Audio Reader
PROGRAMMING REFERENCE MANUAL
for Android Devices

PART NUMBER 99875583-1

SEPTEMBER 2012

MAGTEK®

REGISTERED TO ISO 9001:2008

1710 Apollo Court
Seal Beach, CA 90740
Phone: (562) 546-6400
FAX: (562) 546-6301
Technical Support: (651) 415-6800
www.magtek.com

REVISIONS

Rev Number	Date	Notes
1.01	30 Apr 12	Initial Release
1.02	20 Jun 12	Added setConfigurationParams
1.03	27 Aug 12	Added getConfigurationResponse, getConfigurationXML, setConfigurationXML, setConfiguration
1.04	24 Sept 12	Added more detailed descriptions to getMaskedTracks, getTrack1-3, getTrack1Masked-3, getMagneprint, getMagnePrintStatus, getDeviceSerial, getSessionID, getKSN, getEncryptionStatus

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 ABOVE ADDRESS, 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 ABOVE ADDRESS, OR E-MAILED TO support@magtek.com.

Table of Contents

SECTION 1. MagTekSCRA CLASS

Classes	Description
MagTekSCRA	This class allows you to perform reader functions.

Methods:

openDevice:	Open device
closeDevice:	Close device
isDeviceConnected	Check the connection status of reader
getMaskedTracks	Retrieves the existing stored masked track data
getTrack1	Retrieves encrypted track1
getTrack2	Retrieves encrypted track2
getTrack3	Retrieves encrypted track3
getTrack1Masked	Retrieves masked track1
getTrack2Masked	Retrieves masked track2
getTrack3Masked	Retrieves masked track3
getMagnePrint	Retrieves encrypted MagnePrint
getMagnePrintStatus	Retrieves encrypted MagnePrintStatus
getDeviceSerial	Retrieves device serial number
getSessionID	Retrieves session ID
getKSN	Retrieves key serial number
getDeviceName	Get the associated name of the Bluetooth Device
setDeviceType	Set the type of device (Bluetooth or Audio)
setDeviceID	Set device identifier for the Bluetooth device
clearBuffers	Clears library buffers
getBatteryLevel	Retrieves battery level
getCapMagnePrint	Retrieves MagnePrint Capabilities
getCapMagnePrintEncryption	Retrieves MagnePrint Encryption Capabilities
getCapMagneSafe20Encryption	Retrieves MagneSafe 2.0 Encryption Capabilities
getCapMagStripeEncryption	Retrieves MagStripe Encryption Capabilities
getCapMSR	Retrieves MSR Capabilities
getCapTracks	Retrieves Track Capabilities
getCardDataCRC	Retrieves CRC
getCardExpDate	Retrieves expiration date
getCardIIN	Retrieves IIN
getCardLast4	Retrieves last 4 digits of the card number
getCardName	Retrieves card name
getCardPANLength	Retrieves PAN length
getCardServiceCode	Retrieves service code
getCardStatus	Retrieves status
getDataFieldCount	Retrieves field count
getDeviceConfig	Retrieves device configuration
getDeviceType	Retrieves device type
getEncryptionStatus	Retrieves encryption status
getFirmware	Retrieves firmware version
getMagTekDeviceSerial	Retrieves MagTek device serial number
getOperationStatus	Retrieves operation status
getPANHashSHA1	Retrieves SHA1 PAN hash
getPANHashSHA256	Retrieves SHA256 PAN hash
getResponseData	Retrieves response data
getResponseType	Retrieves response type

getTagValue	Retrieves the value of the specified tag
getTLVVersion	Retrieves TTLV version
getTrackDecodeStatus	Retrieves track decode status
listenForEvents	Sets which events to listen for
sendCommandToDevice	Sends specified command to device, with timeout
getSDKVersion	Retrieves the SDK version
setConfiguration	Sets configuration parameters from server
setConfigurationParams	Sets configuration parameters
setConfigurationXML	Sets configuration parameters
getConfigurationXML	Retrieves the configuration parameters from the server as an XML data
getConfigurationResponse	Retrieves the configuration parameters from XML data as a ProcessMessageResponse Object
getConfigurationResponse	Retrieves the configuration parameters from the server as a ProcessMessageResponse Object.
setConfigurationResponse	Sets configuration parameters

Methods

openDevice:

This function opens the reader.

- `public void openDevice()`

Parameters

None

Return Value

None

closeDevice

This function closes the reader.

- `public void closeDevice(){`

Parameters

None

Return Value

None

isDeviceConnected

This function retrieves the connection status of the reader.

- `public boolean isDeviceConnected()`

Parameters

None

Return Value

TRUE if the device is connected. Otherwise, return FALSE.

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 HID specification requires that reports be fixed in size but the number of bytes encoded on a card may vary. Therefore, the Input Report always contains the maximum amount of bytes that can be encoded on the card and the number of valid bytes in each track is indicated by the Masked Track Data Length field.

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. (Refer Appendix E in Reference Manual 99875475 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

Return Value

Return stored masked tracks data string.

getTrack1

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

- `public String getTrack1()`

Parameters

Return Value

Return stored track1 data string.

getTrack2

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

- `public String getTrack2()`

Parameters

Return Value

Return stored track2 data string.

getTrack3

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

- `public String getTrack3 ()`

Parameters

Return Value

Return stored track3 data string.

getTrack1Masked

Get stored masked track1 data.

- `public String getTrack1Masked()`

Parameters

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.

getTrack2Masked

Get stored masked track2 data.

- `public String getTrack2Masked()`

Parameters

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.

getTrack3Masked

Get stored masked track3 data.

```
- public String getTrack3Masked()
```

Parameters

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.

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

Return Value

Empty String

getMagnePrintStatus

Supported on uDynamo only

- `public String getMagnePrintStatus()`

Parameters

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	=	This is a 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.

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

Return Value

Return stored device serial number.

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 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

Return Value

Empty String

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

Return Value

Return stored key serial number.

getDeviceName

Get the associated name of the Bluetooth Device.
- public String getDeviceName()

Parameters

Return Value

Return associated name of the Bluetooth device.

setDeviceType

Set the type of device (Bluetooth or Audio).
- public void setDeviceType(int lpiDeviceType)

Parameters

lpiDeviceType:

MagTekSCRA.DEVICE_TYPE_NONE
MagTekSCRA.DEVICE_TYPE_AUDIO
MagTekSCRA.DEVICE_TYPE_BLUETOOTH

Return Value

setDeviceID

Set device identifier for the Bluetooth device.
- public void setDeviceID(String lpstrDeviceID)

Parameters

lpstrDeviceID

ID of the Bluetooth device to connect to.

Return Value

clearBuffers

Clears buffered data already retrieved from the reader
- public void clearBuffers()

Parameters

Return Value

getBatteryLevel

Retrieves battery level
- public long getBatteryLevel()

Parameters

Return Value

Battery Level (0 to 100)

getCapMagnePrint

Retrieves MagnePrint Capabilities
- public String getCapMagnePrint()

Parameters

Return Value

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

getCapMagnePrintEncryption

Retrieves MagnePrint Encryption Capabilities
- public String getCapMagnePrintEncryption()

Parameters

Return Value

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

getCapMagneSafe20Encryption

Retrieves MagneSafe 2.0 Encryption Capabilities
- public String getCapMagneSafe20Encryption ()

Parameters

Return Value

String representing MagneSafe 2.0 Encryption Capabilities
0 = Not supported,
other values TBD

getCapMagStripeEncryption

Retrieves MagneStripe Encryption Capabilities
- public String getCapMagStripeEncryption()

Parameters

Return Value

String representing MagStripe Encryption Capabilities
0 = No Encryption,
1 = TDES DUKPT / PIN Variant,
other values TBD

getCapMSR

Retrieves MSR Capabilities
- public String getCapMSR()

Parameters

Return Value

String representing MSR Capabilities
0 = No MSR,
1 = MSR

getCapTracks

Retrieves Track Capabilities
- public String getCapTracks()

Parameters

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.

getCardDataCRC

Retrieves CRC from card data
- public long getCardDataCRC()

Parameters

Return Value

Card data CRC

getCardExpDate

Retrieves CRC from card data
- public String getCardExpDate ()

Parameters

Return Value

String representing card expiration date

Note: 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.

getCardIIN

Retrieves IIN from card data
- public String getCardIIN()

Parameters

Return Value

String representing card IIN

Note: 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.

getCardLast4

Retrieves Last 4 digits of card number from card data
- public String getCardLast4()

Parameters

Return Value

String representing card last 4

Note: 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.

getCardName

Retrieves card name from card data
- public String getCardName()

Parameters

Return Value

String representing card name

Note: 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.

getCardPANLength

Retrieves PAN length from card data
- public int getCardPANLength()

Parameters

Return Value

PAN length

Note: 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.

getCardServiceCode

Retrieves Service Code
- **public String getCardServiceCode()**

Parameters

Return Value

String representing service code

Note: 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.

getCardStatus

Retrieves Card Status
- public String getCardStatus()

Parameters

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).
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.

getDataFieldCount

Retrieves data field count

- public int getDataFieldCount()

Parameters

Return Value

Data field count

getDeviceConfig

Retrieves device configuration

- public String getDeviceConfig(String lpConfigType)

Parameters

lpConfigType

configuration type

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

getDeviceType

Retrieves device type

- public int getDeviceType()

Parameters

Return Value

Device type:
MagTekSCRA.DEVICE_TYPE_NONE
MagTekSCRA.DEVICE_TYPE_AUDIO
MagTekSCRA.DEVICE_TYPE_BLUETOOTH

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

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)

getFirmware

Retrieves firmware version
- public String getFirmware()

Parameters

Return Value
String representing firmware version

getMagTekDeviceSerial

Retrieves MagTek device serial number
- public String getMagTekDeviceSerial()

Parameters

Return Value
String representing MagTek device serial number

getPANHashSHA1

Retrieves SHA 1 hash of PAN and salt
- public String getPANHashSHA1 ()

Parameters

Return Value
String representing SHA 1 hash of PAN

getPANHashSHA256

Retrieves SHA 256 hash of PAN and salt
- public String getPANHashSHA256()

Parameters

Return Value
String representing SHA 256 hash of PAN

getResponseData

Retrieves response data
- public String getResponseData()

Parameters

Return Value

String representing response data

getResponseType

Retrieves response type
- public String getResponseType()

Parameters

Return Value

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

getTagValue

Retrieves the value of the specified tag
- public String getTagValue(String lpstrTag, String lpstrResponse)

Parameters

lpstrTag

tag to search for

lpstrResponse

data to search for that the tag contains

Return Value

String representing tag value

getTLVVersion

Retrieves TLV version
- public String getTLVVersion()

Parameters

Return Value

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

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

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

listenForEvents

Sets which events to listen for
- public void listenForEvents(long lpEvents)

Parameters

lpEvents

event type to listen for:
MagTekSCRA.DEVICE_EVENT_DATA_CHANGE
MagTekSCRA.DEVICE_EVENT_DATA_ERROR
MagTekSCRA.DEVICE_EVENT_DATA_START

sendCommandToDevice

Sends command to device
- public void sendCommandToDevice(String lpstrMessage, int iTimeout)

Parameters

lpstrMessage

message to send

iTimeout

timeout wait time

getSDKVersion

Retrieves SDK version
- public String getSDKVersion()

Parameters

Return Value

SDK Version

setConfigurationParams

Sets configuration parameters
- public void setConfigurationParams(String lpstrParams)

Parameters

lpstrParams

PAN_MOD10_CHECKDIGIT = TRUE/FALSE (default is TRUE)

setConfiguration

Sets configuration parameters from server. The method will throw an exception if there is a problem with setting configuration parameters.
-public void setConfiguration(String lpstrUsername, String lpstrPassword, int lpiReaderType,SCRAConfigurationDeviceInfo lpDeviceInfo,String lpstrAddress, int lpTimeout)throws MTSCRAException

Parameters

lpiReaderType

Type of reader. The type of reader is assigned an integer value starting at 1.

lpDeviceInfo

SCRAConfigurationDeviceInfo object. This object will be used as a search criteria if provided. Currently we recommend the usage of Model number only. If this is set to null, SDK will perform the search based on the model number.

lpstrAddress

Address of the remote configuration server. Please refer to the sample code for details

lpTimeout

Timeout specified for connecting to the server.

setConfigurationParams

Sets configuration parameters . The method will throw an exception if there is a problem with setting configuration parameters.
-public void setConfigurationParams(SCRAConfiguration lpConfig)throws MTSCRAException

Parameters

lpConfig

SCRAConfiguration object.

setConfigurationXML

Sets configuration parameters. The method will throw an exception if there is a problem with setting configuration parameters.

-public void setConfigurationXML(String lpstrXML)throws MTSCRAException

Parameters
lpstrXML
XML data received from the configuration server

getConfigurationXML

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 lpstrUsername, String lpstrPassword,int lpiReaderType,SCRAConfigurationDeviceInfo lpDeviceInfo,String lpstrAddress, int lpTimeout)throws MTSCRAException

Parameters
lpiReaderType
Type of reader. The type of reader is assigned an integer value starting at 1.
lpDeviceInfo
SCRAConfigurationDeviceInfo SCRAConfigurationDeviceInfo object. This object will be used as a search criteria if provided. Currently we recommend the usage of Model number only. If this is set to null, SDK will perform the search based on the model number.
lpstrAddress
Address of the remote configuration server. Please refer to the sample code for details
lpTimeout
Timeout specified for connecting to the server.

getConfigurationResponse

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 response object.
-public ProcessMessageResponse getConfigurationResponse(String lpstrXML)throws MTSCRAException

Parameters
lpstrXML
XML data received from the configuration server

getConfigurationResponse

Retrieves the configuration parameters from the server as a `ProcessMessageResponse Object.` . The method will throw an exception if there is a problem with retrieving the configuration response object.
-public ProcessMessageResponse getConfigurationResponse(String lpstrUsername, String lpstrPassword, int lpiReaderType,SCRAConfigurationDeviceInfo lpDeviceInfo,String lpstrAddress, int lpTimeout)throws MTSCRAException

Parameters

lpiReaderType

Type of reader. The type of reader is assigned an integer value starting at 1.

lpDeviceInfo

SCRAConfigurationDeviceInfo object. This object will be used as a search criteria if provided. Currently we recommend the usage of Model number only. If this is set to null, SDK will perform the search based on the model number.

lpstrAddress

Address of the remote configuration server. Please refer to the sample code for details

lpTimeout

Timeout specified for connecting to the server.

setConfigurationResponse

Sets configuration parameters

-public void setConfigurationResponse(ProcessMessageResponse lpMessageResponse)throws MTSCRAException

Parameters

lpMessageResponse

Messsage Response Object

SECTION 2. CODE EXAMPLES

Open Device:

```
if (! mMTSCRA.isDeviceConnected()) {
                mMTSCRA.setDeviceType(mMTSCRA.DEVICE_TYPE_BLUETOOTH);
        mMTSCRA.setDeviceID("00:06:66:44:C8:2C");
        mMTSCRA.openDevice();
}
```

Close Device:

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

Get Tracks Data From Reader:

```
private class SCRAHandlerCallback implements Callback {
public boolean handleMessage(Message msg) {
switch (msg.what)
{
        case MagTekSCRA.DEVICE_MESSAGE_STATE_CHANGE:
        {
        switch (msg.arg1)
        {
                case MagTekSCRA.DEVICE_STATE_CONNECTED:
                {
                }
                break;
                case MagTekSCRA.DEVICE_STATE_CONNECTING:
                {
                }
                break;
                case MagTekSCRA.DEVICE_STATE_DISCONNECTED:
                {
                }
                break;
                }
                }
                break;
                case MagTekSCRA.DEVICE_MESSAGE_DATA_CHANGE:
                if (msg.obj != null)
                {
                        mStringCardDataBuffer = (String)msg.obj;
                        //Do something
                        mStringCardDataBuffer="";
                        return true;
                }
                break;
                default:
                if (msg.obj != null)
                {
                        mStringCardDataBuffer = (String)msg.obj;
                        //Do something
                        mStringCardDataBuffer="";
                }
                break;
        };
        return false;
        }
}
```

Get Connection Status Of Reader:

```
        if (mMTSCRA.isDeviceConnected()) {
                //Do Something
        }
```