	Any string
Default	

Table 8.4-35 - Get Request Example

Example (Hex)
AA 00 81 04 01 1A D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 0A E1 08 E2 06 E3 04 E3 02 C2 00

Table 8.4-36 - Get Response Example

Example (Hex)
AA 00 81 04 82 1A D1 01 82 04 00 00 00 00 84 82 00 42 D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 32 E1 30 E2 2E E3 2C E3 2A C2 28 31 30 30 30 30 30 38 38 37 38 20 56 65 72 20 41 20 44 79 6E 61 46 6C 65 78 20 50 45 44 20 4C 32 20 4B 65 72 6E 65 6C 00

8.4.3.5 Property 2.1.2.3.3.3 EMV Contact L2 Kernel Checksum (Contact Only)

Table 8.4-37 - Property 2.1.2.3.3.3 EMV Contact L2 Kernel Checksum (Contact Only)

Property Description	
Property OID	2.1.2.3.3.3/ 0x020102030303
Name	EMV Contact L2 Kernel Checksum
Description	The device uses this property to report the EMV Contact L2 Kernel Checksum as string, no padding. Example: vxxCAAgnos33_17 b54f31bcb61a26fc823bce9ab8989b31ab90f9a4
Securing Key	None
Min. Len (b)	Variable

Property Description	
Max. Len (b)	75
Data Type	Alphanumeric
Valid Values	Any string
Default	

Table 8.4-38 - Get Request Example

Example (Hex)
AA 00 81 04 01 1B D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 0A E1 08 E2 06 E3 04 E3 02 C3 00

Table 8.4-39 - Get Response Example

Example (Hex)
AA 00 81 04 82 1B D1 01 82 04 00 00 00 00 84 82 00 52 D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 42 E1 40 E2 3E E3 3C E3 3A C3 38 76 78 78 43 41 41 67 6E 6F 73 33 33 5F 31 37 20 62 35 34 66 33 31 62 63 62 36 31 61 32 36 66 63 38 32 33 62 63 65 39 61 62 38 39 38 39 62 33 31 61 62 39 30 66 39 61 34

8.4.3.6 Property 2.1.2.3.3.4 EMV Contact L2 Kernel Configuration Checksum (Contact Only)

Table 8.4-40 - Property 2.1.2.3.3.4 EMV Contact L2 Kernel Configuration Checksum (Contact Only)

Property Description	
Property OID	2.1.2.3.3.4/ 0x020102030304
Name	EMV Contact L2 Kernel Configuration Checksum
Description	The device uses this property to report the EMV Contact L2 Kernel Configuration Checksum as string, no padding. Example: 17AC3C4A
Securing Key	None
Min. Len (b)	8
Max. Len (b)	8
Data Type	Alphanumeric
Valid Values	Any string
Default	

Table 8.4-41 - Get Request Example

Example (Hex)
AA 00 81 04 01 1F D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 0A E1 08 E2 06 E3 04 E3 02 C4 00

Table 8.4-42 - Get Response Example

Example (Hex)
AA 00 81 04 82 1F D1 01 82 04 00 00 00 00 84 82 00 1E D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 0E E1 0C E2 0A E3 08 E3 06 C4 04 17 AC 3C 4A

8.4.3.7 Property 2.1.2.3.4.1 EMV Contactless L1 Kernel ID (Contactless Only)

Table 8.4-43 - Property 2.1.2.3.4.1 EMV Contactless L1 Kernel ID (Contactless Only)

Property Description	
Property OID	2.1.2.3.4.1 / 0x020102030401
Name	EMV Contactless L1 Kernel ID
Description	The device uses this property to report the EMV Contactless L1 Kernel ID as string, padded with null characters. Example: CL L1 EMVCO 3.0
Securing Key	None
Min. Len (b)	16
Max. Len (b)	16
Data Type	Alphanumeric
Valid Values	Any string
Default	

Table 8.4-44 - Get Request Example

Example (Hex)
AA 00 81 04 01 21 D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 0A E1 08 E2 06 E3 04 E4 02 C1 00

Table 8.4-45 - Get Response Example

Example (Hex)
AA 00 81 04 82 21 D1 01 82 04 00 00 00 00 84 82 00 2A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 1A E1 18 E2 16 E3 14 E4 12 C1 10 43 4C 20 4C 31 20 45 4D 56 43 4F 20 33 2E 30 00

8.4.3.8 Property 2.1.2.3.4.2 EMV Contactless L1 Kernel Firmware Part Number (Contactless Only)

Table 8.4-46 - Property 2.1.2.3.4.2 EMV Contactless L1 Kernel Firmware Part Number (Contactless Only)

Property Description	
Property OID	2.1.2.3.4.2 / 0x020102030402
Name	EMV Contactless L1 Kernel Firmware Part Number

Property Description	
Description	The device uses this property to report the EMV Contactless L1 Kernel Part Number as string, padded with null characters. Example: 1000007177 Ver A
Securing Key	None
Min. Len (b)	17
Max. Len (b)	17
Data Type	Alphanumeric
Valid Values	Any string
Default	

Table 8.4-47 - Get Request Example

Example (Hex)
AA 00 81 04 01 25 D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 0A E1 08 E2 06 E3 04 E4 02 C2 00

Table 8.4-48 - Get Response Example

Example (Hex)
AA 00 81 04 82 25 D1 01 82 04 00 00 00 00 84 82 00 2B D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 1B E1 19 E2 17 E3 15 E4 13 C2 11 31 30 30 30 30 30 37 31 37 37 20 56 65 72 20 41 00

8.4.3.9 Property 2.1.2.3.4.3 EMV Contactless L1 Kernel Checksum (Contactless Only)

Table 8.4-49 - Property 2.1.2.3.4.3 EMV Contactless L1 Kernel Checksum (Contactless Only)

Property Description	
Property OID	2.1.2.3.4.3/ 0x020102030403
Name	EMV Contactless L1 Kernel Checksum
Description	The device uses this property to report the EMV Contactless L1 Kernel Checksum as string, no padding. Example: d3c5d413334178d1d5929a752b8d29adc1e5829c
Securing Key	None
Min. Len (b)	Variable
Max. Len (b)	75
Data Type	Alphanumeric
Valid Values	Any string
Default	

Table 8.4-50 - Get Request Example

Example (Hex)
AA 00 81 04 01 26 D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 0A E1 08 E2 06 E3 04 E4 02 C3 00

Table 8.4-51 - Get Response Example

Example (Hex)
AA 00 81 04 82 26 D1 01 82 04 00 00 00 00 84 82 00 42 D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 32 E1 30 E2 2E E3 2C E4 2A C3 28 64 33 63 35 64 34 31 33 33 33 34 31 37 38 64 31 64 35 39 32 39 61 37 35 32 62 38 64 32 39 61 64 63 31 65 35 38 32 39 63

8.4.3.10 Property 2.1.2.3.5.1 Mastercard MCL Kernel ID (Contactless Only)

Table 8.4-52 - Property 2.1.2.3.5.1 Mastercard MCL Kernel ID (Contactless Only)

Property Description	
Property OID	2.1.2.3.5.1 / 0x020102030501
Name	Mastercard MCL Kernel ID
Description	The device uses this property to report the Mastercard MCL Kernel ID as string, padded with null characters. Example: MCL 3.1.3
Securing Key	None
Min. Len (b)	10
Max. Len (b)	10
Data Type	Alphanumeric
Valid Values	Any string
Default	

Table 8.4-53 - Get Request Example

Example (Hex)
AA 00 81 04 01 28 D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 0A E1 08 E2 06 E3 04 E5 02 C1 00

Table 8.4-54 - Get Response Example

Example (Hex)
AA 00 81 04 82 28 D1 01 82 04 00 00 00 00 84 82 00 24 D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 14 E1 12 E2 10 E3 0E E5 0C C1 0A 4D 43 4C 20 33 2E 31 2E 33 00

8.4.3.11 Property 2.1.2.3.5.2 Mastercard MCL Kernel Firmware Part Number (Contactless Only)

Table 8.4-55 - Property 2.1.2.3.5.2 Mastercard MCL Kernel Firmware Part Number (Contactless Only)

Property Description	
Property OID	2.1.2.3.5.2 / 0x020102030502
Name	Mastercard MCL Firmware Part Number
Description	The device uses this property to report the Mastercard MCL Kernel Part Number as string, padded with null characters. Example: 1000007179 Ver A0
Securing Key	None
Min. Len (b)	18
Max. Len (b)	18
Data Type	Alphanumeric
Valid Values	Any string
Default	

Table 8.4-56 - Get Request Example

Example (Hex)
AA 00 81 04 01 29 D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 0A E1 08 E2 06 E3 04 E5 02 C2 00

Table 8.4-57 - Get Response Example

Example (Hex)
AA 00 81 04 82 29 D1 01 82 04 00 00 00 00 84 82 00 2C D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 1C E1 1A E2 18 E3 16 E5 14 C2 12 31 30 30 30 30 30 37 31 37 39 20 56 65 72 20 41 30 00

8.4.3.12 Property 2.1.2.3.5.3 Mastercard MCL Kernel Checksum (Contactless Only)

Table 8.4-58 - Property 2.1.2.3.5.3 Mastercard MCL Kernel Checksum (Contactless Only)

Property Description	
Property OID	2.1.2.3.5.3/ 0x020102030503
Name	Mastercard MCL Kernel Checksum
Description	The device uses this property to report the Mastercard MCL Kernel Checksum as string, no padding. Example: C2.2.8 -> v1.0.2 [ade94c13b0c6a31f1be682b12536528264b1efc0]
Securing Key	None
Min. Len (b)	Variable

Property Description	
Max. Len (b)	75
Data Type	Alphanumeric
Valid Values	Any string
Default	

Table 8.4-59 - Get Request Example

Example (Hex)
AA 00 81 04 01 2A D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 0A E1 08 E2 06 E3 04 E5 02 C3 00

Table 8.4-60 - Get Response Example

Example (Hex)
AA 00 81 04 82 2A D1 01 82 04 00 00 00 00 84 82 00 55 D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 45 E1 43 E2 41 E3 3F E5 3D C3 3B 43 32 2E 32 2E 38 20 2D 3E 20 76 31 2E 30 2E 32 20 5B 61 64 65 39 34 63 31 33 62 30 63 36 61 33 31 66 31 62 65 36 38 32 62 31 32 35 33 36 35 32 38 32 36 34 62 31 65 66 63 30 5D

8.4.3.13 Property 2.1.2.3.6.1 Visa payWave Kernel ID (Contactless Only)

Table 8.4-61 - Property 2.1.2.3.6.1 Visa payWave Kernel ID (Contactless Only)

Property Description	
Property OID	2.1.2.3.6.1 / 0x020102030601
Name	Visa payWave Kernel ID
Description	The device uses this property to report the Visa payWave Kernel ID as string, padded with null characters. Example: PayWave 2.2
Securing Key	None
Min. Len (b)	12
Max. Len (b)	12
Data Type	Alphanumeric
Valid Values	Any string
Default	

Table 8.4-62 - Get Request Example

Example (Hex)
AA 00 81 04 01 03 D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 0A E1 08 E2 06 E3 04 E6 02 C1 00

Table 8.4-63 - Get Response Example

Example (Hex)
AA 00 81 04 82 03 D1 01 82 04 00 00 00 00 84 82 00 26 D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 16 E1 14 E2 12 E3 10 E6 0E C1 0C 50 61 79 57 61 76 65 20 32 2E 32 00

8.4.3.14 Property 2.1.2.3.6.2 Visa payWave Kernel Firmware Part Number (Contactless Only)

Table 8.4-64 - Property 2.1.2.3.6.2 Visa payWave Kernel Firmware Part Number (Contactless Only)

Property Description	
Property OID	2.1.2.3.6.2 / 0x020102030602
Name	Visa payWave Firmware Part Number
Description	The device uses this property to report the Visa payWave Kernel Part Number as string, padded with null characters. Example: 1000007180 Ver A1
Securing Key	None
Min. Len (b)	18
Max. Len (b)	18
Data Type	Alphanumeric
Valid Values	Any string
Default	

Table 8.4-65 - Get Request Example

Example (Hex)
AA 00 81 04 01 05 D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 0A E1 08 E2 06 E3 04 E6 02 C2 00

Table 8.4-66 - Get Response Example

Example (Hex)
AA 00 81 04 82 05 D1 01 82 04 00 00 00 00 84 82 00 2C D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 1C E1 1A E2 18 E3 16 E6 14 C2 12 31 30 30 30 30 30 37 31 38 30 20 56 65 72 20 41 31 00

8.4.3.15 Property 2.1.2.3.6.3 Visa payWave Kernel Checksum (Contactless Only)

Table 8.4-67 - Property 2.1.2.3.6.3 Visa payWave Kernel Checksum (Contactless Only)

Property Description	
Property OID	2.1.2.3.6.3/ 0x020102030603
Name	Visa payWave Kernel Checksum

Property Description	
Description	The device uses this property to report the Visa payWave Kernel Checksum as string, no padding. Example: VCPS 2.2x [00000000]-> v1.5.6 [F1CBB8A23E00984E9E753EF4884C33EA368E570C]
Securing Key	None
Min. Len (b)	Variable
Max. Len (b)	75
Data Type	Alphanumeric
Valid Values	Any string
Default	

Table 8.4-68 - Get Request Example

Example (Hex)
AA 00 81 04 01 06 D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 0A E1 08 E2 06 E3 04 E6 02 C3 00

Table 8.4-69 - Get Response Example

Example (Hex)
AA 00 81 04 82 06 D1 01 82 04 00 00 00 00 84 82 00 62 D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 52 E1 50 E2 4E E3 4C E6 4A C3 48 56 43 50 53 20 32 2E 32 78 20 5B 30 30 30 30 30 30 30 30 5D 2D 3E 20 76 31 2E 35 2E 36 20 5B 46 31 43 42 42 38 41 32 33 45 30 30 39 38 34 45 39 45 37 35 33 45 46 34 38 38 34 43 33 33 45 41 33 36 38 45 35 37 30 43 5D

8.4.3.16 Property 2.1.2.3.6.5 Entry Point Checksum (Contactless Only)(Common Kernel Only)

Table 8.4-70 - Property 2.1.2.3.6.5 Entry Point Checksum (Contactless Only)(Common Kernel Only)(Common Kernel Only)

Property OID	2.1.2.3.6.5/ 0x020102030605
Name	Entry Point Checksum
Description	The device uses this property to report the Entry Point Checksum as string, no padding. Example: 1B3725A7DBC220805DF369E035E935EF404C4B71
Securing Key	None
Min. Len (b)	Variable
Max. Len (b)	75
Data Type	Alphanumeric
Valid Values	Any string

Default	

Table 8.4-71 - Get Request Example

Example (Hex)
AA 00 81 04 01 06 D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 0A E1 08 E2 06 E3 04 E6 02 C5 00

Table 8.4-72 - Get Response Example

Example (Hex)
AA 00 81 04 82 06 D1 01 82 04 00 00 00 00 84 82 00 42 D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 32 E1 30 E2 2E E3 2C E6 2A C5 28 31 42 33 37 32 35 41 37 44 42 43 32 32 30 38 30 35 44 46 33 36 39 45 30 33 35 45 39 33 35 45 46 34 30 34 43 34 42 37 31

8.4.3.17 Property 2.1.2.3.7.1 Discover D-PAS Kernel ID (Contactless Only)

Table 8.4-73 - Property 2.1.2.3.7.1 Discover D-PAS Kernel ID (Contactless Only)

Property Description	
Property OID	2.1.2.3.7.1 / 0x020102030701
Name	Discover D-PAS Kernel ID
Description	The device uses this property to report the Discover D-PAS Kernel ID as string, padded with null characters. Example: DPAS 1.0
Securing Key	None
Min. Len (b)	9
Max. Len (b)	9
Data Type	Alphanumeric
Valid Values	Any string
Default	

Table 8.4-74 - Get Request Example

Example (Hex)
AA 00 81 04 01 07 D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 0A E1 08 E2 06 E3 04 E7 02 C1 00

Table 8.4-75 - Get Response Example

Example (Hex)
AA 00 81 04 82 07 D1 01 82 04 00 00 00 00 84 82 00 23 D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 13 E1 11 E2 0F E3 0D E7 0B C1 09 44 50 41 53 20 31 2E 30 00

8.4.3.18 Property 2.1.2.3.7.2 Discover D-PAS Kernel Firmware Part Number (Contactless Only)

Table 8.4-76 – Property 2.1.2.3.7.2 Discover D-PAS Kernel Firmware Part Number (Contactless Only)

Property Description	
Property OID	2.1.2.3.7.2 / 0x020102030702
Name	Discover D-PAS Firmware Part Number
Description	The device uses this property to report the Discover D-PAS Kernel Part Number as string, padded with null characters. Example: 1000007181 Ver A0
Securing Key	None
Min. Len (b)	18
Max. Len (b)	18
Data Type	Alphanumeric
Valid Values	Any string
Default	

Table 8.4-77 - Get Request Example

Example (Hex)
AA 00 81 04 01 09 D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 0A E1 08 E2 06 E3 04 E7 02 C2 00

Table 8.4-78 - Get Response Example

Example (Hex)
AA 00 81 04 82 09 D1 01 82 04 00 00 00 00 84 82 00 2C D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 1C E1 1A E2 18 E3 16 E7 14 C2 12 31 30 30 30 30 30 37 31 38 31 20 56 65 72 20 41 30 00

8.4.3.19 Property 2.1.2.3.7.3 Discover D-PAS Kernel Checksum (Contactless Only)

Table 8.4-79 - Property 2.1.2.3.7.3 Discover D-PAS Kernel Checksum (Contactless Only)

Property Description	
Property OID	2.1.2.3.7.3/ 0x020102030703
Name	Discover D-PAS Kernel Checksum
Description	The device uses this property to report the Discover D-PAS Kernel Checksum as string, no padding. Example: DPAS 1.0 + TAS 00x -> v1.3.42 [e28bf053de947cf6ad456a2a7c71059a2c5ac61e]
Securing Key	None
Min. Len (b)	Variable

Property Description	
Max. Len (b)	75
Data Type	Alphanumeric
Valid Values	Any string
Default	

Table 8.4-80 - Get Request Example

Example (Hex)
AA 00 81 04 01 0C D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 0A E1 08 E2 06 E3 04 E7 02 C3 00

Table 8.4-81 - Get Response Example

Example (Hex)
AA 00 81 04 82 0C D1 01 82 04 00 00 00 00 84 82 00 62 D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 52 E1 50 E2 4E E3 4C E7 4A C3 48 44 50 41 53 20 31 2E 30 20 2B 20 54 41 53 20 30 30 78 20 2D 3E 20 76 31 2E 33 2E 34 32 20 5B 65 32 38 62 66 30 35 33 64 65 39 34 37 63 66 36 61 64 34 35 36 61 32 61 37 63 37 31 30 35 39 61 32 63 35 61 63 36 31 65 5D

8.4.3.20 Property 2.1.2.3.8.1 American Express Expresspay Kernel ID (Contactless Only)

Table 8.4-82 - Property 2.1.2.3.8.1 American Express Expresspay Kernel ID (Contactless Only)

Property Description	
Property OID	2.1.2.3.8.1 / 0x020102030801
Name	American Express Expresspay Kernel ID
Description	The device uses this property to report the American Express Expresspay Kernel ID as string, padded with null characters. Example: AMEX 4.0.2
Securing Key	None
Min. Len (b)	11
Max. Len (b)	11
Data Type	Alphanumeric
Valid Values	Any string
Default	

Table 8.4-83 - Get Request Example

Example (Hex)
AA 00 81 04 01 0E D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 0A E1 08 E2 06 E3 04 E8 02 C1 00

Table 8.4-84 - Get Response Example

Example (Hex)
AA 00 81 04 82 0E D1 01 82 04 00 00 00 00 84 82 00 25 D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 15 E1 13 E2 11 E3 0F E8 0D C1 0B 41 4D 45 58 20 34 2E 30 2E 32 00

8.4.3.21 Property 2.1.2.3.8.2 American Express Expresspay Kernel Firmware Part Number (Contactless Only)

Table 8.4-85 – Property 2.1.2.3.8.2 American Express Expresspay Kernel Firmware Part Number (Contactless Only)

Property Description	
Property OID	2.1.2.3.8.2 / 0x020102030802
Name	American Express Expresspay Firmware Part Number
Description	The device uses this property to report the American Express Expresspay Part Number as string, padded with null characters. Example: 1000007181 Ver A0
Securing Key	None
Min. Len (b)	18
Max. Len (b)	18
Data Type	Alphanumeric
Valid Values	Any string
Default	

Table 8.4-86 - Get Request Example

Example (Hex)
AA 00 81 04 01 0F D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 0A E1 08 E2 06 E3 04 E8 02 C2 00

Table 8.4-87 - Get Response Example

Example (Hex)
AA 00 81 04 82 0F D1 01 82 04 00 00 00 00 84 82 00 2C D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 1C E1 1A E2 18 E3 16 E8 14 C2 12 31 30 30 30 30 30 37 31 38 32 20 56 65 72 20 41 30 00

8.4.3.22 Property 2.1.2.3.8.3 American Express Expresspay Kernel Checksum (Contactless Only)

Table 8.4-88 - Property 2.1.2.3.8.3 American Express Expresspay Kernel Checksum (Contactless Only)

Property Description	
Property OID	2.1.2.3.8.3/ 0x020102030803
Name	American Express Expresspay Kernel Checksum

Property Description	
Description	The device uses this property to report the American Express Expresspay Kernel Checksum as string, no padding. Example: C4.2.7 -> v1.0.6 [5D5CC3073F64FE7F14F5454D62026EDB9E202930]
Securing Key	None
Min. Len (b)	Variable
Max. Len (b)	75
Data Type	Alphanumeric
Valid Values	Any string
Default	

Table 8.4-89 - Get Request Example

Example (Hex)
AA 00 81 04 01 11 D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 0A E1 08 E2 06 E3 04 E8 02 C3 00

Table 8.4-90 - Get Response Example

Example (Hex)
AA 00 81 04 82 11 D1 01 82 04 00 00 00 00 84 82 00 55 D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 45 E1 43 E2 41 E3 3F E8 3D C3 3B 43 34 2E 32 2E 37 20 2D 3E 20 76 31 2E 30 2E 36 20 5B 35 44 35 43 43 33 30 37 33 46 36 34 46 45 37 46 31 34 46 35 34 35 34 44 36 32 30 32 36 45 44 42 39 45 32 30 32 39 33 30 5D

8.4.3.23 Property 2.1.2.3.9.1 Apple VAS Kernel ID

Table 8.4-91 - Property 2.1.2.3.9.1 Apple VAS Kernel ID

Property Description	
Property OID	2.1.2.3.9.1 / 0x020102030901
Name	Apple VAS Kernel ID
Description	The device uses this property to report the Apple VAS Kernel ID as a string, padded with null characters. Example: APPLE VAS 1.0.0
Securing Key	None
Min. Len (b)	16
Max. Len (b)	16
Data Type	Alphanumeric
Valid Values	Any string
Default	

Table 8.4-92 - Get Request Example

Example (Hex)
AA 00 81 04 01 0E D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 0A E1 08 E2 06 E3 04 E9 02 C1 00

Table 8.4-93 - Get Response Example

Example (Hex)
AA008104820ED10182040000000008482002AD10181072B06010401F609850102891AE118E216E314E912C1104150504C452056415320312E302E3000

8.4.3.24 Property 2.1.2.3.A.1 JCB Kernel ID (Contactless Only) (Common Kernel Only)

Table 8.4-94 - Property 2.1.2.3.A.1 JCB Kernel ID (Contactless Only) (Common Kernel Only)

Property Description	
Property OID	2.1.2.3.A.1 / 0x020102030A01
Name	JCB Kernel ID
Description	The device uses this property to report the JCB Kernel ID as string, padded with null characters. Example: JCB 1.6
Securing Key	None
Min. Len (b)	11
Max. Len (b)	11
Data Type	Alphanumeric
Valid Values	Any string
Default	

Table 8.4-95 - Get Request Example

Example (Hex)
AA 00 81 04 01 0E D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 0A E1 08 E2 06 E3 04 EA 02 C1 00

Table 8.4-96 - Get Response Example

Example (Hex)
AA 00 81 04 82 06 D1 01 82 04 00 00 00 00 84 82 00 22 D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 12 E1 10 E2 0E E3 0C EA 0A C1 08 4A 43 42 20 31 2E 36 00

8.4.3.25 Property 2.1.2.3.A.2 JCB Kernel Firmware Part Number (Contactless Only) (Common Kernel Only)

Table 8.4-97 – Property 2.1.2.3.A.2 JCB Kernel Firmware Part Number (Contactless Only) (Common Kernel Only)

Property Description	
Property OID	2.1.2.3.A.2 / 0x020102030A02
Name	JCB Firmware Part Number
Description	The device uses this property to report the JCB Part Number as string, padded with null characters. Example: 1000009650 Ver A0
Securing Key	None
Min. Len (b)	18
Max. Len (b)	18
Data Type	Alphanumeric
Valid Values	Any string
Default	

Table 8.4-98 - Get Request Example

Example (Hex)
AA 00 81 04 01 0F D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 0A E1 08 E2 06 E3 04 EA 02 C2 00

Table 8.4-99 - Get Response Example

Example (Hex)
AA 00 81 04 82 06 D1 01 82 04 00 00 00 00 84 82 00 2D D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 1D E1 1B E2 19 E3 17 EA 15 C2 13 31 30 30 30 30 30 39 36 35 30 20 56 65 72 20 41 41 30 00

8.4.3.26 Property 2.1.2.3.A.3 JCB Kernel Checksum (Contactless Only) (Common Kernel Only)

Table 8.4-100 - Property 2.1.2.3.A.3 JCB Kernel Checksum (Contactless Only) (Common Kernel Only)

Property Description	
Property OID	2.1.2.3.A.3/ 0x020102030A03
Name	JCB Kernel Checksum
Description	The device uses this property to report the JCB Kernel Checksum as string, no padding. Example: JCB 1.6 -> v1.0.29 [ca44a90cd42d379088fcb8ca8d1fab195546b278]
Securing Key	None
Min. Len (b)	Variable

Property Description	
Max. Len (b)	75
Data Type	Alphanumeric
Valid Values	Any string
Default	

Table 8.4-101 - Get Request Example

Example (Hex)
AA 00 81 04 01 11 D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 0A E1 08 E2 06 E3 04 EA 02 C3 00

Table 8.4-102 - Get Response Example

Example (Hex)
AA 00 81 04 82 06 D1 01 82 04 00 00 00 00 84 82 00 57 D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 47 E1 45 E2 43 E3 41 EA 3F C3 3D 4A 43 42 20 31 2E 36 20 2D 3E 20 76 31 2E 30 2E 32 39 20 5B 63 61 34 34 61 39 30 63 64 34 32 64 33 37 39 30 38 38 66 63 62 38 63 61 38 64 31 66 61 62 31 39 35 35 34 36 62 32 37 38 5D

8.4.3.27 Property 2.1.2.3.A.4 Reader Core Checksum (Contactless Only) (Common Kernel Only)

Table 8.4-103 - Property 2.1.2.3.A.4 Reader Core Checksum (Contactless Only) (Common Kernel Only)

Property Description	
Property OID	2.1.2.3.A.4/ 0x020102030A04
Name	Reader Core Checksum
Description	The device uses this property to report the Reader Core Checksum as string, no padding. Example: 371835cedcd7f8a3e4cf8b32cc03803dfdc1f507
Securing Key	None
Min. Len (b)	Variable
Max. Len (b)	75
Data Type	Alphanumeric
Valid Values	Any string
Default	

Table 8.4-104 - Get Request Example

Example (Hex)
AA 00 81 04 01 11 D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 0A E1 08 E2 06 E3 04 EA 02 C4 00

Table 8.4-105 - Get Response Example

Example (Hex)
AA 00 81 04 82 06 D1 01 82 04 00 00 00 00 84 82 00 42 D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 32 E1 30 E2 2E E3 2C EA 2A C4 28 33 37 31 38 33 35 63 65 64 63 64 37 66 38 61 33 65 34 63 66 38 62 33 32 63 63 30 33 38 30 33 64 66 64 63 31 66 35 30 37

8.4.3.28 Property 2.1.2.3.A.5 Entry Point Checksum (Contactless Only) (Common Kernel Only)

Table 8.4-106 - Property 2.1.2.3.A.5 Entry Point Checksum (Contactless Only) (Common Kernel Only) (Common Kernel Only)

Property Description	
Property OID	2.1.2.3.A.5/ 0x020102030A05
Name	Entry Point Checksum
Description	The device uses this property to report the Entry Point Checksum as string, no padding. Example: 1B3725A7DBC220805DF369E035E935EF404C4B71
Securing Key	None
Min. Len (b)	Variable
Max. Len (b)	75
Data Type	Alphanumeric
Valid Values	Any string
Default	

Table 8.4-107 - Get Request Example

Example (Hex)
AA 00 81 04 01 06 D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 0A E1 08 E2 06 E3 04 EA 02 C5 00

Table 8.4-108 - Get Response Example

Example (Hex)
AA 00 81 04 82 06 D1 01 82 04 00 00 00 00 84 82 00 42 D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 32 E1 30 E2 2E E3 2C EA 2A C5 28 31 42 33 37 32 35 41 37 44 42 43 32 32 30 38 30 35 44 46 33 36 39 45 30 33 35 45 39 33 35 45 46 34 30 34 43 34 42 37 31

8.4.3.29 Property 2.1.2.3.B.1 China Union Pay Kernel ID (Contactless Only) (Common Kernel Only)

Table 8.4-109 - Property 2.1.2.3.B.1 China Union Pay Kernel ID (Contactless Only) (Common Kernel Only)

Property Description	
Property OID	2.1.2.3.B.1 / 0x020102030B01

Property Description	
Name	China Union Pay Kernel ID
Description	The device uses this property to report the China Union Pay Kernel ID as string, padded with null characters. Example: CUP 1.0.2
Securing Key	None
Min. Len (b)	11
Max. Len (b)	11
Data Type	Alphanumeric
Valid Values	Any string
Default	

Table 8.4-110 - Get Request Example

Example (Hex)
AA 00 81 04 01 0E D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 0A E1 08 E2 06 E3 04 EB 02 C1 00

Table 8.4-111 - Get Response Example

Example (Hex)
AA 00 81 04 82 06 D1 01 82 04 00 00 00 00 84 82 00 24 D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 14 E1 12 E2 10 E3 0E EB 0C C1 0A 43 55 50 20 31 2E 30 2E 32 00

8.4.3.30 Property 2.1.2.3.B.2 China Union Pay Kernel Firmware Part Number (Contactless Only) (Common Kernel Only)

Table 8.4-112 – Property 2.1.2.3.B.2 China Union Pay Kernel Firmware Part Number (Contactless Only) (Common Kernel Only)

Property Description	
Property OID	2.1.2.3.B.2 / 0x020102030B02
Name	China Union Pay Firmware Part Number
Description	The device uses this property to report the China Union Pay Part Number as string, padded with null characters. Example: 1000009651 Ver AA0
Securing Key	None
Min. Len (b)	18
Max. Len (b)	18
Data Type	Alphanumeric
Valid Values	Any string

Property Description	
Default	

Table 8.4-113 - Get Request Example

Example (Hex)
AA 00 81 04 01 0F D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 0A E1 08 E2 06 E3 04 EB 02 C2 00

Table 8.4-114 - Get Response Example

Example (Hex)
AA 00 81 04 82 06 D1 01 82 04 00 00 00 00 84 82 00 2D D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 1D E1 1B E2 19 E3 17 EB 15 C2 13 31 30 30 30 30 30 39 36 35 31 20 56 65 72 20 41 41 30 00

8.4.3.31 Property 2.1.2.3.B.3 China Union Pay Kernel Checksum (Contactless Only) (Common Kernel Only)

Table 8.4-115 - Property 2.1.2.3.B.3 China Union Pay Kernel Checksum (Contactless Only) (Common Kernel Only)

Property Description	
Property OID	2.1.2.3.B.3/ 0x020102030B03
Name	China Union Pay Kernel Checksum
Description	The device uses this property to report the China Union Pay Kernel Checksum as string, no padding. Example: CUP v1.0.2 -> v1.3.40 [23e9bc82b4ecbf422c2054c91914848017e0ed0f]
Securing Key	None
Min. Len (b)	Variable
Max. Len (b)	75
Data Type	Alphanumeric
Valid Values	Any string
Default	

Table 8.4-116 - Get Request Example

Example (Hex)
AA 00 81 04 01 11 D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 0A E1 08 E2 06 E3 04 EB 02 C3 00

Table 8.4-117 - Get Response Example

Example (Hex)
AA 00 81 04 82 06 D1 01 82 04 00 00 00 00 84 82 00 5A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 4A E1 48 E2 46 E3 44 EB 42 C3 40 43 55 50 20 76 31 2E 30 2E 32 20 2D 3E 20 76 31 2E 33 2E 34 30 20 5B 32 33 65 39 62 63 38 32 62 34 65 63 62 66 34 32 32 63 32 30 35 34 63 39 31 39 31 34 38 34 38 30 31 37 65 30 65 64 30 66 5D

8.4.3.32 Property 2.1.2.3.C.1 Interact Flash Kernel ID (Contactless Only) (Common Kernel Only)

Table 8.4-118 - Property 2.1.2.3.C.1 Interact Flash Kernel ID (Contactless Only) (Common Kernel Only)

Property Description	
Property OID	2.1.2.3.C.1 / 0x020102030C01
Name	Interact Flash Kernel ID
Description	The device uses this property to report the Interact Flash Kernel ID as string, padded with null characters. Example: FLASH 1.9
Securing Key	None
Min. Len (b)	11
Max. Len (b)	11
Data Type	Alphanumeric
Valid Values	Any string
Default	

Table 8.4-119 - Get Request Example

Example (Hex)
AA 00 81 04 01 0E D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 0A E1 08 E2 06 E3 04 EC 02 C1 00

Table 8.4-120 - Get Response Example

Example (Hex)
AA 00 81 04 82 06 D1 01 82 04 00 00 00 00 84 82 00 24 D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 14 E1 12 E2 10 E3 0E EC 0C C1 0A 46 4C 41 53 48 20 31 2E 39 00

8.4.3.33 Property 2.1.2.3.C.2 Interact Flash Kernel Firmware Part Number (Contactless Only) (Common Kernel Only)

Table 8.4-121 – Property 2.1.2.3.C.2 Interact Flash Kernel Firmware Part Number (Contactless Only) (Common Kernel Only)

Property Description	
Property OID	2.1.2.3.C.2 / 0x020102030C02

Property Description	
Name	Interact Flash Firmware Part Number
Description	The device uses this property to report the Interact Flash Part Number as string, padded with null characters. Example: 1000009652 Ver AA0
Securing Key	None
Min. Len (b)	18
Max. Len (b)	18
Data Type	Alphanumeric
Valid Values	Any string
Default	

Table 8.4-122 - Get Request Example

Example (Hex)
AA 00 81 04 01 0F D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 0A E1 08 E2 06 E3 04 EC 02 C2 00

Table 8.4-123 - Get Response Example

Example (Hex)
AA 00 81 04 82 06 D1 01 82 04 00 00 00 00 84 82 00 2D D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 1D E1 1B E2 19 E3 17 EC 15 C2 13 31 30 30 30 30 30 39 36 35 32 20 56 65 72 20 41 41 30 00

8.4.3.34 Property 2.1.2.3.C.3 Interact Flash Kernel Checksum (Contactless Only) (Common Kernel Only)

Table 8.4-124 - Property 2.1.2.3.C.3 Interact Flash Kernel Checksum (Contactless Only) (Common Kernel Only)

Property Description	
Property OID	2.1.2.3.C.3/ 0x020102030C03
Name	Interact Flash Kernel Checksum
Description	The device uses this property to report the Interact Flash Kernel Checksum as string, no padding. Example: Flash 1.9 -> v1.3.41 [394e45aed865e276e4f6737de26aa84c6eb1b174]
Securing Key	None
Min. Len (b)	Variable
Max. Len (b)	75
Data Type	Alphanumeric
Valid Values	Any string

Property Description	
Default	

Table 8.4-125 - Get Request Example

Example (Hex)
AA 00 81 04 01 11 D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 0A E1 08 E2 06 E3 04 EC 02 C3 00

Table 8.4-126 - Get Response Example

Example (Hex)
AA 00 81 04 82 06 D1 01 82 04 00 00 00 00 84 82 00 59 D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 49 E1 47 E2 45 E3 43 EC 41 C3 3F 46 6C 61 73 68 20 31 2E 39 20 2D 3E 20 76 31 2E 33 2E 34 31 20 5B 33 39 34 65 34 35 61 65 64 38 36 35 65 32 37 36 65 34 66 36 37 33 37 64 65 32 36 61 61 38 34 63 36 65 62 31 62 31 37 34 5D

8.4.4 Property Subgroup 2.1.2.8.nn Custom UI

8.4.4.1 Property 2.1.2.8.1.1 UI Configuration File

Table 8.4-127 - Property 2.1.2.8.1.1 UI Configuration File

Property Description	
Property OID	2.1.2.8.1.1 / 0x020102080101
Name	UI Configuration Filename
Description	This string contains the part number and the revision of the UI Configuration File.
Securing Key	None
Min. Len (b)	14
Max. Len (b)	14
Data Type	ASCII
Valid Values	CFG000xxxx-xxx
Default	None

Table 8.4-128 - Get Request Example

Example (Hex)
AA0081040106D101841AD10181072B06010401F609850102890AE108E206E804E102C100

Table 8.4-129 - Get Response Example

Example (Hex)
AA0081048206D10182040000000084820028D10181072B06010401F609850102 8918E116E214E812E110C10E4346473030303631322D323030

8.5 Property Group 2.2.1.nnn Device Hardware Information

8.5.1 Property Subgroup 2.2.1.1.1.n Device Common Configuration Information

8.5.1.1 Property 2.2.1.1.1.1 Serial Number

Table 8.5-1 - Property 2.2.1.1.1.1 Serial Number

Property Description	
Property OID	2.2.1.1.1.1 / 0x020201010101
Name	Device Serial Number
Description	The device uses this property to report its serial number. The left 3.5 bytes represent the 7 digit serial number, and the remaining half byte is always 0. Example: Serial number **B603226** is reported as 0xB6, 0x03, 0x22, 0x60.
Securing Key	None
Min. Len (b)	4
Max. Len (b)	4
Data Type	Binary
Valid Values	Min 0x00000000 Max 0xFFFFFFF0
Default	

Table 8.5-2 - Get Request Example

Example (Hex)
AA 00 81 04 01 B5 D1 01 84 18 D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 87 04 02 01 01 01 89 02 C1 00

Table 8.5-3 - Get Response Example

Example (Hex)
AA 00 81 04 82 B5 D1 01 82 04 00 00 00 00 84 82 00 1C D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 87 04 02 01 01 01 89 06 C1 04 B6 13 78 A0

8.5.1.2 Property 2.2.1.1.1.3 PCI Hardware ID

Table 8.5-4 - Property 2.2.1.1.1.3 PCI Hardware ID

Property Description	
Property OID	2.2.1.1.1.3 / 0x020201010103
Name	PCI Hardware ID
Description	The device uses this property to report its PCI Hardware ID. Customers can use this value to compare against the device's certification records on the PCI web site.
Securing Key	None

Property Description	
Min. Len (b)	10
Max. Len (b)	256
Data Type	Alphanumeric
Valid Values	Any string
Default	

Table 8.5-5 - Get Request Example

Example (Hex)
AA 00 81 04 01 E3 D1 01 84 1A D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 0A E2 08 E1 06 E1 04 E1 02 C3 00

Table 8.5-6 - Get Response Example

Example (Hex)
AA 00 81 04 82 E3 D1 01 82 04 00 00 00 00 84 82 00 24 D1 01 81 07 2B 06 01 04 01 F6 09 85 01 02 89 14 E2 12 E1 10 E1 0E E1 0C C3 0A 33 36 50 43 49 34 35 30 41 30

8.6 Property Group 2.3.1.nnn System Status Information

8.6.1 Property Subgroup 2.3.1.1.1.n System Status Injected Key Information

8.6.1.1 Property 2.3.1.1.1.1 Device Key Status

Table 8.6-1 - Property 2.3.1.1.1.1 Device Key Status

Property Description	
Property OID	2.3.1.1.1.1 / 0x020301010101
Name	Device Key Status
Description	This OID contains a 32-bit bitmap. Each bit indicates the status of a device key. A bit value of 1 indicates the corresponding key has been injected.
Securing Key	None
Min. Len (b)	4
Max. Len (b)	4
Data Type	Binary

Property Description	
Valid Values	Bit 0: TMPTK_1000 Bit 1: MTK_1001 Bit 2: DEVTK_1002 Bit 3: FINTK_1003 Bit 4: PRODTK_1021 Bit 5: MFGTK_1022 Bit 6: MFIFTK_1081 Bit 7: AKIFTK_1091 Bit 8: FREQMK_1101 Bit 9: MREQMK_1102 Bit 10: MFRQMK_1111 Bit 11: ARQ1MK_1121 Bit 12: ARQ2MK_1122 Bit 13 – 31: Reserved
Default	None

Table 8.6-2 - Get Request Example

Example (Hex)
AA00 8104010ED101 841AD101 81072B06010401F609 850102 890AE308E106E104E102C100

Table 8.6-3 - Get Response Example

Example (Hex)
AA00 8104820ED101 820400000000 8482001E D101 81072B06010401F609 850102 890EE30CE10AE108E106C1040000077E

8.6.1.2 Property 2.3.1.1.1.2 Transaction Key Status

Table 8.6-4 - Property 2.3.1.1.1.2 Transaction Key Status

Property Description	
Property OID	2.3.1.1.1.2 / 0x020301010102
Name	Transaction Key Status
Description	This OID contains a 32-bit bitmap. Each bit indicates the status of a transaction key. A bit value of 1 indicates the corresponding key has been injected.
Securing Key	None
Min. Len (b)	4
Max. Len (b)	4
Data Type	Binary

Property Description	
Valid Values	Bit 0: DKPTM0_2000 Bit 1: DKPTM1_2001 Bit 2: DKPTM2_2002 Bit 3: DKPTM3_2003 Bit 4: DKPTM4_2004 Bit 5: DKPTM5_2005 Bit 6: DKPTM6_2006 Bit 7: DKPTM7_2007 Bit 8: DKPTM8_2008 Bit 9: DKPTM9_2009 Bit 10: DKPTMA_200A Bit 11: DKPTMB_200B Bit 12: DKPTMC_200C Bit 13: DKPTMD_200D Bit 14: DKPTME_200E Bit 15: DKPTMF_200F Bit 16: DKPTM10_2010 Bit 17: DKPTM11_2011 Bit 18: DKPTM12_2012 Bit 19: DKPTM13_2013 Bit 20: DKPTM14_2014 Bit 21: DKPTM15_2015 Bit 22: DKPTM16_2016 Bit 23: DKPTM17_2017 Bit 24: DKPTM18_2018 Bit 25: DKPTM19_2019 Bit 26: DKPTM1A_201A Bit 27: DKPTM1B_201B Bit 28: DKPTM1C_201C Bit 29: DKPTM1D_201D Bit 30: DKPTM1E_201E Bit 31: DKPTM1F_201F
Default	None

Table 8.6-5 - Get Request Example

Example (Hex)
AA00 8104010FD101 841AD101 81072B06010401F609 850102 890AE308E106E104E102C200

Table 8.6-6 - Get Response Example

Example (Hex)
AA00 8104820FD101 820400000000 8482001ED101 81072B06010401F609 850102 890EE30CE10AE108E106C20400000081

8.6.2 Property Subgroup 2.3.1.1.2.n System Status Security Protection Information

8.6.2.1 Property 2.3.1.1.2.1 Real Time Clock Enabled

Table 8.6-7 - Property 2.3.1.1.2.1 Real Time Clock Enabled

Property Description	
Property OID	2.3.1.1.2.1 / 0x020301010201
Name	Real Time Clock Enabled
Description	The device maps this property to its internal register that enables its internal real-time clock. Because the device ensures that register is enabled every time it powers up, this property should always report Real Time Clock Enabled.
Securing Key	None
Min. Len (b)	1
Max. Len (b)	1
Data Type	Binary

Property Description	
Valid Values	0x00 = Real Time Clock Not Enabled 0x01 = Real Time Clock Enabled
Default	None

Table 8.6-8 - Get Request Example

Example (Hex)
AA00 8104 0155D101 840F D101 8501 02 8704 03010102 8902 C100

Table 8.6-9 - Get Response Example

Example (Hex)
AA00 81048255D101 820400000000 84820010 D101 850102 870403010102 8903 C101 01

8.6.2.2 Property 2.3.1.1.2.2 Tamper Sensors Activated

Table 8.6-10 - Property 2.3.1.1.2.2 Tamper Sensors Activated

Property Description	
Property OID	2.3.1.1.2.2 / 0x020301010202
Name	Tamper Sensors Activated
Description	A device that is operating normally should always have its tamper sensors activated.
Securing Key	None
Min. Len (b)	1
Max. Len (b)	1
Data Type	Binary
Valid Values	0x00 = Tamper Sensors Not Activated 0x01 = Tamper Sensors Activated
Default	None

Table 8.6-11 - Get Request Example

Example (Hex)
AA00 8104 0155D101 840F D101 8501 02 8704 03010102 8902 C100

Table 8.6-12 - Get Response Example

Example (Hex)
AA00 81048255D101 820400000000 84820010 D101 850102 870403010102 8903 C101 01

8.6.2.3 Property 2.3.1.1.2.3 Tamper Sensor Tampered

Table 8.6-13 - Property 2.3.1.1.2.3 Tamper Sensor Tampered

Property Description	
Property OID	2.3.1.1.2.3 / 0x020301010203
Name	Tamper Sensor Tampered
Description	The device uses this property to report whether a tamper sensor has tampered. A device that is operating normally should not reported that it has been tampered with.
Securing Key	None
Min. Len (b)	1
Max. Len (b)	1
Data Type	Binary
Valid Values	0x00 = Not Tampered 0x01 = Tampered
Default	None

Table 8.6-14 - Get Request Example

Example (Hex)
AA00 8104 0155D101 840F D101 8501 02 8704 03010102 8902 C300

Table 8.6-15 - Get Response Example

Example (Hex)
AA00 81048255D101 820400000000 84820010 D101 850102 870403010102 8903 C301 00

8.6.3 Property Subgroup 2.3.1.2.1.n System Status Device State Information

8.6.3.1 Property 2.3.1.2.1.1 Device Operational Status

Table 8.6-16 - Property 2.3.1.2.1.1 Device Operational Status

Property Description	
Property OID	2.3.1.2.1.1 / 0x020301020101
Name	Device Operational Status

Property Description	
Description	The device uses this property to report its operational status. **Online** is the status the manufacturer populates in this property before shipping, and is the result of the device going through processes that configure its security subsystems and features. In this state, the device is fully functional and can perform transactions. **Offline** means the device is no longer fully functional and can no longer perform transactions. The device automatically transitions to this state if it detects a problem with security or any of the subsystems it checks. For example, trying to open the device triggers a tamper response, which causes the device to change its operational status to Offline. To retrieve more information about the cause of an Offline status, the host can retrieve **Property 2.3.1.2.1.2 Offline Status Detail**.
Securing Key	None
Min. Len (b)	1
Max. Len (b)	1
Data Type	Binary
Valid Values	0x01 = Offline 0x02 = Online
Default	None

Table 8.6-17 - Get Request Example

Example (Hex)
AA00 8104 0155D101 840F D101 8501 02 8704 03010201 8902 C100

Table 8.6-18 - Get Response Example

Example (Hex)
AA00 81048255D101 820400000000 84820010 D101 850102 870403010201 8903 C101 02

8.6.3.2 Property 2.3.1.2.1.2 Offline Status Detail

Table 8.6-19 - Property 2.3.1.2.1.2 Offline Status Detail

Property Description	
Property OID	2.3.1.2.1.2 / 0x020301020102
Name	Offline Status Detail

Property Description	
Description	The device uses this property to report the status of every subsystem that can cause the device's operation status to be set to **Offline** in **Property 2.3.1.2.1.1 Device Operational** Status. The property consists of a sequence of bytes where each bit in each byte represents the status of a subsystem. If a bit is set to one, then there is a problem with the subsystem, otherwise no problem was detected. Some subsystems provide other properties or commands that can be used to get more information about the subsystem's status. For example, **Property 2.3.1.1.2.2 Tamper Sensors Activated** and **Property 2.3.1.1.2.3 Tamper Sensor Tampered**.
Securing Key	None
Min. Len (b)	1
Max. Len (b)	16 (reserved for future expansion)
Data Type	Binary
Valid Values	Byte 0 • Bit 0 = Tamper problem present • Bit 1 = Master Key problem present • Bit 2 = Keys and Certificates problem present • Bit 3 = Real Time Clock problem present • Bit 4 = Random Number Generator problem present • Bit 5 = Cryptography Engine problem present • Bit 6 = Magnetic Stripe Reader Hardware problem present • Bit 7 = Reserved
Default	None

Table 8.6-20 - Get Request Example

Example (Hex)
AA00 8104 0155D101 840F D101 8501 02 8704 03010201 8902 C200

Table 8.6-21 - Get Response Example

Example (Hex)
AA00 81048255D101 820400000000 84820010 D101 850102 870403010201 8903 C201 00

8.6.3.3 Property 2.3.1.2.1.3 External Power Supplied

Table 8.6-22 - Property 2.3.1.2.1.3 External Power Supplied

Property Description	
Property OID	2.3.1.2.1.3 / 0x020301020103
Name	External Power Supplied
Description	The device uses this property to report the status of external power.

Property Description	
Securing Key	None
Min. Len (b)	1
Max. Len (b)	1
Data Type	Binary
Valid Values	0x00 = No external power supplied 0x01 = Power supplied by USB port
Default	None

Table 8.6-23 - Get Request Example

Example (Hex)
AA00 8104 0155D101 840F D101 8501 02 8704 03010201 8902 C300

Table 8-200 - Get Response Example

Example (Hex)
AA00 81048255D101 820400000000 84820010 D101 850102 870403010201 8903 C301 01

8.6.3.4 Property 2.3.1.2.1.4 Battery State of Charge

Table 8-201 - Property 2.3.1.2.1.4 Battery State of Charge

Property Description	
Property OID	2.3.1.2.1.4 / 0x020301020104
Name	Battery State of Charge
Description	The device uses this property to report the charge status of the internal battery. The charge status is reported as a percentage of full charge.
Securing Key	None
Min. Len (b)	1
Max. Len (b)	1
Data Type	Binary
Valid Values	0x00..0x64
Default	None

Table 8-202 - Get Request Example

Example (Hex)
AA00 8104 0155D101 840F D101 8501 02 8704 03010201 8902 C400

Table 8-203 - Get Response Example

Example (Hex)
AA00 81048255D101 820400000000 84820010 D101 850102 870403010201 8903 C401 64

8.6.3.5 Property 2.3.1.2.1.5 Battery Charger Status

Table 8-204 - Property 2.3.1.2.1.5 Battery Charger Status

Property Description	
Property OID	2.3.1.2.1.5 / 0x020301020105
Name	Battery Charger Status
Description	The device uses this property to report the status of the battery charger.
Securing Key	None
Min. Len (b)	1
Max. Len (b)	1
Data Type	Binary
Valid Values	0x00 = Precharge 0x01 = Fast charge – constant current 0x02 = Fast charge – constant voltage 0x03 = End of charge 0x04 = Charge complete 0xFF = No external power supplied
Default	None

Table 8-205 - Get Request Example

Example (Hex)
AA00 8104 0155D101 840F D101 8501 02 8704 03010201 8902 C500

Table 8-206 - Get Response Example

Example (Hex)
AA00 81048255D101 820400000000 84820010 D101 850102 870403010201 8903 C501 04

8.6.3.6 Property 2.3.1.2.1.6 Device Temperature

Table 8-207 - Property 2.3.1.2.1.6 Device Temperature

Property Description	
Property OID	2.3.1.2.1.6 / 0x020301020106
Name	Device Temperature
Description	The device uses this property to report the temperature in Celsius
Securing Key	None
Min. Len (b)	1

Property Description	
Max. Len (b)	1
Data Type	Signed Binary
Valid Values	0x80 .. 0x7F (-128 .. 127)
Default	None

Table 8-208 - Get Request Example

Example (Hex)
AA00 8104 0155D101 840F D101 8501 02 8704 03010201 8902 C600

Table 8-209 - Get Response Example

Example (Hex)
AA00 81048255D101 820400000000 84820010 D101 850102 870403010201 8903 C601 26

Appendix A Object IDs (OIDs) and ITU-T X660

A.1 About OIDs

For modular management of device functions, information, and settings, this standard makes extensive use of Object Identifiers (also known as Object IDs or OIDs) as defined in ***ITU-T X.660 | ISO/IEC 9834-1***, which can be found by searching for **X.660** in the publications on www.itu.int.

OIDs are identifiers for any generic data element, and are managed by standards bodies to be globally unique (much like web domains, IP addresses, and Media Access Control MAC addresses). They are managed by using a tree structure consisting of **nodes**, where the tree structure is defined and controlled by a hierarchy of subordinate **Registration Authorities**, each with their authority delegated by a Registration Authority one level higher in the tree, starting with the root nodes managed by ITU-T and ISO. The ***X.660*** standard for OIDs is harmonized with the data representation standard of ***ASN.1*** via that standard's `OBJECT IDENTIFIER` and `OID-IRI` types. Every node is assigned a **primary integer value** (or primary value for short) which serves to uniquely identify the node, and may be assigned secondary identifiers (such as strings) for human readability.

MagTek is the Registration Authority for the OID tree beginning at **{iso(1) identified-organization(3) dod(6) internet(1) private(4) enterprise(1) MagTek(15113)}** (using **value notation** of the ***ASN.1*** `OBJECT IDENTIFIER` type). The branch can also be represented in numerical shorthand as **1.3.6.1.4.1.15113**, which is the **encoding** form of the ***ASN.1*** `OBJECT IDENTIFIER` type.

Per ***X.660***, OIDs can be encoded in **constructed** form or **primitive** form. The primitive form is a sequence of octets (bytes) and is summarized in section **A.2 Octet Encoding of OIDs**. The constructed form is a TLV data object and is summarized in section **A.3 TLV Encoding of OIDs**.

In many cases, the MagTek implementation of OID-based message elements allows the host to specify an optional **Company ID** and **Tree prefix** alongside a **Relative OID**, where the full OID is a concatenation of the company ID, followed by the tree prefix, followed by the relative OID. If a message does not include the optional company ID or tree prefix, company ID is assumed to be the MagTek arc above, and the tree prefix is assumed to be a reasonable sequence of additional nodes based on the purpose and scope of the message.

A.2 Octet Encoding of OIDs

In addition to human-readable ***ASN.1*** value notation and encoding forms above, OIDs can be encoded as binary values (octets) as follows (per ***X.660*** section ***8.19 Encoding of an object identifier value***):

1) Start with the numerical form (for example, MagTek's arc **1.3.6.1.4.1.15113** from section **A.1**).
2) Multiply the primary value of the first node in the OID by 40 decimal (e.g., 1*40=40) and add it to the primary value of the second node. The sum is used as the first byte (e.g., 43 decimal = **0x2B**) of the octet form. The first two nodes receive this special treatment because their possible value ranges are expected to remain very small indefinitely.
3) The next bytes in the octet encoded form are simply equal to each primary value in the OID sequence, unless the primary value is greater than 128 decimal (0x80). For primary values greater than 0x80, additional encoding rules apply that are not necessary to detail here, because for simplicity MagTek selects OID primary values that are equal to 0x79 or less (and in many cases, 0x30 or less); In addition, TLV encoding places additional restrictions on selection of OID, in that if the OID is to be used directly as a tag in a TLV data object, it must not be greater than 0x1F (see section **3.2.1.1 About TLV Encoding**). The only portion of MagTek's arc that is subject to these additional encoding rules is the primary value **15113** at the end, which encodes to **0xF609** hex (per the rules not detailed here, see ***X.660*** for details). The full arc **1.3.6.1.4.1.15113** therefore encodes to **0x2B 06 01 04 01 F609**.

A.3 TLV Encoding of OIDs

The primary method of encoding OIDs, as specified in ***X.660*** section ***8 Basic encoding rules***, is BER Tag-Length-Value (TLV) format. A TLV-encoded OID is a set of nested TLV data objects, where:

- The **tag** of each TLV data object equals the primary value of the node it represents in the OID sequence, encoded according to the BER TLV tag number rules; the assigned primary values in this standard are all **private** (11nnnnnn), the bottom level value / leaf of any OID arc is encoded as **primitive** (nn0nnnnn), and the levels above it are encoded as **composed** (nn1nnnnn). This information makes it possible to use the bit definitions for **Tag** in section **3.2.1 Tag-Length-Value (TLV) Encoding** as a reference to encode each level of the OID as a tag. Generally the primary values in this standard are selected to be less than 0x1F, and so do not require multi-byte tags.
- The **length** of each TLV data object, like all TLV data objects, is equal to the length of its value.
- The **value** of each TLV data object is equal to a nested TLV data object containing all subsequent nodes in the OID sequence, so TLV-encoded OIDs must be built from the bottom up. The primitive value in the bottom node of a TLV-encoded OID is the set of actual values contained by that OID (e.g., string, byte[s], and so on), or in the case of requests where the value is (by definition) not known to the requester, the length is **0x00**.

For example, the process of TLV encoding the OID in a request to retrieve the contents of **OID 3.5.7.9** with 9 being a leaf node looks like this (all values are hexadecimal unless otherwise noted):

1) First, realize OID 3.5.7.9 is to be interpreted as a nested set of containers. Node 3 contains node 5, which contains node 7, which contains node 9. Although node 9 presumably contains one or more values, the entity making the request for its contents, by definition, does not know the contents when composing the request, so in this example, node 9 is treated like an "empty bucket." In the corresponding response, node 9 would include its actual contents.
2) Because TLV is built from the inside out (bottom up), start the encoding process with the node at the end which has a primary value equal to 9. Like all OIDs in this standard it is **private**, and because it is a leaf node at the bottom / end of the OID tree and therefore does not contain any further nested TLV data objects, it is **primitive**: 0x09 primary value OR 0b11000000 private OR 0b00000000 primitive = **C9**. As explained above, in this case it is an empty bucket, so its length is **00**. Its TLV encoding is **C9 00**.
3) Up one level to the node with primary value equal to 7, it is also **private**, but unlike the previous level it contains a nested TLV data object (C9 00 from above) so it is **constructed** instead of primitive: 0x07 primary value OR 0b11000000 private OR 0b00100000 constructed = **E7**. It has contents C9 00 so length is **02**. Its TLV encoding is **E7 02 C9 00**.
4) Up one level to the node with primary value 5, it is **private** and contains TLV data object E7 02 C9 00 from above, so it is **constructed**: 0x05 primary value OR 0b11000000 private OR 0b00100000 constructed = **E5**. It has contents E7 02 C9 00 so length is **04**. Its TLV encoding is **E5 04 E7 02 C9 00**.
5) Up to the root with primary value 3, it is **private** and **constructed** so tag is **E3**, value is the TLV data object from above, so length is equal to that object's length **06**. Its encoding is **E3 06 E5 04 E7 02 C9 00**, which is the final encoding of whole "empty bucket" OID **3.5.7.9**.

Appendix B Barcode Reader Symbologies

A Barcode symbology refers to the way in which data is encoded in a barcode. It uses either spaced lines, dots or squares. When read, these symbols are decoded and converted to data. The table below lists all of the supported Symbologies and which are enabled by default.

Table D8.6-1 – Barcode Reader Supported Symbologies

Symbology	Default
AIM 128	Disabled
Aztec	Enabled
Codabar	Enabled
Code 11	Disabled
Code128	Enabled
Code 32	Disabled
Code 39	Enabled
Code 93	Disabled
Data Matrix	Enabled
EAN-8	Enabled
EAN-13	Enabled
Febraban	Disabled
GSI-128 (UCC/EAN-128)	Enabled
GS1 Databar (RSS)	Disabled
Industrial 25	Disable
Interleaved 2 of 5,	Enabled
ISSN	Disabled
ISBN	Disabled
ITF-14	Disabled
ITF-6	Disabled
Matrix 2 of 5	Enabled
Micro QR	Disabled
MSI Plessey	Disabled
PDF417	Enabled
Plessey	Disabled
QR Code	Enabled
Standard 25	Disabled
UPC-E	Enabled
UPC-A	Enabled

Appendix C Erasing EMV Configurations

C.1 Load CAPK with AID: 0000000000 to erase all current CAPK keys with this CAPK key:

Table 8.6-24 - Get Request Example

Example (Hex)
AA00810401CED8128444D812810400000300A22B81040000008D8201048320B9C1F228E41A1F0 B6173E00423C5B58A952DE1111E0CC5E33C3044A5D3FD2FCAA30A8108303030303333030870 101

Table 8.6-25 - Get Response Example

Example (Hex)
AA00810482CED812820400000000

Table 8.6-26 - Get Request Example

Example (Hex)
AA0081080400D812000030084818D4D47544B41503130C10400000300CE7D00000000000001600 00000000000000000000000000000000000000000000000000000000000000000000000000000000 00000000000000000000000000000000000000000000000000000000000000000000000000000000 00000000000000000000000000583F99A8DF1A414B11A1607402AE67722C4F59B9

Table 8.6-27 - Get Response Example

Example (Hex)
AA0081048200D812820400000000

C.2 Empty Amex DRL: Load AMEX DRL file with “EMPTY” content

Table 8.6-28 - Get Request Example

Example (Hex)
AA008104018DD8128444D812810400000500A22B810400000011820104832O5E39BD3937CF6B80DECC2B6FD0ABED138D5CC69A6CE55C8790BBA58254B66ED5A30A8108303030303035303087 0101

Table 8.6-29 - Get Response Example

Example (Hex)
AA008104828DD812820400000000

Table 8.6-30 - Get Request Example

Example (Hex)
AA0081080400D8120000050084114D47544B41503130C10400000500CE01AA

Table 8.6-31 - Get Response Example

Example (Hex)
AA0081048200D812820400000000

C.3 Empty Terminal: Load Terminal file with “EMPTY” content

Table 8.6-32 - Get Request Example

Example (Hex)
AA0081040114D8128444D812810400000000A22B810400000011820104832054366603 7C01700B9215F365346E7C883290F201E0D844D9F32B0D05A098B519A30A8108303030303030303030870101

Table 8.6-33 - Get Response Example

Example (Hex)
AA0081048214D812820400000000

Table 8.6-34 - Get Request Example

Example (Hex)
AA0081080400D8120000000084114D47544B41503130C10400000000CE01AA

Table 8.6-35 - Get Response Example

Example (Hex)
AA0081048200D812820400000000

C.4 Empty Processing: Load Processing file with “EMPTY” content and 1 delimiter (FF33)

Table 8.6-36 - Get Request Example

Example (Hex)
AA008104011BD8128444D812810400000100A22B810400000288201048320B25A77AD6582338898AA0498D1C15512AC9AE7C1B021884238813A28423B20E9A30A8108303030303031303087010 1

Table 8.6-37 - Get Response Example

Example (Hex)
AA008104821BD812820400000000

Table 8.6-38 - Get Request Example

Example (Hex)
AA0081080400D8120000010084284D47544B41503130C10400000100CE18AA9A6CCE52123D2315BF3759D466BC0F7C4572A754FF3300

Table 8.6-39 - Get Response Example

Example (Hex)
AA0081048200D812820400000000

C.5 Empty Entry Point: Load Entry file with “EMPTY” content and 1 delimiter (FF35)

Table 8.6-40 - Get Request Example

Example (Hex)
AA0081040135D8128444D812810400000200A22B810400000288201048320d747a07102af345bbe0f41669ebd63aac8770aef5fa37ca00a2eaaf4112b8a2cA30A8108303030303030323030870101

Table 8.6-41 - Get Response Example

Example (Hex)
AA0081048235D812820400000000

Table 8.6-42 - Get Request Example

Example (Hex)
AA0081080400D8120000020084284D47544B41503130C10400000200CE18AA3dcfbc51fcdfd06ada36181f1cd3aee3f8f879bcFF3500

Table 8.6-43 - Get Response Example

Example (Hex)
AA0081048200D812820400000000