MAGTEK®
SECURITY FROM THE INSIDE

DynaMAX, aDynamo, uDynamo, iDynamo

Secure Card Reader Authenticators Programmer's Reference (iOS)

January 2015

Manual Part Number:
D99875568-20

REGISTERED TO ISO 9001:2008

Table 0.1 - Revisions

Rev Number	Date	Notes
1.01	2011 Dec 22	Initial Release
1.02	2011 Aug 02	Remove Audio reader
1.03	2012 Feb 29	Added Functionality
1.04	2012 Mar 26	Added getBatteryLevel
1.05	2012 Apr 19	Added getSDKVersion & getOperationStatus
1.06	2012 May 01	Made iDynamo-specific
20	2015 Jan 30	Added DynaMAX; reformat; added introduction and how to set up; included aDynamo and uDynamo; general cleanup and clarifying detail. Updated the MTSCRATransactionData enum. Updated the MTSCRADeviceType enum. Added delegate methods: onDataReceived, cardSwipeDidStart, cardSwipeDidGetTransError, onDeviceConnectionDidChange, bleReaderConnected, bleReaderDidDiscoverPeripheral, bleReaderStateUpdated

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.

Table of Contents

1 Introduction

This document provides instructions for software developers who want to create custom software solutions that communicate with DynaMAX, aDynamo, iDynamo, or uDynamo connected to an iOS host via audio connector or BLE. It is part of a larger library of documents designed to assist DynaMAX, aDynamo, iDynamo, and uDynamo implementers, which includes:

- ***D99875475 MagneSafe V5 Programmer's Reference (Commands)***

1.1 About MTSCRADemo

The MTSCRADemo software, available from MagTek, provides demonstration source code and a reusable MTSCRA library that provides developers of custom iOS software solutions with an easy-to-use interface for DynaMAX, aDynamo, iDynamo, and uDynamo. 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 MSR device that receives and responds to the command set specified in this document; in this case, DynaMAX, aDynamo, iDynamo, or uDynamo.
- **Host** refers to the piece of general-purpose electronic equipment the device is connected or paired to, which can send data to and receive data from the device. Host types include PC and Mac computers/laptops, tablets, smartphones, teletype terminals, and even test harnesses. In many cases the host may have custom software installed on it that communicates with the device. When "host" must be used differently, it is qualified as something specific, such as "USB host."

The word "user" is also often used in different ways in different contexts. In this document, **user** generally refers to the **cardholder**.

1.3 SDK Contents

File name	Description
MTSCRA.h	Header file for the MTSCRA SDK
libMTSCRA.a	Library binary for the MTSCRA SDK
MTSCRADemo Folder	Sample code and projects

1.4 System Requirements

Tested devices:

- iPhone 4, 4s, 5, 5s, 6, 6 Plus
- iPad, iPad 2, iPad 3, iPad 4, iPad Air, iPad Air 2
- iPad Mini, iPad Mini with Retina Display, iPad Mini 2
- iPod touch 5th generation

Tested operating systems: iOS 6, iOS 7, iOS 8

Build Platforms: XCode 5, XCode 6

1.5 Interfaces for Operating Systems

The following table matches the device interface to operating system.

Device	Interface	Operating System
aDynamo	Audio	iOS 6, iOS 7, iOS 8
iDynamo	30-pin	iOS 6, iOS 7, iOS 8
	Lightning	iOS 6, iOS 7, iOS 8
uDynamo	Audio	iOS 6, iOS 7, iOS 8
DynaMAX	BLE 4.0	iOS 6, iOS 7, iOS 8

2 How to Set Up the MTSCRA SDK

To add the MTSCRA SDK libraries to a custom software project in the XCode development environment, follow these steps:

1) Download the MTSCRA Demo app from MagTek.com.
2) Open your custom software project in XCode.
3) Open the MTSCRA Demo app folder in Finder.
4) Open the `Lib` subfolder.
5) Include the following files in your custom software project within XCode:
    a) `libMTSCRA.a`
    b) `MTSCRA.h`
6) Ensure the library search paths are set up correctly.
7) Clean, build, and run your custom software project to make sure the library imported correctly.
8) In your custom software, create an instance of `MTSCRA`. For examples, see the source code included with the MTSCRA Demo app and / or **Appendix C Code Examples**.
9) Begin using the features provided by the `MTSCRA` object's methods. For details about these methods, see section **3 MTSCRA Functions**.

3 MTSCRA Functions

To develop an iOS app using the MTSCRA SDK, follow the setup steps in section **2 How to Set Up the MTSCRA SDK**, then create an instance of the `MTSCRA` object in your software project, then call the functions described in this chapter to communicate with the device. For sample code that demonstrates how to use these functions, see the contents of the MTSCRADemo folder included with the SDK.

Generally, these functions will run in one of two modes:

- **Asynchronous** functions will return data using the event handlers (callback functions) defined in section **4 MTSCRA Delegate Methods**.
- **Synchronous** functions will return requested data immediately in the function's return value. If the requested data is not available immediately, synchronous calls will generally block until a specified wait time has elapsed.

Most calls that wait for input from the user will run in the asynchronous mode.

3.1 getSDKVersion

This function retrieves the SDK revision number.

```
(NSString *) getSDKVersion
```

Parameters: None

Return Value: String containing the SDK revision number.

3.2 openDevice

This function opens a connection to the device. After calling this function, call **isDeviceOpened** to make sure the device was successfully opened.

```
(BOOL) openDevice
```

Parameters: None

Return Value:
YES = Success
NO = Error

3.3 closeDevice

This function closes the connection to the currently opened device. After calling this function, call **isDeviceOpened** to make sure the device was successfully closed.

```
(BOOL) closeDevice
```

Parameters: None

Return Value:
YES = Success
NO = Error

3.4 isDeviceConnected

This function reports whether any compatible devices are connected to the host.

```
(BOOL) isDeviceConnected
```

Parameters: None

Return Value:
YES = host is connected to a device
NO = host is not connected to a device

3.5 isDeviceOpened

This function retrieves device opened status, which changes on successful completion of a call to **openDevice** or **closeDevice**.

```
(BOOL) isDeviceOpened
```

Parameters: None

Return Value:
YES = Device is opened
NO = Device is not opened

3.6 sendCommandToDevice

This function sends a direct command to device. See ***D99875475 MagneSafe V5 Programmer's Reference (Commands)*** for details about available commands and syntax.

```
(Void *) sendCommandToDevice:(NSString *)pData
```

Parameters:

Parameter	Description
pData	Command to send to the device. For example, pass command string "C10206C20503C30100" to call the Discovery command.

Return Value: None

3.7 getResponseData

This function retrieves card data from a string separated by '|' after a cardholder swipes a card. The host software should call it in response to the **trackDataReadyNotification** callback.

```
(NSString *) getResponseData
```

Parameters: None

Return Value:
A null terminated hex string for Card Data, Field separated by '|'.NULL value for failed.
Fields:
Device ID, Device Serial Number, Card Swipe Status, CardEncode Type, Track 1 Decode Status, Track 2 Decode Status, Track 3 Decode Status, MagnePrint Status, Track 1 Length, Track 2 Length, Track 3 Length, Masked Track 1 Length, Masked Track 2 Length, Masked Track 3 Length, MagnePrint Length, Card Data, Masked Card Data, DUKPT Session ID, DUKPT Key Serial Number, First Name, Last Name, PAN, Month, Year, Track 1 Data, Track 2 Data, Track 3 Data, Masked Track 1 Data, Masked Track 2 Data, Masked Track 3 Data, MagnePrint Data

3.8 clearBuffers

This function clears the SDK library's local cache of card swipe data.

```
(void) clearBuffers
```

Parameters: None

Return Value: None

3.9 listenForEvents

This function sets a callback function to notify when the device has card data to send to the host or when the device state changes. See example in **Open Device** code example.

```
(void) listenForEvents:(UInt32)event
```

Parameters: Event
  TRANS_EVENT_OK = Transaction succeeded.
  TRANS_EVENT_START = Reader started sending data.
  TRANS_EVENT_ERROR = Reader failed sending data.

Return Value: None

3.10 getMaskedTracks [iDynamo/uDynamo Only]

This function retrieves masked card track data after a cardholder swipes a card. Only available on iDynamo/uDynamo; other devices will return an empty string.

```
(NSString *) getMaskedTracks
```

Parameters: None

Return Value:
Return stored masked track data string. Tracks are delimited with start and end sentinels.

3.11 getTrack1Masked

This function retrieves masked track 1 data after a cardholder swipes a card.

```
(NSString *) getTrack1Masked
```

Parameters: None

Return Value: Return stored masked track1 data string.

3.12 getTrack2Masked

This function retrieves masked track 2 data, if any, after a cardholder swipes a card.

```
(NSString *) getTrack2Masked
```

Parameters: None

Return Value: Return stored masked track2 data string.

3.13 getTrack3Masked

This function retrieves masked track 3 data, if any, after a cardholder swipes a card.

```
(NSString *) getTrack3Masked
```

Parameters: None

Return Value: Return stored masked track3 data string.

3.14 getTrack1

This function retrieves the card's track 1 data in encrypted format after a cardholder swipes a card.

```
(NSString *) getTrack1
```

Parameters: None

Return Value: String containing encrypted track 1 data.

3.15 getTrack2

This function retrieves the card's track 2 data in encrypted format, if any, after a cardholder swipes a card.

```
(NSString *) getTrack2
```

Parameters: None

Return Value: String containing encrypted track 2 data.

3.16 getTrack3

This function retrieves the card's track 3 data in encrypted format, if any, after a cardholder swipes a card.

```
(NSString *) getTrack3
```

Parameters: None

Return Value: String containing encrypted track 3 data.

3.17 getMagnePrint

This function retrieves the card's encrypted MagnePrint, for readers that support MagnePrint.

```
(NSString *) getMagnePrint
```

Parameters: None

Return Value: String containing the card's encrypted MagnePrint.

3.18 getMagnePrintStatus [iDynamo Only]

This function retrieves the card MagnePrint status. For more information, see ***D99875475***. Only available on iDynamo; it will return an empty string in audio reader.

```
(NSString *) getMagnePrintStatus
```

Parameters: None

Return Value:
Return stored MagnePrintStatus string.

3.19 getDeviceSerial

This function retrieves the device serial number.

```
(NSString *) getDeviceSerial
```

Parameters: None

Return Value: String containing the device serial number.

3.20 getMagTekDeviceSerial

This function returns the MagTek serial number of the currently opened device.

```
(NSString *) getMagTekDeviceSerial
```

Parameters: None

Return Value: Return stored serial number created by MagTek.

3.21 getSessionID [iDynamo/uDynamo Only]

This function retrieves the Session ID from the currently opened device, which the host can use to uniquely identify a transaction to prevent replay. Only supported by iDynamo/uDynamo; on other devices this function will return an empty string. For more information, see ***D99875475***

```
(NSString *) getSessionID
```

Parameters: None

Return Value: Stored session ID.

3.22 getKSN

This function retrieves the Key Serial Number (KSN) from the device.

```
(NSString *) getKSN
```

Parameters: None

Return Value: String containing the stored key serial number.

3.23 getDeviceName

This function gets the device’s product name.

```
(NSString *) getDeviceName
```

Parameters: None

Return Value: String containing the device product name.

3.24 getDeviceType

This function gets the device type.

```
(int) getDeviceType
```

Parameters: None

Return Value: Device Type

3.25 setDeviceType

This function sets the type of device to open. Call this function before calling openDevice.

```
(void) setDeviceType:(UInt32 *)deviceType
```

Parameters:
Device Type:
  MAGTEKAUDIOREADER = Audio readers aDynamo, uDynamo.
  MAGTEKIDYNAMO = iOS 30-pin and Lightning readers iDynamo.
  MAGTEKDYNAMAX = BLE reader DynaMAX.

Return Value: None

3.26 getDeviceCaps

This function gets the capabilities of the currently opened device.

```
(NSString *) getDeviceCaps
```

Parameters: None

Return Value: Return device capabilities.
  CAP_MASKING = 1,
  CAP_ENCRYPTION=2,
  CAP_CARD_AUTH = 4,
  CAP_DEVICE_AUTH = 8,
  CAP_SESSION_ID = 16,
  CAP_DISCOVERY= 32,

3.27 getCapMSR

This function gets the MSR capability of the device. For more information, see ***D99875483*** – Track ID Enable Property.

```
(NSString *) getCapMSR
```

Parameters: None

Return Value:
Return MSR Capability bit masking.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Id	0	T_3	T_3	T_2	T_2	T_1	T_1

Id	0 – Decodes standard ISO/ABA cards only 1 – Decodes AAMV and 7-bit cards also If this flag is set to 0, only tracks that conform to the ISO format allowed for that track will be decoded. If the track cannot be decoded by the ISO method it will be considered to be in error.
T#	00 – Track Disabled 01 – Track Enabled 10 – Track Enabled/Required (Error if blank)

3.28 getCapMagStripeEncryption

This function gets the device's capability for encrypting track data.

```
(NSString *) getCapMagStripeEncryption
```

Parameters: None

Return Value:
"1" = Available
"0" = Unavailable.

3.29 getCapTracks

This function gets information about the device's tracks capability.

```
(NSString *) getCapTracks
```

Parameters: None

Return Value: A hex string for the track capability. See Track ID Enable Property in ***D99875475***.

3.30 getCardExpDate

This function retrieves the card expiration date after a cardholder swipes a card.

```
(NSString *) getCardExpDate
```

Parameters: None

Return Value: String containing the card expiration date

3.31 getCardLast4

This function gets the last 4 digits of the card account number (PAN) after a cardholder swipes a card.

```
(NSString *) getCardLast4
```

Parameters: None

Return Value: String containing the last 4 digits of the PAN

3.32 getCardIIN

This function gets the issuer identification number (IIN) of the card number after a cardholder swipes a card.

```
(NSString *) getCardIIN
```

Parameters: None

Return Value: String containing the IIN

3.33 getCardName

This function gets the cardholder name after a cardholder swipes a card.

```
(NSString *) getCardName
```

Parameters: None

Return Value: String containing the cardholder name, for example, “John Wayne”.

3.34 getCardPANLength

This function gets the length of the PAN after a cardholder swipes a card.

```
(int) getCardPANLength
```

Parameters: None

Return Value: Length of card number or PAN

3.35 getCardServiceCode

This function retrieves the card’s service code after a cardholder swipes a card.

```
(NSString *) getCardServiceCode
```

Parameters: None

Return Value: String containing the card’s service code

3.36 getFirmware

This function retrieves the part number and revision of the device’s firmware.

```
(NSString *) getFirmware
```

Parameters: None

Return Value: String containing firmware part number and revision.

3.37 getTrackDecodeStatus

This function retrieves the track decode status after a cardholder swipes a card.

```
(NSString *) getTrackDecodeStatus
```

Parameters: None

Return Value:
Hex string, each 2 digits represent one track's decode status, where the left most 2 digits for Track 1.
"00" = success
"01" = Error or not Decodable
"02" = No track present.
Example:
"000000" = Track 1, 2, and 3 success.
"000100" = Track 1 and 3 success. Track 2 had error.
"000002" = Track 1 and 2 success. Track 3 not present.

3.38 getBatteryLevel

This function retrieves device's battery level percentage between 0% and 100%, if the device has a battery and supports battery level monitoring.

```
(long) getBatteryLevel
```

Parameters: None

Return Value: Long value between 0 and 100

3.39 getDevicePartNumber

This function returns the currently opened device's part number.

```
(NSString *) getDevicePartNumber
```

Parameters: None

Return Value: String containing the device part number.

3.40 getCardStatus

Retrieves the Card Status

```
(NSString *) getCardStatus
```

Parameters: None

Return Value: Card Status, which depends on the device.

3.41 getTagValue [aDynamo/uDynamo Only]

This function retrieves individual TLV tag values. Only supported on aDynamo/UDynamo.

```
(NSString *) getTagValue:(UInt32)tag
```

Parameters:
Tag = An `MTSCRATransactionData` type (see section **A.4 MTSCRATransactionData**).

Return Value: String containing the value of the specified tag.

3.42 getDeviceStatus

This function gets the status of the currently connected device.

```
(NSString *) getDeviceStatus
```

Parameters: None

Return Value: Return device status of swipe count and battery level.

3.43 getOperationStatus

This function gets the status of the current operation.

```
(NSString *) getOperationStatus
```

Parameters: None

Return Value: Operation Status

3.44 getTLVVersion

This function returns the version of the tag-length-value (TLV) format supported by the device.

```
(NSString *) getTLVVersion
```

Parameters: None

Return Value: String containing the firmware TLV version.

3.45 setDeviceProtocolString [iDynamo Only]

This function sets the protocol string the SDK will use to communicate with the device. See example in **Open Device** code example.

```
(void) setDeviceProtocolString:(NSString *)pData
```

Parameters: Protocol String

Return Value: None

3.46 getResponseType

This function gets the response type.

```
(NSString *) getResponseType
```

Parameters: None

Return Value: Response Type

3.47 setUUIDString [DynaMAX Only]

This function sets the UUIDString for the BLE connection.

```
(void) setUUIDString:(NSString *)uuidString
```

Parameters: UUID String of the device

Return Value: None

3.48 getConnectedPeripheral [DynaMAX Only]

This function gets the current connected peripheral (device).

```
(NSString *) getConnectedPeripheral
```

Parameters: None

Return value: Current connected device

3.49 getDiscoveredPeripherals [DynaMAX Only]

This function gets an array of DynaMAX devices connected to the host.

```
(NSMutableArray *) getDiscoveredPeripherals
```

Parameters: None

Return Value: Array of DynaMAX devices.

4 MTSCRA Delegate Methods

After issuing the methods in section **3 MTSCRA Functions**, the MTSCRA SDK libraries will call these Delegate methods (callback functions) to provide the requested data and / or a detailed response. Further information about the data received by these functions can be found in these documents:

- For iDynamo and uDynamo: ***D99875475 MagneSafe V5 Programmer's Reference (Commands)***

For details about registering Delegate methods, see the demo application included with the SDK.

4.1 trackDataReadyNotification

The SDK sends this notification when card data is available from the device.

4.2 devConnectionNotification

The SDK sends this notification when the connection status of the device changes.

4.3 onDataReceived

Return a card object type with card swipe data.

4.4 cardSwipeDidStart

Card swipe has started

4.5 cardSwipeDidGetTransError

Card swipe got an error during transmission.

4.6 onDeviceConnectionDidChange

Device connection changed whether from close to open or vice versa.

4.7 bleReaderConnected

DynaMAX Reader was connected.

4.8 bleReaderDidDiscoverPeripheral

DynaMAX Reader was discovered.

4.9 bleReaderStateUpdated

DynaMAX State did change.

Appendix A Enums

A.1 MTSCRADeviceType

MAGTEKAUDIOREADER = Audio readers aDynamo, uDynamo.
MAGTEKIDYNAMO = iOS 30-pin and Lightning readers iDynamo.
MAGTEKDYNAMAX = BLE reader DynaMAX.

A.2 MTSCRATransactionStatus

TRANS_STATUS_OK = Transaction succeeded.
TRANS_STATUS_START = Reader started sending data.
TRANS_STATUS_ERROR = Reader failed sending data.

A.3 MTSCRATransactionEvent

TRANS_EVENT_OK = Transaction succeeded.
TRANS_EVENT_START = Reader started sending data.
TRANS_EVENT_ERROR = Reader failed sending data.

A.4 MTSCRATransactionData

TLV_OPSTS = Operation Status
TLV_CARDSTS = Card Information
TLV_TRACKSTS = Card tracks status
TLV_CARDNAME = Cardholder name
TLV_CARDIIN = Card issuer identification number
TLV_CARDLAST4 = Last four digits of PAN number
TLV_CARDEXPDATE = Card Expiration date
TLV_CARDSVCCODE = Card service code
TLV_CARDPANLEN = Length of the PAN
TLV_ENCTK1 = Encrypted track 1
TLV_ENCTK2 = Encrypted track 2
TLV_ENCTK3 = Encrypted track 3
TLV_DEVSN = Device serial number
TLV_DEVSNMAGTEK = Device serial number created by MagTek
TLV_DEVFW = Device firmware version
TLV_DEVNAME = Device model name
TLV_DEVCAPS = Device capabilities
TLV_DEVSTATUS = Device status
TLV_TLVVERSION = Firmware TLV version
TLV_DEVPARTNUMBER = Device part number
TLV_CAPMSR = Magstripe capabilities
TLV_CAPTRACKS = Track capabilities
TLV_CAPMAGSTRIPEENCRYPTION = Magstripe encryption capabilities
TLV_KSN = KSN
TLV_CMAC = CMAC
TLV_SWPCOUNT = Swipe count
TLV_BATTLEVEL = Batter level
TLV_CFGTLVVERSION = TLV version
TLV_CFGDISCOVERY = Discovery

TLV_CFGCARDNAME = Card name
TLV_CFGCARDIIN = Card issuer identification number
TLV_CFGCARDLAST4 = Card last 4 PAN
TLV_CFGCARDEXPDATE = Card expiration date
TLV_CFGCARDSVCCODE = Card service code
TLV_CFGCARDPANLEN = Card PAN length
TLV_MSKTK1 = Masked Track 1
TLV_MSKTK2 = Masked Track 2
TLV_MSKTK3 = Masked Track 3
TLV_HASHCODE = Hash code
TLV_SESSIONID = Session ID
TLV_MAGNEPRINT = MagnePrint
TLV_MAGNEPRINT_STS = MagnePrint status

Appendix B Troubleshooting

To troubleshoot runtime issues with custom software, use standard XCode debugging methods and tools.

Appendix C Code Examples

C.1 Open Device

```
self.mtSCRALib = [[MTSCRA alloc] init];
[self.mtSCRALib
listenForEvents:(TRANS_EVENT_OK|TRANS_EVENT_START|TRANS_EVENT_ERROR)];

//iDynamo
[self.mtSCRALib setDeviceType:(MAGTEKIDYNAMO)];
[self.mtSCRALib setDeviceProtocolString:("com.magtek.idynamo")];
[self.mtSCRALib setDeviceType:(MAGTEKIDYNAMO)];

//Audio
//[self.mtSCRALib setDeviceType:(MAGTEKAUDIOREADER)];
[self.mtSCRALib openDevice];
```

C.2 Close Device

```
[self.mtSCRALib closeDevice];
```

C.3 Get Tracks Data From Reader

```
[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(trackDataReady:) name:@"trackDataReadyNotification"
object:nil];

- (void)trackDataReady:(NSNotification *)notification
{
      NSNumber *status = [[notification userInfo]
valueForKey:@"status"];
      [self performSelectorOnMainThread:@selector(onDataEvent:)
withObject:status waitUntilDone:YES];
}

- (void)onDataEvent:(id)status
{
      //[self clearLabels];
      switch ([status intValue]) {

            case TRANS_STATUS_OK:
                  NSLog(@"TRANS_STATUS_OK");
                  break;

            case TRANS_STATUS_ERROR:
                  NSLog(@"TRANS_STATUS_ERROR");
                  break;

            default:
                  break;
      }
}
```

C.4 Get Connection Status Of Reader

```
 [[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(devConnStatusChange)
name:@"devConnectionNotification" object:nil];

- (void)devConnStatusChange
{
      BOOL isDeviceConnected = [self.mtSCRALib isDeviceConnected];
      if (isDeviceConnected)
      {
            self.deviceStatus.text = @"Device Connected";
      }
      else
      {
            self.deviceStatus.text = @"Device Disconnected";
      }
}
```