MAGTEK®
SECURITY FROM THE INSIDE

SCRA Reader

Audio Reader Family

Programmer's Reference (Windows Phone)

December 1, 2014

Manual Part Number:
D99875647-11

REGISTERED TO ISO 9001:2008

Table 0.1 - Revisions

Rev Number	Date	Notes
11	Dec 1, 2014	Initial Release

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 a Windows Phone's audio connector. It is part of a larger library of documents designed to assist SCRA implementers.

2 About MagTekSCRA

Custom Windows software installed on a host PC can communicate with a MagTek SCRA device via the audio connector interface using the MagTekSCRA library. The software project must link to MagTekSCRA.dll.

Supported platforms include Windows Phone 8.0.

3 MagTekSCRA Class Methods

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

3.1 openDevice

This function opens the reader.

```
public void openDevice()
```

Parameters: None

Return Value: None

3.2 closeDevice

This function closes the reader.

```
public void closeDevice()
```

Parameters: None

Return Value: None

3.3 isDeviceConnected

This function retrieves the connection status of the reader.

```
public bool isDeviceConnected()
```

Parameters: None

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

3.4 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 the ISO Track Masking property and the AAMVA Track Masking property for more information. See ***99875475*** for a description on how ISO/ABA and AAMVA cards are identified.

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

```
public string getMaskedTracks()
```

Parameters: None

Return Value:
Return stored masked tracks data string.

3.5 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.

3.6 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.

3.7 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.

3.8 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 cardholder'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.

3.9 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.

3.10 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.

3.11 getMagnePrint

Supported on uDynamo only. 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: Empty String

3.12 getMagnePrintStatus

Supported on uDynamo only.

```
public string getMagnePrintStatus()
```

Parameters: None

Return Value: Empty String

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.

3.13 getDeviceSerial

Get stored device serial number. This 16-byte ASCII field contains the device serial number. The device serial number is a NULL (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.

3.14 getSessionID

Not suported on Audio Reader. 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 host software 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: Empty String

3.15 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.

3.16 getDeviceName

Get device model name.

```
public string getDeviceName()
```

Parameters: None

Return Value:
Return device model name.

3.17 setDeviceType

Set the type of device (Audio or None).

```
public void setDeviceType(int deviceType)
```

Parameters:
deviceType can be one of

- MagTekSCRA.DEVICE_TYPE_NONE
- MagTekSCRA.DEVICE_TYPE_AUDIO

Return Value: None

3.18 setDeviceID

Set device identifier for the Audio device.

```
public void setDeviceID(string deviceID)
```

Parameters:
deviceID = ID of the Audio device to connect to.

Return Value: None

3.19 clearBuffers

Clears buffered data retrieved from the reader.

```
public void clearBuffers()
```

Parameters: None

Return Value: None

3.20 getBatteryLevel

Retrieves battery level.

```
public long getBatteryLevel()
```

Parameters: None

Return Value:
Battery Level (0 to 100)

3.21 getSwipeCount

Retrieves swipe count.

```
public long getSwipeCount()
```

Parameters: None

Return Value:
Long value representing swipe count

3.22 getCapMagnePrint

Retrieves MagnePrint Capabilities.

```
public string getCapMagnePrint()
```

Parameters: None

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

3.23 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.

3.24 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.

3.25 getCapMagStripeEncryption

Retrieves magnetic stripe encryption capabilities.

```
public string getCapMagStripeEncryption()
```

Parameters: None

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

3.26 getCapMSR

Retrieves MSR Capabilities.

```
public string getCapMSR()
```

Parameters: None

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

3.27 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.

3.28 getCardDataCRC

Retrieves CRC from card data.

```
public long getCardDataCRC()
```

Parameters: None

Return Value:
Card data CRC

3.29 getCardExpDate

Retrieves card expiration date from card data.

```
public string getCardExpDate ()
```

Parameters: None

Return Value:
String representing card expiration date. The data returned is dependent on the partial information provided by the reader. If the reader supports emitting partial data for the fields below, the SDK will provide them as is. If the reader supports masking, the SDK will retrieve the information from the masked tracks supplied by the reader.

3.30 getCardIIN

Retrieves Issuer Identification Number (IIN) from card data.

```
public string getCardIIN()
```

Parameters: None

Return Value: String representing card IIN.

3.31 getCardLast4

Retrieves Last 4 digits of card number from card data.

```
public string getCardLast4()
```

Parameters: None

Return Value:
String representing card last 4 digits.

3.32 getCardName

Retrieves card name from card data.

```
public string getCardName()
```

Parameters: None

Return Value:
String representing card name.

3.33 getCardPANLength

Retrieves PAN length from card data.

```
public int getCardPANLength()
```

Parameters: None

Return Value: PAN length.

3.34 getCardServiceCode

Retrieves Service Code.

```
public string getCardServiceCode()
```

Parameters: None

Return Value: String representing service code.

3.35 getCardStatus

Retrieves Card Status.

```
public string getCardStatus()
```

Parameters: None

Return Value:
String representing card status

Card status

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

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	
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).

Value	Encode Type	Description
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.

3.36 getDataFieldCount

Retrieves data field count.

```
public int getDataFieldCount()
```

Parameters: None

Return Value:
Data field count

3.37 getHashCode

Retrieves SHA-x hash code.

```
public string getHashCode()
```

Parameters: None

Return Value:
String representing SHA-x hash code

3.38 getDeviceConfig

Retrieves device configuration.

```
public string getDeviceConfig(string configType)
```

Parameters:
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

3.39 getDeviceType

Retrieves device type.

```
public int getDeviceType()
```

Parameters: None

Return Value
Device type is one of the following:

- MagTekSCRA.DEVICE_TYPE_NONE
- MagTekSCRA.DEVICE_TYPE_AUDIO

3.40 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)

3.41 getFirmware

Retrieves firmware version.

```
public string getFirmware()
```

Parameters: None

Return Value:
String representing firmware version

3.42 getMagTekDeviceSerial

Retrieves MagTek device serial number.

```
public string getMagTekDeviceSerial()
```

Parameters: None

Return Value:
String representing MagTek device serial number

3.43 getResponseData

Retrieves response data.

```
public string getResponseData()
```

Parameters: None

Return Value:
String representing response data

3.44 getResponseType

Retrieves response type.

```
public string getResponseType()
```

Parameters: None

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

3.45 getTagValue

Retrieves the value of the specified tag.

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

Parameters:
tag = Tag to search for
data = Data to search from

Return Value: String representing tag value

3.46 getTLVVersion

Retrieves TLV version.

```
public string getTLVVersion()
```

Parameters: None

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

3.47 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

3.48 getSDKVersion

Retrieves SDK version.

```
public string getSDKVersion()
```

Parameters: None

Return Value: SDK Version

3.49 sendCommandToDevice

Send command to device.

```
public void sendCommandToDevice(string command)
```

Parameters:
command is the command string to send to the device.

Return Value:
Send command to device. See section 5 **Commands** for a list of valid commands.

3.50 setConfigurationParams

Sets configuration parameters.

```
public void setConfigurationParams(string configParams)
```

Parameters:
configParams = “PAN_MOD10_CHECKDIGIT = TRUE” or “PAN_MOD10_CHECKDIGIT = FALSE” (default is TRUE).

3.51 getConfigurationParams

Retrieves configuration parameters.

```
public string getConfigurationParams()
```

Parameters: None

Return Value: String representing configuration parameters.

4 MagTekSCRA Events

4.1 DeviceEventStateChange

Occurs when connection state is changed.

```
public event EventHandler<int>

public delegate void DeviceEventStateChangeHandler(object sender, int
state)
```

Parameter	Description
sender	Object representing the SDK instance that published the event
state	Value representing the connection state of the device. MagTekSCRA.DEVICE_STATE_DISCONNECTED MagTekSCRA.DEVICE_STATE_CONNECTED MagTekSCRA.DEVICE_STATE_CONNECTING MagTekSCRA.DEVICE_STATE_CANNOT_CONNECT

4.2 DeviceEventDataStart

Occurs when data transmission is started.

```
public event EventHandler DeviceEventDataStart

public delegate void DeviceEventDataStartHandler(object sender)
```

Parameter	Description
sender	Object representing the publisher of the event

4.3 DeviceEventDataChange

Occurs when data transmission is completed.

```
public event EventHandler<string> DeviceEventDataChange

public delegate void DeviceEventDataChangeHandler(object sender,
string data)
```

Parameter	Description
sender	Object representing the publisher of the event
data	String representing data received.

4.4 DeviceEventDataError

Occurs when data has error.

```
public event EventHandler<string> DeviceEventDataError

public delegate void DeviceEventDataErrorHandler(object sender, string
error)
```

Parameter	Description
sender	Object representing the publisher of the event
error	String representing error received.

4.5 DeviceEventDataSample

This notification is received when a sample of data is available.

```
public event EventHandler<short> DeviceEventDataSample

public delegate void DeviceEventDataSampleHandler(object sender, short
value)
```

Parameter	Description
sender	Object representing the publisher of the event
value	Value representing a sample of data received.

5 Commands

5.1 Discovery

Send discovery command to device.

```
public void sendCommandToDevice(string command)
```

Parameters: Use "C10206C20503840900" as command string.

Return Value:
The following device information will be retrieved:

Parameter	Description
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	Use getCapMSR method to retrieve data. 0 = No MSR 1 = MSR
Capability - TRACKS	Use getCapTracks method to retrieve data. 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
Capability – Magstripe Encryption	Use getCapMagStripeEncryption method to retrieve data. 0 = No Encryption 1 = TripDES DUKPT

Appendix A Code Examples

A.1 Open Device

```
if (! mMTSCRA.isDeviceConnected())
{
      mMTSCRA.setDeviceType(mMTSCRA.DEVICE_TYPE_AUDIO);
      mMTSCRA.openDevice();
}
```

A.2 Close Device

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

A.3 Get Reader Connection Status

```
if (! mMTSCRA.isDeviceConnected())
{
      …
}
```

A.4 Receiving Card Data From Reader

```
if (! mMTSCRA.isDeviceConnected())
{
      mMTSCRA.setDeviceType(mMTSCRA.DEVICE_TYPE_AUDIO);
      mMTSCRA.DeviceEventDataChange += OnDeviceEventDataChange;
      mMTSCRA.openDevice();
}

public void OnDeviceEventDataChange(Object sender, string data)
{
      // Display raw card data
      CardData.Text = data;

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

A.5 Send Command To Reader

```
if (mMTSCRA.isDeviceConnected())
{
      // Send discovery command
      mMTSCRA.sendCommandToDevice("C10206C20503840900");
}
```