MAGTEK®
SECURITY FROM THE INSIDE

DynaMag, DynaMAX, eDynamo, uDYNAMO, aDynamo, BulleT, mDynamo, Dynawave, tDynamo, iDynamo 6, DynaGlass, iDynamo 5 (Gen III)

Secure Card Reader Authenticator Programmer's Manual (Android)

June 2024

Manual Part Number:
D99875728-600

REGISTERED TO ISO 9001:2015

Table 0-1 – Revisions

Revision Number	Date	Notes
10	09/29/2015	Initial Release
11	11/09/2015	Added SendExtendedCommand and OnDeviceExentededResponse.
12	11/11/2015	Added 0x08 event value for Card Removed in OnTransactionStatus.
13	11/18/2015	Updated event value definitions for 0x03 and 0x04 in OnTransactionStatus.
20	08/04/2016	Added DynaPro format for EMV transaction messages. Added getCardPAN, and setConnectionRetry.
30	03/10/2017	Updated the device capatibility list for setConnectionType.
31	06/09/2017	Fix table in section **5.5** providing values for card events; misc. formatting fixes
40	08/21/2018	Added support for tDynamo and DynaWave.
50	01/31/2019	Updated startTransaction to support Quick Chip mode. Removed service declarations to be specified in the AndroidManifest.xml file. Updated to correctly reference Bluetooth LE. Added getDeviceFeature and getPowerManagementValue methods.

Revision Number	Date	Notes
51	11/19/2020	Removed appendix B,C,D and advised customer to use command manuals. Updated tested OS. Added support of iDynamo 6, DynaGlass.
60	09/01/2021	Updated to support banking functions for DynaGlass. Added section for enums, constants, and status.
600	06/04/2024	Added iDynamo 5 (Gen III) in the supported device list at sections 1.1 and 3.1.

SOFTWARE LICENSE AGREEMENT

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

TERMS, CONDITIONS, AND RESTRICTIONS

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

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

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

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

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

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

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

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

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

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

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

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

Table of Contents

1 Introduction

This document provides instructions for software developers who want to create software solutions that include a MagTek Secure Card Reader Authenticator (SCRA) device connected to an Android device via the Audio/Headset Interface, Bluetooth, Bluetooth LE, or USB.

1.1 About the MagTek SCRA Demo

The MagTek SCRA Demo, available from MagTek, provides demonstration source code and a reusable MTSCRA library that provides developers of custom software solutions with an easy-to-use interface for Dynamag, DynaMAX, eDynamo, uDynamo, aDynamo, BulleT, mDynamo, DynaWave, tDynamo, iDynamo 6, DynaGlass, and iDynamo 5 (Gen III) readers. Developers can include the MTSCRA library in custom branded software which can be distributed to customers or distributed internally as part of an enterprise solution.

1.2 Nomenclature

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

- **Device** refers to the reader device that receives and responds to the command set specified in this document.
- **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, laptops, tablets, smartphones, 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 “USB host.”

1.3 SDK Contents

File	Description
mtscra.jar	MagTek SCRA Library

1.4 System Requirements

Development Environment: Eclipse 4.3 and above or Android Studio 3 and above.

Android Operating System: 4.4.2 and above

2 How to Set Up the MTSCRA Library for Projects

To add the MTSCRA library to a custom software project in the Eclipse development environment, follow these steps:

1) Create or open your custom software project in Eclipse.
2) Copy the following JAR file to the **`libs`** subfolder of your software project:
   `mtscra.jar`
3) Ensure your project settings are set up correctly.
4) Clean, build, and run your custom software project to make sure the library imported correctly.
5) In your custom software, create an instance of `MTSCRA`. For examples, see the source code included with the MagTek SCRA Demo project and/or the Code Examples section in this document.
6) Depending on the connection types supported, the project should include the uses-features and uses-permissions as specified in the table below in its AndroidManifest.xml file. For examples, see the AndroidManifest.xml included with the MagTek SCRA Demo project

Connection Type	AndroidManifest

Audio	`<uses-permission android:name=”android.permission.RECORD_AUDIO”/>`
	`<uses-permission android:name=”android.permission.MODIFY_AUDIO_SETTINGS”/>`

BLE
BLEEMV
	`<uses-feature android:name=”android.hardware.bluetooth_le”/>`
	`<uses-permission android:name=”android.permission.BLUETOOTH”/>`
	`<uses-permission android:name=”android.permission.BLUETOOTH_ADMIN”/>`

Bluetooth	`<uses-permission android:name=”android.permission.BLUETOOTH”/>`
	`<uses-permission android:name=”android.permission.BLUETOOTH_ADMIN”/>`

USB Serial	`<uses-feature android:name=”android.hardware.usb.host” />`

3 MTSCRA Class Methods

After creating an instance of the MTSCRA class in your custom software project, use the methods described in this section to communicate with SCRA device.

3.1 setConnectionType

This method sets the connection type of the device.

```
public void setConnectionType(MTConnectionType connectionType)
```

Parameters:

Parameter	Description

connectionType	MTConnectionType value:
	`MTConnectionType.Unknown,`
	`MTConnectionType.Audio,`
	`MTConnectionType.BLE,`
	`MTConnectionType.BLEEMV,`
	`MTConnectionType.Bluetooth,`
	`MTConnectionType.USB,`
	`MTConnectionType.Serial,`
	`MTConnectionType.Net,`
	`MTConnectionType.Net_TLS12,`
	`MTConnectionType.Net_TLS12_Trust_All,`
	`MTConnectionType.BLEEMVT,`
	`MTConnectionType.AIDL`

The following table shows the connection types supported by the various SCRA devices:

Connection Type	SCRA Device	Note
Audio	aDynamo uDynamo	
Bluetooth LE	DynaMAX	Android 4.4.2 and above
Bluetooth LE EMV	eDynamo	Android 4.4.2 and above
Bluetooth LE EMVT	tDynamo	Android 4.4.2 and above
Bluetooth	BulleT	Card swipe only
USB	BulletT DynaMag DynaMAX eDynamo tDynamo DynaWave iDynamo 6 iDynamo 5 (Gen III)	Host must support USB On-The-Go.
Serial	DynaGlass	Android 7.1.2

Connection Type	SCRA Device	Note
Serial	DynaWave	Android 4.4.2 and above
AIDL	DynaGlass	Android 7.1.2

Return Value: None

3.2 setConnectionRetry

This function when set to True instructs the SDK to automatically retry to connect to the Bluetooth LE reader upon pairing. If set to False, the Bluetooth LE reader will be disconnected after pairing. This function is set to False by default in the SDK, but MagTek highly recommends to set this flag to True in your application in order to make sure that you'll have a successful secure connection between Android OS and Bluetooth LE reader after the paring.

```
public void setConnectionRetry(boolean connectionRetry)
```

Parameters:

Parameter	Description
connectionRetry	Connection retry value: `True` = SDK will retry the conection after pairing. `False` = SDK will not retry the connection after pairing.

Return Value: None

3.3 setAddress

This method sets the address of the device.

```
public void setAddress(String deviceAddress)
```

Parameters:

Parameter	Description
deviceAddress	String value of the address.

Return Value: None

3.4 setDeviceConfiguration

This method sets the configuration parameters for the device.

```
public void setDeviceConfiguration(String configuration)
```

Parameters:

Parameter	Description
configuration	String value of the configuration parameters to be used for the device.

Audio Reader Device Configuration		
Parameter	**Default Value**	**Configurable Values**
PAN_MOD10_CHECKDIGIT	TRUE	FALSE, TRUE
INPUT_AUDIO_SOURE	VRECOG	MIC, VRECOG
INPUT_SAMPLE_RATE_IN_HZ	44100	32000, 44100, 48000

Return Value: None

3.5 openDevice

This method opens connection to the device.

```
public void openDevice()
```

Parameters: None

Return Value: None

3.6 closeDevice

This method closes the connection to the device.

```
public void closeDevice()
```

Parameters: None

Return Value: None

3.7 isDeviceConnected

This method returns whether the device is connected or not.

```
public boolean isDeviceConnected()
```

Parameters: None

Return Value:
Return true if the device is connected. Otherwise, return false.

3.8 isDeviceEMV

This method returns whether the device supports EMV or not.

```
public boolean isDeviceEMV()
```

Parameters: None

Return Value:
Return true if EMV is supported by the device. Otherwise, return false.

3.9 getMaskedTracks

Get stored masked tracks data. If decodable track data exists for a given track, it is located in the Masked Track Data field that corresponds to the track number. The length of each Masked Track Data field is fixed at 112 bytes, but the length of valid data in each field is determined by the Masked Track Data

Length field that corresponds to the track number. Masked Track Data located in positions greater than indicated in the Masked Track Data Length field are undefined and should be ignored.

The Masked Track Data is decoded and converted to ASCII and then it is masked. The Masked Track Data includes all data starting with the start sentinel and ending with the end sentinel. Much of the data is masked; a specified mask character is sent instead of the actual character read from the track. Which characters are masked depends on the format of the card. Only ISO/ABA (Financial Cards with Format Code B) and AAMVA cards are selectively masked; all other card types are either entirely masked or sent totally in the clear. There is a separate masking property for ISO/ABA cards and AAMVA cards. See ***D99875475*** for the ISO Track Masking property and the AAMVA Track Masking property for more information. See ***D99875475*** for a description on how ISO/ABA and AAMVA cards are identified.

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

```
public String getMaskedTracks()
```

Parameters: None

Return Value:
Return stored masked tracks data string.

3.10 getTrack1

Get stored track1 data. This field contains the encrypted track data for track 1.

```
public String getTrack1()
```

Parameters: None

Return Value:
Return stored track1 data string.

3.11 getTrack2

Get stored track2 data. This field contains the encrypted track data for track 2.

```
public String getTrack2()
```

Parameters: None

Return Value:
Return stored track2 data string.

3.12 getTrack3

Get stored track3 data. This field contains the encrypted track data for track 3.

```
public String getTrack3()
```

Parameters: None

Return Value:

Return stored track3 data string.

3.13 getTrack1Masked

Get stored masked track1 data.

```
public String getTrack1Masked()
```

Parameters: None

Return Value:
Return stored masked track1 data string.

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

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

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

3.14 getTrack2Masked

Get stored masked track2 data.

```
public String getTrack2Masked()
```

Parameters: None

Return Value:
Return stored masked track2 data string.

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

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

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

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

- The Expiration Date and Birth Date are transmitted unmasked.
- All other characters are set to the specified mask character.

3.15 getTrack3Masked

Get stored masked track3 data.

```
public String getTrack3Masked()
```

Parameters: None

Return Value:
Return stored masked track3 data string.

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

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

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

3.16 getMagnePrint

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

```
public String getMagnePrint()
```

Parameters: None

Return Value:
String containing the MagnePrint data.

3.17 getMagnePrintStatus

Not supported on aDynamo. Get the card MagnePrint status. For more information, see ***D9875475***

```
public String getMagnePrintStatus()
```

Parameters: None

Return Value:

String containing the MagnePrint status.

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

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

Usage Legend:

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

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

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

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

3.18 getDeviceSerial

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

```
public String getDeviceSerial()
```

Parameters: None

Return Value:
Return stored device serial number.

3.19 getSessionID

Not suported on aDynamo. This 8-byte Binary field contains the encrypted version of the current Session ID. Its primary purpose is to prevent replays. After a card is read, this property will be encrypted, along with the card data, and supplied as part of the transaction message. The clear text version of this will never be transmitted. To avoid replay, the application sets the Session ID property before a transaction and verifies that the Encrypted Session ID returned with card data decrypts to the value set.

```
public String getSessionID()
```

Parameters: None

Return Value:
Returns a string containing the session id.

3.20 getKSN

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

```
public String getKSN()
```

Parameters: None

Return Value:
Return stored key serial number.

3.21 getDeviceName

Get device model name.

```
public String getDeviceName()
```

Parameters: None

Return Value:
Return device model name.

3.22 clearBuffers

Clears buffered data retrieved from the reader.

```
public void clearBuffers()
```

Parameters: None

Return Value: None

3.23 getBatteryLevel
Retrieves battery level.

```
public long getBatteryLevel()
```

Parameters: None

Return Value:
Battery Level (0 to 100)

3.24 getSwipeCount
Retrieves swipe count.

```
public long getSwipeCount()
```

Parameters: None

Return Value:
Long value representing swipe count.

3.25 getCapMagnePrint
Retrieves MagnePrint capabilities.

```
public String getCapMagnePrint()
```

Parameters: None

Return Value:
String representing MagnePrint capabilities:
0 = No MagnePrint,
1 = Short MagnePrint,
2 = Long MagnePrint

3.26 getCapMagnePrintEncryption
Retrieves MagnePrint Encryption capabilities.

```
public String getCapMagnePrintEncryption()
```

Parameters: None

Return Value:
String representing MagnePrint Encryption capabilities:
0 = No Encryption,
1 = Same as MagStripe (8122),
other values TBD.
If absent, default value is 1.

3.27 getCapMagneSafe20Encryption
Retrieves MagneSafe 2.0 Encryption capabilities.

```
public String getCapMagneSafe20Encryption()
```

Parameters: None

Return Value:
String representing MagneSafe 2.0 Encryption capabilities. 0 = Not supported, other values TBD.

3.28 getCapMagStripeEncryption
Retrieves MagneStripe Encryption capabilities.

```
public String getCapMagStripeEncryption()
```

Parameters: None

Return Value:
String representing MagStripe Encryption capabilities. 0 = No Encryption, 1 = TDES DUKPT / PIN Variant, other values TBD

3.29 getCapMSR
Retrieves MSR capabilities.

```
public String getCapMSR()
```

Parameters: None

Return Value:
String representing MSR capabilities. 0 = No MSR, 1 = MSR.

3.30 getCapTracks
Retrieves Track capabilities.

```
public String getCapTracks()
```

Parameters: None

Return Value:
String representing Track capabilities:

- Bit 0 = 1 / Track 1 supported,
- Bit 1 = 1 / Track 2 supported,
- Bit 2 = 1 / Track 3 supported,
- All other bits = 0.

3.31 getCardDataCRC
Retrieves CRC from card data.

```
public long getCardDataCRC()
```

Parameters: None

Return Value:
Card data CRC

3.32 getCardExpDate

Retrieves card experiation date from card data.

```
public String getCardExpDate()
```

Parameters: None

Return Value:
String representing card expiration date.

3.33 getCardIIN

Retrieves Issuer Identification Number (IIN) from card data.

```
public String getCardIIN()
```

Parameters: None

Return Value: String representing card IIN.

3.34 getCardLast4

Retrieves Last 4 digits of card number from card data.

```
public String getCardLast4()
```

Parameters: None

Return Value:
String representing card last 4 digits.

3.35 getCardName

Retrieves card name from card data.

```
public String getCardName()
```

Parameters: None

Return Value:
String representing card name.

3.36 getCardPAN

Retrieves PAN from card data.

```
public String getCardPAN()
```

Parameters: None

Return Value:

String representing card PAN.

3.37 getCardPANLength

Retrieves PAN length from card data.

```
public int getCardPANLength()
```

Parameters: None

Return Value:
PAN length

3.38 getCardServiceCode

Retrieves Service Code.

```
public String getCardServiceCode()
```

Parameters: None

Return Value:
String representing service code.

3.39 getCardStatus

Retrieves Card Status.

```
public String getCardStatus()
```

Parameters: None

Return Value:
String representing card status.

3.40 getEncodeType

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

```
public int getEncodeType()
```

Parameters: None

Return Value:

Value	Encode Type	Description
0	ISO/ABA	ISO/ABA encode format. At least one track in ISO/ABA format, Track 3 not AAMVA format.
1	AAMVA	AAMVA encode Track 3 is AAMVA format, Tracks 1 and 2 are ISO/ABA if correctly decoded.
2	Reserved	

Value	Encode Type	Description
3	Blank	The card is blank. Only occurs if all tracks decode without error and without data.
4	Other	The card has a non-standard encode format. For example, ISO/ABA track 1 format on track 2.
5	Undetermined	The card encode type could not be determined because no tracks could be decoded. (Combination of Error tracks and Blank Tracks, at least one Error track).
6	None	No decode has occurred. This type occurs if no magnetic stripe data has been acquired since the data has been cleared or since the reader was powered on. This reader only sends an Input report when a card has been swiped so this value will never occur.

3.41 getDataFieldCount

Retrieves data field count.

```
public int getDataFieldCount()
```

Parameters: None

Return Value:
Data field count

3.42 getHashCode

Retrieves SHA-x hash code.

```
public String getHashCode()
```

Parameters: None

Return Value:
String representing SHA-x hash code.

3.43 getDeviceConfig

Retrieves device configuration.

```
public String getDeviceConfig(String configType)
```

Parameters:

Parameter	Description

configType	configType can be one of:
	
	`8180: Send TLV Version on Power Up`
	`8181: Send Discovery on Power Up`
	`8280: Send Card name`
	`8281: Send Card IIN`
	`8282: Send Card Last 4 Digits of PAN`
	`8283: Send Card Expiration`
	`8284: Send Card Service Code`
	`8285: Send Card PAN Length`

Return Value:
String representing device configuration.

3.44 getEncryptionStatus

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

```
public String getEncryptionStatus()
```

Parameters: None

Return Value:
String representing decryption status as a 2-byte binary field.

- Bit 0 = DUKPT Keys exhausted (1=exhausted, 0=keys available)
- Bit 1 = Initial DUKPT key Injected, always set to One (Primary DUKPT Key)
- Bit 2 = Encryption Enabled, always set to One
- Bit 3 = Reserved (always set to zero)
- Bit 4 = Reserved (always set to zero)
- Bit 5 = Reserved (always set to zero)
- Bit 6 = Reserved (always set to zero)
- Bit 7 = Reserved (always set to zero)
- Bit 8 = Reserved (always set to zero)
- Bit 9 = Initial DUKPT key injected (Secondary DUKPT Key)
- Bit 10 = DUKPT Key used for encryption, 0=Primary, 1=Secondary
- Bit 11 = DUKPT Key Variant used to encrypt data, 0=PIN Variant, 1=Data Variant/Bidirectional
- Bits 12–15 = Unassigned (always set to Zero)

3.45 getFirmware

Retrieves firmware version.

```
public String getFirmware()
```

Parameters: None

Return Value:
String representing firmware version

3.46 getMagTekDeviceSerial

Retrieves MagTek device serial number.

```
public String getMagTekDeviceSerial()
```

Parameters: None

Return Value:
String representing MagTek device serial number

3.47 getResponseData

Retrieves response data.

```
public String getResponseData()
```

Parameters: None

Return Value:
String representing response data.

3.48 getResponseType

Retrieves response type.

```
public String getResponseType()
```

Parameters: None

Return Value:
String representing response type. For Audio Reader, always "C101".

3.49 getTagValue

Retrieves the value of the specified tag.

```
public String getTagValue(String tag, String data)
```

Parameters:

Parameter	Description
tag	Tag to search for.
data	Data to search from.

Return Value:
String representing tag value.

3.50 getTLVVersion

Retrieves TLV version.

```
public String getTLVVersion()
```

Parameters: None

Return Value:
String representing TLV version as a two-byte hex string.

3.51 getTrackDecodeStatus

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

```
public String getTrackDecodeStatus()
```

Parameters: None

Return Value:
Track Decode Status. Consists of three 2-byte hex values representing the decode status for tracks 1, 2, and 3 (respectively from left to right).   Values are:

- 00 = Track OK
- 01 = Track read Error
- 02 = Track is Blank

3.52 getSDKVersion

Retrieves SDK version.

```
public String getSDKVersion()
```

Parameters: None

Return Value:
The version information of the SDK.

3.53 sendCommandToDevice

Send command to device.

```
public int sendCommandToDevice(String command)
```

Parameters:

Parameter	Description
command	Command string to send to the device.

Return Value:

- 0 = Success
- 9 = Error
- 15 = Busy

3.54 startTransaction (EMV Device Only)

This function starts an EMV L2 transaction.

```
public int startTransaction(
     byte timeLimit,
     byte cardType,
     byte option,
     byte[] amount,
     byte transactionType,
     byte[] cashBack,
     byte[] currentCode,
     byte reportingOption)
```

Parameters:

Parameter	Description
timeLimit	Specifies the maximum time, in seconds, allowed to complete the total transaction. This includes time for the user to insert the card, choose a language, choose an application, and online processing. If this time is exceeded, the transaction will be aborted and an appropriate Transaction Status will be available. Value 0 is not allowed.
cardType	Card Type to Read: 0x01 = Magnetic Stripe (as alternative to EMV L2, card swipe causes abort of EMV L2) 0x02 = Contact smart card 0x04 = Contactless smart card Note: Multiple Card Types can be selected, for example: Set this byte to 3 to read both Magnetic Stripe and Contact Smart Card.
option	0x00 = Normal 0x01 = Bypass PIN 0x02 = Force Online 0x04 = Acquirer not available (Note: prevents long timeout on waiting for host approval) (causes “decline” to be generated internally if ARQC is generated) To use Quick Chip mode, set the most significant bit to ‘1’. 0x80 = Quick Chip, Normal 0x81 = Quick Chip, Bypass PIN 0x82 = Quick Chip, Force Online
amount	Amount Authorized (EMV Tag 9F02, format n12, 6 bytes) in hex string For example: “000000000999”, means 9.99 dollars.

Parameter	Description
transactionType	Valid values: 0x00 = Purchase (listed as “Payment” on ICS) 0x01 = Cash Advance (not supported for this reader) 0x02 or 0x09 = Cash back (0x09 not supported, contactless) 0x04 = Goods (Purchase) 0x08 = Services (Purchase) 0x10 = International Goods (Purchase) 0x20 = Refund 0x40 = International Cash Advance or Cash Back 0x80 = Domestic Cash Advance or Cash Back
cashBack	Cash back Amount (if non-zero, EMV Tag 9F03, format n12, 6 bytes) in hex string. For example: “000000001000”, means 10.00 dollars.
currencyCode	Transaction Currency Code (EMV Tag 5F2A, format n4, 2 bytes) Sample Valid values: 0x0840 – US Dollar 0x0978 – Euro 0x0826 – UK Pound
reportingOption	This single byte field indicates the level of Transaction Status notifications the host desires to receive during the course of this transaction. 0x00 = Termination Status only (normal termination, card error, timeout, host cancel) 0x01 = Major Status changes (terminations plus card insertions and waiting on user) 0x02 = All Status changes (documents the entire transaction flow)

Return Value:
- 0 = Success
- 9 = Error
- 15 = Busy

3.55 setUserSelectionResult (EMV Device Only)

This function sets the user selection result. It should be called after receiving the OnUserSelectRequest event which is triggered after the user makes a selection.

```
public int setUserSelectionResult(byte status, byte selection)
```

Parameters:

Parameter	Description
status	Indicates the status of User Selection: 0x00 – User Selection Request completed, see Selection Result 0x01 – User Selection Request aborted, cancelled by user 0x02 – User Selection Request aborted, timeout

Parameter	Description
selection	Indicates the menu item selected by the user. This is a single byte zero based binary value.

Return Value:

- 0 = Success
- 9 = Error
- 15 = Busy

3.56 setAcquirerResponse (EMV Device Only)

This function informs EMV device to process transaction decision from acquirer.

```
public int setAcquirerResponse(byte[] response)
```

Parameters:

Parameter	Description
response	The first two bytes (most significant byte first) indicate the total length of the following byte array. The byte array contains the ARQC Response message. For details about the ARQC response, see the ***Programmer's Manual (COMMANDS)*** for the specific device you are communicating with.

Return Value:

- 0 = Success
- 9 = Error
- 15 = Busy

3.57 cancelTransaction (EMV Device Only)

This function cancels a transaction while waiting for the user to insert a card.

```
public int cancelTransaction()
```

Parameters: None

Return Value:

- 0 = Success
- 9 = Error
- 15 = Busy

3.58 sendExtendedCommand (EMV Device Only)

Send extended command to device.

```
public int sendExtendedCommand(String command)
```

Parameters:

Parameter	Description
command	Hexadecimal string of the byte array for the extended command. The first two bytes represent the value of the extended command. The next two bytes (most significant byte first) indicate the total length the following data in bytes.

Return Value:

- 0 = Success
- 9 = Error
- 15 = Busy

3.59 getDeviceFeatures()

Retrieves features for the device.

```
public MTDeviceFeatures getDeviceFeatures()
```

Parameters: None

Return Value:

```
public class MTDeviceFeatures
    {
    public boolean MSR;
    public boolean Contact;
    public boolean Contactless;
    public boolean PINPad;
    public boolean MSRPowerSaver;
    public boolean BatteryBackedClock;
    public boolean SRED;
    public boolean SignatureCapture;
    public boolean ManualEntry;
    }
```

3.60 getPowerManagementValue()

Retrieves power management value for the device.

```
public String getPowerManagementValue()
```

Parameters: None

Return Value: Returns the PM value associated with the device.

Parameter	Power Management
BulleT KB BulleT SPP	PM1
cDynamo	

Parameter	Power Management
Dynamag, Dynamag Duo, USB Enc IntelliHead V5	
Dynasty	PM3
DynaMAX	PM2
DynaPAD	
DynaWave	
eDynamo	PM3
Flash	PM1
iDynamo	
iDynamo 5	
Home Banking (Dynamo LCD)	
kDynamo	PM5
mDynamo	
P-series and I-65 w/V5	
pDynamo	PM6
sDynamo	
SPI Encrypting IntelliHead V5	
tDynamo	PM5
UART Enc IntelliHead V5	
uDynamo	PM4
U-Finity	PM1

3.61 requestCardSwipe

Request the device to display a message for swiping a card.

```
public int requestSwipeCard(
      byte waitTime,
      byte messageID,
      byte beepCount)
```

Parameters:

Parameter	Description
waitTime	Wait Time in seconds, (0x01 – 0xFF; 0x00 = Infinite Wait Time)

Parameter	Description
messageID	Card Message ID to display: 0x00 = Swipe Card / Idle (alternating) 0x01 = Swipe Card 0x02 = Please Swipe Card 0x03 = Please Swipe Card Again 0x04 = Chip Error, Use Mag Stripe 0x07 = Please Swipe, Insert or Tap 0x08 = Insert Card 0x09 = Please Swipe or Insert Card 0x0C = Tap Card 0x0D = Please Insert or Tap Card
beepCount	0x00 = None 0x01 = Single Beep 0x02 = Double Beep

Return Value:

- 0 = Success
- 9 = Error
- 15 = Busy

3.62 getMSRData

This function requests the device to send MSR data after calling requestCardSwipe().

```
public int getMSRData()
```

Return Value:

- 0 = Success
- 9 = Error
- 15 = Busy

3.63 requestPINEntry

Request PIN entry by the cardholder.

```
public int requestPINEntry(
      byte waitTime,
      byte pinMode,
      byte pinLength,
      byte beepCount,
      byte pinOption,
      byte[] amount,
      byte[] pan)
```

Parameters:

Parameter	Description
waitTime	Wait Time in seconds, (0x01 – 0xFF; 0x00 = 256 seconds)

Parameter	Description
pinMode	Message mode to display on the device. 0x00 = Enter PIN 0x01 = Enter PIN Amount 0x02 = Reenter PIN Amount 0x03 = Reenter PIN 0x04 = Verify PIN
pinLength	Range for the PIN entered. High nibble = Max PIN length (<=12) Low nibble = Min PIN length (>=4) Example: 0x64 is for PIN length of 4 to 6 Example: 0xC4 is for PIN length of 4 to 12 Example: 0xC8 is for PIN length of 8 to 12
beepCount	0x00 = None 0x01 = Single Beep 0x02 = Double Beep
pinOption	Bit(7,6,5) PIN Block Format 0b000 = ISO Format 0 0b001 = ISO Format 1 (No PAN Required) 0b011 = ISO Format 3 0b100 = ISO Format 4 Bit(4,3) (PIN Language Select Only) Language Select: 0b00 = Disabled 0b01 = English or French Only 0b10 = All Languages as defined by DFDF2D Bit 2 Wait Message Bit 1 Verify PIN Bit 0 Reserved
amount	Numeric: n12
pan	Min.: 9 bytes; Max.: 21 bytes

Return Value:

- 0 = Success
- 9 = Error
- 15 = Busy

3.64 requestManualCardEntry

Request the card holder to enter card data manually.

```
public int requestManualCardEntry(
      byte waitTime,
      byte options,
      byte beepCount)
```

Parameters:

Parameter	Description
waitTime	Wait Time in seconds, (0x01 – 0xFF; 0x00 = 256 seconds)
options	Message and mode to display on the device. Bit (7,6,5) Reserved Bit 4 0=Use PAN min 9, max 19 1=Use PAN min 14, max 21 Bit 3 1=Use PAN in PIN block creation Bit 2 1=Use QwickCodes entry Bit (1,0) 0 = Acct, Date, CVC 1 = Acct, Date 2 = Acct, CVC 3 = Acct
beepCount	0x00 = None 0x01 = Single Beep 0x02 = Double Beep

Return Value:

- 0 = Success
- 9 = Error
- 15 = Busy

3.65 requestSignature

Request the card holder to sign on the screen.

```
public int requestSignature(
      byte waitTime,
      byte options,
      byte beepCount)
```

Parameters:

Parameter	Description
waitTime	Wait Time in seconds, (0x01 – 0xFF; 0x00 = 256 seconds)

Parameter	Description
options	0x00 = Timeout clears any signature data 0x01 = Timeout returns timeout status plus length collected. Sig Data can be requested.
beepCount	0x00 = None 0x01 = Single Beep 0x02 = Double Beep

Return Value:

- 0 = Success
- 9 = Error
- 15 = Busy

3.66 getSignature

Request the card holder signature from the device. To be used after calling requestSignature().

```
public int getSignature()
```

Return Value:

- 0 = Success
- 9 = Error
- 15 = Busy

3.67 requestEncryptedInputData

Request the card holder to enter text for encryption.

```
public int requestEncryptedInputData(
      byte waitTime,
      byte beepCount)
```

Parameters:

Parameter	Description
waitTime	Wait Time in seconds, (0x01 – 0xFF; 0x00 = 256 seconds)
beepCount	0x00 = None 0x01 = Single Beep 0x02 = Double Beep

Return Value:

- 0 = Success
- 9 = Error
- 15 = Busy

3.68 getEncryptedInputData

Request the encrypted input data from the device. To be used after calling requestEncryptedInputData().

```
public int getEncryptedInputData()
```

Return Value:

- 0 = Success
- 9 = Error
- 15 = Busy

3.69 cancelRequest

Request to cancel a command or request while the device is waiting for the card holder.

```
public int cancelRequest()
```

Return Value:

- 0 = Success
- 9 = Error
- 15 = Busy

4 MTSCRAConfig Class Methods

This class provides methods to retrieve configuration parameters from the server.

4.1 getConfigurationXML

This method retrieves the configuration parameters from the server as an XML data. The method will throw an exception if there is a problem with retrieving the configuration XML.

```
public String getConfigurationXML(
     String username,
     String password,
     int readerType,
     SCRAConfigurationDeviceInfo deviceInfo,
     String address,
     int timeout) throws MTSCRAException
```

Parameters:

Parameter	Description
username	String value of the username.
password	String value of the password.
readerType	Integer value indicating the type of reader device.
deviceInfo	SCRAConfigurationDeviceInfo value containing information pertaining to the device.
address	String value of the address for connection to the server.
timeout	Integer value of the timeout in seconds for connection to the server.

Return Value:
String value of the configuration parameters retrieved from the server.

4.2 getConfigurationResponse

This method retrieves the configuration parameters from XML data as a ProcessMessageResponse object. The method will throw an exception if there is a problem with retrieving the configuration XML.

```
public ProcessMessageResponse getConfigurationResponse(
     String xmlConfig) throws MTSCRAException
```

Parameters:

Parameter	Description
xmlConfig	String value of the the configuration parameters from the server.

Return Value: ProcessMessageResponse containing the configuration parameters.

4.3 getConfigurationParams

This method retrieves the configuration parameters from the server as an XML data. The method will throw an exception if there is a problem with retrieving the configuration XML.

```
public String getConfigurationParams(
      String model,
      ProcessMessageResponse messageReponse) throws MTSCRAException
```

Parameters:

Parameter	Description
model	String value containing the device model.
messageResponse	ProcessMessageResponse containing the configuration parameters.

Return Value:
String value of the configuration parameters for the specified device model.

5 MTSCRA Callback Messages

5.1 OnDeviceConnectionStateChanged

This message occurs when the state of the device is changed.

Parameter	Description

Obj	MTConnectionState value indicating the state of the device:
	`Disconnected`
	`Connecting`
	`Connected`
	`Disconneting`

5.2 OnCardDataStateChanged

This message occurs when the state of the card information is changed.

Parameter	Description

obj	MTCardDataState value indicating the state of the card data:
	`DataNotReady`
	`DataReady`
	`DataError`

5.3 OnDataReceived

This message occurs when card information is received from the device.

Parameter	Description
obj	IMTCardData value containing the card data received.

5.4 OnDeviceResponse

This message occurs when a response is received from the device.

Parameter	Description
obj	String containing the response data received from the device.

5.5 OnTransactionStatus (EMV Device Only)

This message occurs when transaction status update is received from the EMV device.

Parameter	Description
obj	Byte array containing the data received from the device. See table below for descriptions of the data.

Offset	Field Name	Value
0	Event	Indicates the event that triggered this notification: 0x00 = No events since start of transaction 0x01 = Card Inserted 0x02 = Card Error 0x03 = Transaction Progress Change 0x04 = Waiting for User Response 0x05 = Timed Out 0x06 = Transaction Terminated 0x07 = Host Cancelled Transaction 0x08 = Card Removed
1	Current Transaction Time remaining	Indicates the remaining time available, in seconds, for the transaction to complete. If the transaction does not complete within this time it will be aborted.
2	Current Transaction Progress Indicator	This one byte field indicates the current processing stage for the transaction: 0x00 = No transaction in progress 0x01 = waiting for user to insert card 0x02 = powering up the card 0x03 = selecting the application 0x04 = waiting user language selection 0x05 = waiting user application selection 0x06 = initiating application 0x07 = reading application data 0x08 = offline data authentication 0x09 = process restrictions 0x0A = card holder verification 0x0B = terminal risk management 0x0C = terminal action analysis 0x0D = generating first application cryptogram 0x0E = card action analysis 0x0F = online processing 0x10 = waiting online processing response 0x11 = transaction completion 0x12 = transaction error 0x13 = transaction approved 0x14 = transaction declined
3-4	Final Status	TBD

5.6 OnDisplayMessageRequest (EMV Device Only)

This message occurs when the EMV device has display message to present to the user.

Parameter	Description
obj	Byte array containing the display message. If the length is zero, the request to clear the display.

5.7 OnUserSelectionRequest (EMV Device Only)

This message occurs when the EMV device has user selection message to present to the user.

Parameter	Description
obj	Byte array containing the data received from the device. See table below for descriptions of the data.

Offset	Field Name	Value
0	Selection Type	This field specifies what kind of selection request this is: 0x00 = Application Selection 0x01 = Language Selection
1	Timeout	Specifies the maximum time, in seconds, allowed to complete the selection process. If this time is exceeded, the host should send the User Selection Result command with transaction will be aborted and an appropriate Transaction Status will be available. Value 0 is not allowed.
2	Menu Items	This field is variable length and is a collection of "C" style zero terminated strings (maximum 17 strings). The maximum length of each string is 20 characters, not including a Line Feed (0x0A) character that may be in the string. The last string may not have the Line Feed character. The first string is a title and should not be considered for selection. It is expected that the receiver of the notification will display the menu items and return (in the User Selection Result request) the number of the item the user selects. The minimum value of the Selection Result should be 1 (the first item, #0, was a title line only). The maximum value of the Selection Result is based on the number of items displayed.

5.8 OnARQCReceived (EMV Device Only)

This message occurs when ARQC is received from the EMV device.

Parameter	Description
Obj	Byte array containing the data received from the device. See table below for descriptions of the data.

Offset	Field Name	Value
0	Message Length	Two byte binary, most significant byte first. This gives the total length of the ARQC message that follows.
2	ARQC Message	Byte array containing the ARQC Message. For details about the ARQC format, see the ***Programmer's Manual (COMMANDS)*** for the specific device you are communicating with.

5.9 OnTransactionResult (EMV Device Only)

This message occurs when transaction result is received from the EMV device.

Parameter	Description
Obj	Byte array containing the data received from the device. See table below for descriptions of the data.

Offset	Field Name	Value
0	Signature Required	This field indicates whether a card holder signature is required to complete the transaction: 0x00 = No signature required 0x01 = Signature required If a signature is required, it is expected that the host will acquire the signature from the card holder as part of the transaction data.
1	Batch Data Length	Two byte binary, most significant byte first. This gives the total length of the Batch Data that follows.
3	Batch Data	Byte array containing the Batch Data. For details about the batch data format, see the ***Programmer's Manual (COMMANDS)*** for the specific device you are communicating with..

5.10 OnEMVCommandResult (EMV Device Only)

This message occurs when an EMV command result is received from the EMV device.

Result Code Description
0x0000 = Success 0x0001 = Failure 0x0381 = Failure, DUKPT scheme is not loaded 0x0382 = Failure, DUKPT scheme is loaded but all of its keys have been used 0x0383 = Failure, DUKPT scheme is not loaded (Security Level not 3 or 4) 0x0384 = Invalid Total Transaction Time field 0x0385 = Invalid Card Type field 0x0386 = Invalid Options field 0x0387 = Invalid Amount Authorized field 0x0388 = Invalid Transaction Type field 0x0389 = Invalid Cash Back field 0x038A = Invalid Transaction Currency Code field 0x038B = Invalid Selection Status 0x038C = Invalid Selection Result 0x038D = Failure, no transaction currently in progress 0x038E = Invalid Reporting Option 0x038F = Failure, transaction already in progress 0x0390 = Device Has No Keys 0x0391 = Invalid Device Serial Number 0x0392 = Invalid Type of MAC field 0x0393 = Invalid Slot Number field 0x0394 = Invalid Operation field 0x0395 = Invalid Database Selector field 0x0396 = Invalid System Date and Time 0x0396 = Invalid Objects to Write field 0x0396 = Invalid Tags to Read field 0x0396 = Invalid Date / Time data (Date / Time has not been set yet) 0x0397 = Invalid MAC 0x0398 = No Slots Available 0x0399 = Object Write Protected 0x039B = Invalid CAPK Checksum 0x039C = Invalid Configuration Identifier

5.11 OnDeviceExtendedResponse (EMV Device Only)

This message occurs when an extended response is received from the device.

Parameter	Description
obj	Byte array containing the extended response data received from the device. The first two bytes represent the result codes for the extended command. The next two bytes (most significant byte first) indicate the total length of the following data in bytes.

5.12 OnDeviceState (EMV Device Only)

This message occurs when the device changes state.

Parameter	Description
obj	Byte array containing the data received from the device.

Offset	Field Name	Value							
0	Device State	0x00 = Idle 0x01 = Session 0x02 = Wait For Card 0x03 = Wait For PIN 0x04 = Wait For Selection 0x05 = Displaying Message 0x06 = Test (Reserved for future use) 0x07 = Manual Card Entry 0x08 = Wait for Signature Capture (SC-S Only | SC-F Only) 0x09 = Wait Cardholder Entry 0x0A = Chip Card 0x0B = ICC Kernel Test 0x0C = EMV Transaction 0x0D = Show PAN							
1	Session State	Bit 7	6	5	4	3	2	1	0
		Pwr Chg	RFU	RFU	RFU	Card Data	MSRP AN	EXPAN	Amt
		The bits of Session State mean the following: Pwr Chg: 1 = Power Change Occurred (occurs on Power up or after a USB resume) Card Data: 1 = Card Data Available MSR PAN: 1 = PAN Parsed from Card EXPAN: 1 = External PAN Sent Amt: 1 = Amount sent							

Offset	Field Name	Value							
2	Device Status	0x00 = OK. Otherwise, the possible values are listed below: Bit 7 = Device Error Status: 1 = Device Error Bit 6 = Authentication Status: 0 = Not Authenticated 1 = Authenticated Bit 5 = 0 Bit 4 = Tamper: 0 = Normal 1 = Tamper Detected Bits [3,2] = MSR Key Status: 00 = MSR Key OK 01 = MSR Key Exhausted 10 = No MSR Key 11 = MSR Key Not Bound Bits [1,0] = PIN Key Status: 00 = PIN Key OK 01 = PIN Key Exhausted 10 = No PIN Key 11 = PIN Key Not Bound							
3	Device Certificate Status	Bit 7	6	5	4	3	2	1	0
		MSR CRL	PIN CRL	TLS (RSA)Cert	Manufacturer Unbind	MSR Key Loader CA	PIN Key Loader CA	Device CA	Device Cert
		0 = Certificate does not exist in the device 1 = Certificate exists in the device							

5.13 OnCardStatus (EMV Device Only)

This message occurs when the card status has changed.

Parameter	Description
obj	Byte array containing the data received from the device.

Offset	Field Name	Value
0	Operation Status	
1	Card Status	

Offset	Field Name	Value
2	Card Type	

5.14 OnCardData (EMV Device Only)

This message occurs when the device sends card data.

Parameter	Description
obj	Byte array containing the data received from the device.

```
F8<len> /*container tag for encryption */
     DFDF59(Encrypted Data Primitive)<len><Encrypted Data val (Decrypt
     data to read tags)>
     DFDF56(Encrypted Transaction Data KSN)<len><val>
     DFDF57(Encrypted Transaction Data Encryption Type)<val>
     DFDF58(# of bytes of padding in DFDF59)<len><val>
     FC<len> /*container tag for encrypted generic data */
          F4<len>/* container tag for encrypted MSR data */
          DFDF36 <EncT1status><len><val>
          DFDF37 <EncT1data><len><val>
          DFDF38 <EncT2status><len><val>
          DFDF39 <EncT2data><len><val>
          DFDF3A <EncT3status><len><val>
          DFDF3B <EncT3data><len><val>
          DFDF3C <Encrypted Magneprint Data><len><val>
          DFDF43 <Magneprint Status Data><len><val>
          DFDF50(MSR KSN Data)<len><val> /*sent in the clear*/
          DFDF51(MSR EncryptionType)<len><val>
```

5.15 OnPINResponse (EMV Device Only)

This message occurs when the device sends the response to a PIN request.

Parameter	Description
obj	Byte array containing the data received from the device.

Offset	Field Name	Value
0	Operation Status	
1	PIN BLOCK Format	0000 0000 = ISO Format 0 (PAN Required) 0000 0001 = ISO Format 1 (No PAN Required) 0000 0011 = ISO Format 3 (PAN Required) 0000 0100 = ISO Format 4 (Not Supported)
2	PIN BLOCK Encryption Type	0xxx xxxx = Fixed key 1xxx xxxx = DUKPT key xx00 xxxx = TDES xx01 xxxx = AES128 xx10 xxxx = AES256 xxxx xx00 = Data variant xxxx xx01 = PIN variant xxxx xx10 = MAC variant
3 - 14	PIN KSN.	PIN KSN. If fixed PIN Key is used, KSN is all zeroes.
14 - 21	EPB	Encrypted PIN Block (EPB). If PIN entry was successful, this contains the PIN data, encrypted using the PIN variant of the current PIN DUKPT working key. Format after decryption depends on the PIN Option the host specified, and on the device's Session State: • If the Session State indicates there is no PAN available (from card swipe or sent via command), the device creates the EPB using ISO Format 1. • If there is a PAN, the device creates the EPB using the PIN Option the host specified in the command.

5.16 OnSignatureState (EMV Device Only)

This message occurs when the signature state has change.

Parameter	Description
obj	Byte array containing the data received from the device.

Offset	Field Name	Value
0	Operation Status	
1	Reserved	
2	Signature length (low byte)	
3	Signature length (high byte)	

5.17 OnSignature (EMV Device Only)

This message occurs when the device sends the response to a signature request.

Parameter	Description
obj	Byte array containing the data received from the device.

Offset	Field Name	Value
0	Message	This Signature is an array of bytes.

5.18 OnEncryptedDataState (EMV Device Only)

This message occurs when the encrypted data state has changed.

Parameter	Description
obj	Byte array containing the data received from the device.

Offset	Field Name	Value
0	Operation Status	
1	Reserved	
2	Input Data length (low byte)	
3	Input Data length (high byte)	

5.19 OnEncryptedData (EMV Device Only)

This message occurs when the device returns encrypted data.

Parameter	Description
obj	Byte array containing the data received from the device.

Offset	Field Name	Value
0	Operation Status	
1 - 10	KSN.	KSN. If fixed Key is used, KSN is all zeroes.
11 - n	Encrypted Input Data	The encrypted data encrypted using the DATA variant of the current DATA DUKPT working key.

6 Enums/Constants/Status

6.1 Operation Status

Value (Hex)	Result Code	Description
0x00	Success	The command completed successfully.
0x01	Failure	The command failed.
0x02	Bad Parameter	The command failed due to a bad parameter or command syntax error.
0x03	Redundant	The command is redundant.
0x04	Bad Cryptography	A bad cryptography operation occurred.
0x05	Delayed	The request is refused because the device is delaying requests as a defense against brute-force hacking.
0x06	No Keys	No keys are loaded.
0x07	Invalid Operation	Depends on the context of the command.
0x08	Response not available	The response is not available.
0x09	Not enough power	The battery is too low to operate reliably.
0x0A	Extended response first packet	The device is returning the first (and possibly only) packet of an Extended Response.
0x0B	Extended command pending	An extended command is pending and the device is waiting for more data.
0x0C	Extended command notification	Deprecated
0x0D	Not implemented	The command is not implemented.
0x0E	Unarmed tamper, device not ready	The tamper device is not ready to be armed.
0x0F	Unarmed tamper, bad signature	The tamper is not armed because of a bad signature.

6.2 Operation Status for Extended commands

Result Code Description
0x0000 = Success 0x0001 = Failure 0x0381 = Failure, DUKPT scheme is not loaded 0x0382 = Failure, DUKPT scheme is loaded but all of its keys have been used 0x0383 = Failure, DUKPT scheme is not loaded (Security Level not 3 or 4) 0x0384 = Invalid Total Transaction Time field 0x0385 = Invalid Card Type field 0x0386 = Invalid Options field 0x0387 = Invalid Amount Authorized field 0x0388 = Invalid Transaction Type field 0x0389 = Invalid Cash Back field 0x038A = Invalid Transaction Currency Code field 0x038B = Invalid Selection Status 0x038C = Invalid Selection Result 0x038D = Failure, no transaction currently in progress 0x038E = Invalid Reporting Option 0x038F = Failure, transaction already in progress 0x0390 = Device Has No Keys 0x0391 = Invalid Device Serial Number 0x0392 = Invalid Type of MAC field 0x0393 = Invalid Slot Number field 0x0394 = Invalid Operation field 0x0395 = Invalid Database Selector field 0x0396 = Invalid System Date and Time 0x0396 = Invalid Objects to Write field 0x0396 = Invalid Tags to Read field 0x0396 = Invalid Date / Time data (Date / Time has not been set yet) 0x0397 = Invalid MAC 0x0398 = No Slots Available 0x0399 = Object Write Protected 0x039B = Invalid CAPK Checksum 0x039C = Invalid Configuration Identifier

6.3 MTBankingEvent

```
public class MTBankingEvent
{
    public static final int OnDeviceState           = 1400;
    public static final int OnCardStatus            = 1401;
    public static final int OnCardData              = 1402;
    public static final int OnPINResponse           = 1403;
    public static final int OnSignatureState        = 1404;
    public static final int OnSignature             = 1405;
    public static final int OnEncryptedDataState    = 1406;
    public static final int OnEncryptedData         = 1407;
}
```

Parameter	Description
OnDeviceState	The device sends the host this notification to report the condition/status of the device.
OnCardStatus	The device sends the host this notification to report the card status.
OnCardData	The device sends the host this notification to report the card data.
OnPINResponse	The device sends the host this notification to report the PIN data.
OnSignatureState	The device sends the host this notification to report that signature data is available.
OnSignature	The device sends the host this notification to report the signature data.
OnEncryptedDataState	The device sends the host this notification to report the status of encrypted data.
OnEncryptedData	The device sends the host this notification to report the encrypted data.

6.4 MTEMVEvent

```
public class MTEMVEvent
{
    public static final int OnTransactionStatus         = 200;
    public static final int OnDisplayMessageRequest     = 201;
    public static final int OnUserSelectionRequest      = 202;
    public static final int OnARQCReceived              = 203;
    public static final int OnTransactionResult         = 204;
    public static final int OnEMVCommandResult          = 205;
    public static final int OnDeviceExtendedResponse    = 206;
}
```

Parameter	Description
OnTransactionStatus	The device sends the host this notification to report progress during an EMV transaction.
OnDisplayMessageRequest	The device sends this notification to request that the host display a message for the cardholder. The host should display the message.

Parameter	Description
OnUserSelectionRequest	This device sends the host this notification to inform the host that a cardholder selection is needed before the device can continue processing the current transaction.
OnARQCReceived	The device sends the host this notification to send ARQC data for the host to process. After the host processes the ARQC data, it should send the command setAcquirerResponse() to inform the device it can proceed with the transaction.
OnTransactionResult	The device sends this notification to provide the host with final information from the transaction. It usually includes data and an indication of whether a signature is required.
OnEMVCommandResult	The device sends the host this notification to report the result of an EMV command.
OnDeviceExtendedResponse	The device sends the host this notification to report the response to an extended command.

6.5 MTError

```
public enum MTError
{
   CardDataError,
   ConnectionError
}
```

Parameter	Description
CardDataError	The device sends the host this notification to report and error in card data.
ConnectionError	This this notification reports an error when attempting to connect to the device.

6.6 MTSCRAEvent

```
public class MTSCRAEvent
{
   public static final int OnDeviceConnectionStateChanged     = 0;
   public static final int OnCardDataStateChanged             = 1;
   public static final int OnDataReceived                     = 2;
   public static final int OnDeviceResponse                   = 3;
   public static final int OnDeviceNotPaired                  = 4;
}
```

6.7 MTConnectionState

```
public enum MTConnectionState
{
   Disconnected,
   Connected,
   Error,
   Connecting,
   Disconnecting
}
```

Parameter	Description
Disconnected	Device is disconnected.
Connected	Device is connected and ready for transacting.
Error	There was an error either connecting or disconnecting the device.
Connecting	Device is in the process of connecting. The next state is to be Connected.
Disconnecting	Device is in the process of disconnecting. The next state is to be Disconnected.

6.8 MTConnectionType

```
public enum MTConnectionType
{
   Unknown(0),
   Audio(1),
   BLE(2),
   BLEEMV(3),
   Bluetooth(4),
   USB(5),
   Serial(6),
   Net(7),
   Net_TLS12(8),
   Net_TLS12_Trust_All(9),
   BLEEMVT(10),
   AIDL(11);
}
```

7 Commands

Custom software can use the **sendCommandToDevice** method to send direct commands to the device. This section provides information about commonly used commands.

7.1 Discovery

To send a command to device, use:

```
public void sendCommandToDevice(String command)
```

Parameters: Use “C10206C20503840900” as command string for audio readers.

Return Value:
The following device information will be retrieved.
Device SN, internal: Device serial number created by chip manufacturer. Use `getDeviceSerial` method to retrieve data.

Device SN, MagTek: Device serial number created by MagTek. Use `getDeviceSerialMagTek` method to retrieve data.

Device Firmware Part Number: Device firmware part number. Use `getFirmware` method to retrieve data.

Device Model Name: Device model name. Use `getDeviceName` method to retrieve data.

Device TLV Version: Device TLV version. Use getTLVVersion method to retrieve data.

Device Part Number: Device part number. Use `getDevicePartNumber` method to retrieve data.

Capability - MSR: 0 = No MSR, 1 = MSR. Use `getCapMSR` method to retrieve data.

Capability - TRACKS:

- 0 = Supported tracks: None.
- 1 = Supported tracks: Track1.
- 2 = Supported tracks: Track2.
- 3 = Supported tracks: Track1, Track2.
- 4 = Supported tracks: Track3.
- 5 = Supported tracks: Track1, Track3.
- 6 = Supported tracks: Track2, Track3.
- 7 = Supported tracks: Track1, Track2, Track3.

Use `getCapTracks` method to retrieve data.

Capability - MagStripe Encryption: 0 = No Encryption, 1 = TripDES DUKPT. Use `getCapMagStripeEncryption` method to retrieve data.

Appendix A Code Examples

A.1 Open Device

```
if (! m_SCRA.isDeviceConnected())
{
      m_SCRA.openDevice();
}
```

A.2 Close Device

```
if (m_SCRA != null)
{
      m_SCRA.closeDevice();
}
```

A.3 Get Connection Status Of Device

```
if (! m_SCRA.isDeviceConnected())
{

}
```

A.4 Receiving Card Data From Device

```
private Handler m_SCRAHandler =
      new Handler(new SCRAHandlerCallback());

private MTSCRA m_SCRA = new MTSCRA(m_SCRAHandler);

private class SCRAHandlerCallback implements Callback
{
      public boolean handleMessage(Message msg)
      {
            Switch (msg.what)
            {
                  Case MTSCRAEvent.CardDataReceived:
                        OnCardDataReceived();
                        break;
            }
      }
}

public void OnCardDataReceived()
{
      // Display raw card data
      CardData.Text = m_SCRA.getResponseData();
```

```
     // Display last 4 digits of the card
     CardLast4.Text = m_SCRA.getCardLast4();
}
```

A.5 Send Command To Device

```
if (mSCRA.isDeviceConnected())
{
     // Send discovery command
     m_SCRA.sendCommandToDevice("C10206C20503840900", 0);
}
```