MAGTEK®
SECURITY FROM THE INSIDE

DynaMag, DynaMAX, DynaWave eDynamo, mDynamo, uDynamo, BulleT, iDynamo 6, tDynamo

Secure Card Reader Authenticators Programmer's Manual (macOS)

October 2022

Manual Part Number:
D998200161-12

REGISTERED TO ISO 9001:2015

Table 0.1 – Revisions

Rev Number	Date	Notes
10	February 7, 2017	Initial Release
11	June 9, 2017	Fix table in section **4.10** that provides values for card events; misc. format fixes
12	October 19, 2022	Updated to support iDynamo 6. Updated enumerations. Added details for setting Date and Time before starting an EMV transaction.

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.

DEMO SOFTWARE / SAMPLE CODE: Unless otherwise stated, all demo software and sample code are to be used by Licensee for demonstration purposes only and MAY NOT BE incorporated into any production or live environment. The PIN Pad sample implementation is for software PIN Pad test purposes only and is not PCI compliant. To meet PCI compliance in production or live environments, a third-party PCI compliant component (hardware or software-based) must be used.

Table of Contents

1 Introduction

This document provides instructions for software developers who want to create custom software solutions that communicate with DynaMag, DynaMAX, DynaWave, eDynamo, mDynamo, uDynamo, BulleT, iDynamo 6, or tDynamo connected to a macOS host via USB or BLE. This document is part of a larger library of documents designed to assist MagTek device implementers.

The following documents are essential:

- ***D99875475 MagneSafe V5 Programmer's Reference (Commands)***
- ***D998200176 DYNAMAG / MAGNESAFE V5 INTELLIHEAD USB / MAGNESAFE V5 READERS USB PROGRAMMER'S MANUAL (COMMANDS)***
- ***D998200175 DYNAMAX PROGRAMMER'S MANUAL (COMMANDS)***
- ***D998200215 DYNAWAVE PROGRAMMER'S MANUAL (COMMANDS)***
- ***D998200151 MDYNAMO PROGRAMMER'S MANUAL (COMMANDS)***
- ***D998200324 IDYNAMO 6 PROGRAMMER'S MANUAL (COMMANDS)***
- ***D998200115 EDYNAMO PROGRAMMER'S MANUAL (COMMANDS)***
- ***D998200151 MDYNAMO PROGRAMMER'S MANUAL (COMMANDS)***
- ***D998200226 TDYNAMO PROGRAMMER'S MANUAL (COMMANDS)***

1.1 About MTSCRA Demo

The MTSCRA Demo software, available from MagTek, provides demonstration source code and a reusable MTSCRA library that provides developers of custom macOS software solutions with an easy-to-use interface for MagTek devices. 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 About MTSCRA OEM Demo

The MTSCRA Demo software, available from MagTek, provides demonstration source code and a reusable MTSCRA library that provides developers of custom macOS software solutions with an easy-to-use interface for mDynamo. 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.

1.3 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 MagTek 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 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**.

1.4 SDK Contents

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

1.5 System Requirements

Tested devices:

- Apple Mac

Tested operating systems: macOS 12 and above.

Build Platforms: xCode 13, xCode 14

1.6 Interfaces for Operating Systems

The following table matches the device interface to operating system.

Device	Interface	Operating System
eDynamo	BLE 4.0/USB	macOS 12 and above
uDynamo	USB	macOS 12 and above
DynaMAX	BLE 4.0/USB	macOS 12 and above
Dynamag	USB	macOS 12 and above
DynaWave	USB	macOS 12 and above
mDynamo	USB	macOS 12 and above
BulleT	USB	macOS 12 and above
tDynamo	BLE 4.2/USB	macOS 12 and above
iDynamo 6	USB	macOS 12 and above

2 How to Set Up the MTSCRA SDK

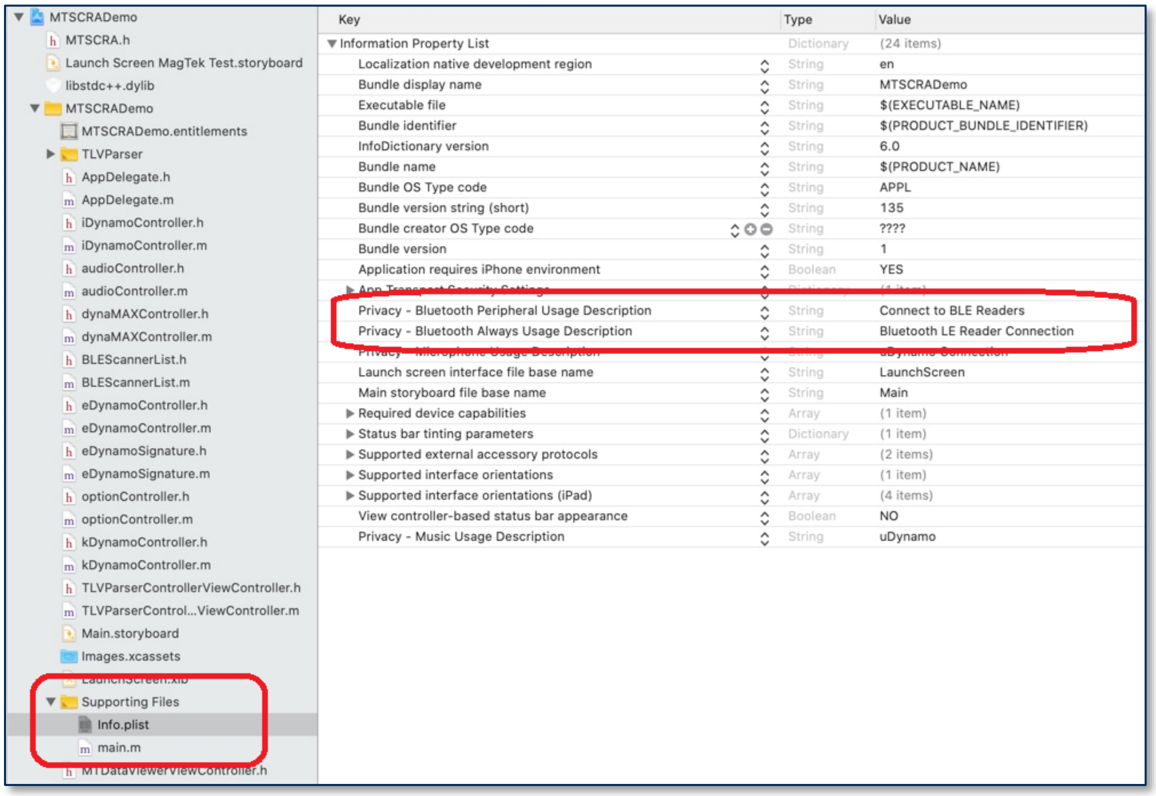
2.1 How to Set Up the X Code Development Environment

To add the MTSCRAOSX 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) `libMTSCRAOSX.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**.

2.2 Important Information About Bluetooth LE

1) When calling functions **startScanningForPeripherals** or Error! Reference source not found., the application should make sure Error! Reference source not found. has received a device status and that the most recent status was **OK**, otherwise the device will not be able to connect, and macOS will not throw any error if Bluetooth is not ready.
2) In macOS, app projects must specify the Privacy Usage Description for Bluetooth by including **NSBluetoothAlwaysUsageDescription** in the **info.plist** file. Accessing Core Bluetooth without the usage descriptions will cause a runtime crash. For backward compatibility with older versions of macOS, define **NSBluetoothPeripheralUsageDescription** as well.



```
▼ MTSCRADemo
    MTSCRA.h
    Launch Screen MagTek Test.storyboard
    libstdc++.dylib
  ▼ MTSCRADemo
      MTSCRADemo.entitlements
    ▶ TLVParser
      AppDelegate.h
      AppDelegate.m
      iDynamoController.h
      iDynamoController.m
      audioController.h
      audioController.m
      dynaMAXController.h
      dynaMAXController.m
      BLEScannerList.h
      BLEScannerList.m
      eDynamoController.h
      eDynamoController.m
      eDynamoSignature.h
      eDynamoSignature.m
      optionController.h
      optionController.m
      kDynamoController.h
      kDynamoController.m
      TLVParserControllerViewController.h
      TLVParserControl...ViewController.m
      Main.storyboard
      Images.xcassets
      LaunchScreen.xib
    ▼ Supporting Files
        Info.plist
        main.m
      MTDataViewerViewController.h
```

Key	Type	Value
▼ Information Property List	Dictionary	(24 items)
Localization native development region	String	en
Bundle display name	String	MTSCRADemo
Executable file	String	$(EXECUTABLE_NAME)
Bundle identifier	String	$(PRODUCT_BUNDLE_IDENTIFIER)
InfoDictionary version	String	6.0
Bundle name	String	$(PRODUCT_NAME)
Bundle OS Type code	String	APPL
Bundle version string (short)	String	135
Bundle creator OS Type code	String	????
Bundle version	String	1
Application requires iPhone environment	Boolean	YES
▶ App Transport Security Settings	Dictionary	(1 item)
Privacy - Bluetooth Peripheral Usage Description	String	Connect to BLE Readers
Privacy - Bluetooth Always Usage Description	String	Bluetooth LE Reader Connection
Privacy - Microphone Usage Description	String	uDynamo Connection
Launch screen interface file base name	String	LaunchScreen
Main storyboard file base name	String	Main
▶ Required device capabilities	Array	(1 item)
▶ Status bar tinting parameters	Dictionary	(1 item)
▶ Supported external accessory protocols	Array	(2 items)
▶ Supported interface orientations	Array	(1 item)
▶ Supported interface orientations (iPad)	Array	(4 items)
View controller-based status bar appearance	Boolean	NO
Privacy - Music Usage Description	String	uDynamo

2.3 Important Information About USB Dynamag

In certain Intel based macOS systems a driver needs to be removed if you cannot scan for the device.

Instructions:

1) Go to folder /Library/Extension.
2) Remove file com.magtek.mtscra.kext.
3) Reboot your system.

3 MTSCRA Functions

To develop an macOS 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 MTSCRA Demo 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 Method**.
- **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 startScanningForPeripherals

This function retrieves a list of available Bluetooth LE devices. After calling this function to locate the device you wish to connect to, use Error! Reference source not found. to tell the library which device you want to connect to. Use **stopScanningForPeripherals** to stop the scan.

```
(void)startScanningForPeripherals
```

Parameters: None

Return Value: An array of peripherals

3.3 stopScanningForPeripherals

This function stops the scanning of available Bluetooth LE devices.

```
(void)stopScanningForPeripherals
```

Parameters: None

Return Value: None

3.4 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

3.5 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

3.6 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

3.7 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

After opened is a good time to set the device's date and time if needed. The device's system date and time must be set before starting an EMV transaction. This is done at the factory for devices with a battery-backed real time clock. Otherwise, the host software must set the date and time every time the device is power cycled or reset. Use sendExtendedCommand to send the **Extended Command 0x030C - Set Date and Time**. See the appropriate document for details for **0x030C - Set Data and Time**:

- ***D998200324 IDYNAMO 6 PROGRAMMER'S MANUAL (COMMANDS)***
- ***D998200115 EDYNAMO PROGRAMMER'S MANUAL (COMMANDS)***
- ***D998200151 MDYNAMO PROGRAMMER'S MANUAL (COMMANDS)***

- ***D998200226 TDYNAMO PROGRAMMER'S MANUAL (COMMANDS)***
- ***D998200215 DYNAWAVE PROGRAMMER'S MANUAL (COMMANDS)***

Sample code for setting date and time:

```
- (void)setDateTime
{
    NSDate *date = [NSDate date];
    NSCalendar *calendar = [NSCalendar currentCalendar];
    // year, offset from 2008 (value from 00 to ff)
    NSInteger year = [calendar component:NSCalendarUnitYear fromDate:date] - 2008;
    // month, value from 1 to 12
    NSInteger month = [calendar component:NSCalendarUnitMonth fromDate:date];
    // day, value from 1 to 31
    NSInteger day = [calendar component:NSCalendarUnitDay fromDate:date];
    // hour, value from 0 to 23
    NSInteger hour = [calendar component:NSCalendarUnitHour fromDate:date];
    // minute, value from 0 to 59
    NSInteger minute = [calendar component:NSCalendarUnitMinute fromDate:date];
    // second, value from 0 to 59
    NSInteger second = [calendar component:NSCalendarUnitSecond fromDate:date];

    NSString* cmd = @"030C";
    NSString* size = @"0018";
    NSString* macType = @"00";
    NSString* deviceSn = @"00000000000000000000000000000000";
    NSString* strMonth = [NSString stringWithFormat:@"%02lX", (long)month];
    NSString* strDay = [NSString stringWithFormat:@"%02lX", (long)day];
    NSString* strHour = [NSString stringWithFormat:@"%02lX", (long)hour];
    NSString* strMinute = [NSString stringWithFormat:@"%02lX", (long)minute];
    NSString* strSecond = [NSString stringWithFormat:@"%02lX", (long)second];
    NSString* unused = @"00";
    NSString* strYear = [NSString stringWithFormat:@"%02lX", (long)year];
    NSString* commandToSend = [NSString stringWithFormat:@"%@%@%@%@%@%@%@%@%@%@%@",
cmd, size, macType, deviceSn, strMonth, strDay, strHour, strMinute, strSecond, unused,
strYear];

    [self.scra sendExtendedCommand:commandToSend];
}
```

3.8 sendCommandToDevice

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

```
(int) 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:
 0 = Success
 9 = Error
15 = Busy

3.9 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

3.10 clearBuffers

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

```
(void) clearBuffers
```

Parameters: None

Return Value: None

3.11 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

3.12 getMaskedTracks

This function retrieves masked card track data after a cardholder swipes a card. Only available on 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.

3.13 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.

3.14 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.

3.15 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.

3.16 getCardPAN

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

```
(NSString *) getCardPAN
```

Parameters: None

Return Value: Return stored masked PAN data string.

3.17 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.

3.18 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.

3.19 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.

3.20 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.

3.21 getMagnePrintStatus

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

```
(NSString *) getMagnePrintStatus
```

Parameters: None

Return Value:
Return stored MagnePrintStatus string.

3.22 getDeviceSerial

This function retrieves the device serial number.

```
(NSString *) getDeviceSerial
```

Parameters: None

Return Value: String containing the device serial number.

3.23 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.

3.24 getSessionID

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 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.

3.25 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.

3.26 getDeviceName

This function gets the device's product name.

```
(NSString *) getDeviceName
```

Parameters: None

Return Value: String containing the device product name.

3.27 getDeviceType

This function gets the device type.

```
(int) getDeviceType
```

Parameters: None

Return Value: Device Type

3.28 setDeviceType

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

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

Parameters:
Device Type:
  MAGTEKDYNAMAX = BLE reader DynaMAX.
  MAGTEKEDYNAMO = BLE reader eDynamo.
  MAGTEKUSBMSR = USB reader uDynamo.

Return Value: None

3.29 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,

3.30 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)

3.31 getCapMagStripeEncryption

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

```
(NSString *) getCapMagStripeEncryption
```

Parameters: None

Return Value:
 "1" = Available

"0" = Unavailable.

3.32 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***.

3.33 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

3.34 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

3.35 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

3.36 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".

3.37 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

3.38 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

3.39 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.

3.40 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 are 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.

3.41 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

3.42 getDevicePartNumber

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

```
(NSString *) getDevicePartNumber
```

Parameters: None

Return Value: String containing the device part number.

3.43 getCardStatus

Retrieves the Card Status

```
(NSString *) getCardStatus
```

Parameters: None

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

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

3.44 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.

3.45 getOperationStatus

This function gets the status of the current operation.

```
(NSString *) getOperationStatus
```

Parameters: None

Return Value: Operation Status

3.46 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.

3.47 getResponseType

This function gets the response type.

```
(NSString *) getResponseType
```

Parameters: None

Return Value: Response Type

3.48 setUUIDString [DynaMAX/eDynamo Only]

This function sets the UUIDString for the BLE connection.

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

Parameters: UUID String of the device

Return Value: None

3.49 getConnectedPeripheral [DynaMAX/eDynamo BLE Only]

This function gets the current connected peripheral (device).

```
(NSString *) getConnectedPeripheral
```

Parameters: None

Return value: Current connected device

3.50 getDiscoveredPeripherals [DynaMAX/eDynamo BLE Only]

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

```
(NSMutableArray *) getDiscoveredPeripherals
```

Parameters: None

Return Value: Array of DynaMAX/eDynamo devices.

3.51 startTransaction (EMV Device Only)

This function start an EMV Transaction. The device's system date and time must be set before starting an EMV transaction. See **isDeviceOpened** for details and sample code.

```
(int) startTransaction:
(Byte)timeLimit cardType:(Byte)cardType option:(Byte)option
amount:(Byte*)amount transactionType:(Byte)transactionType
cashBack:(Byte*)cashBack currencyCode:(Byte*)currencyCode
reportingOption:(Byte)reportingOption
```

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 (not supported at this time) 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 (not used on this reader) 0x02 = Force Online (not used on this reader) 0x04 = Acquirer not available (Note: prevents long timeout on waiting for host approval) (causes “decline” to be generated internally if ARQC is generated)
amount	Amount Authorized (EMV Tag 9F02, format n12, 6 bytes) in hex string For example: “000000000999”, means 9.99 dollars.
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

Parameter	Description
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

3.52 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.

```
(int) setUserSelectionResult:(Byte)status selection:(Byte)selection;
```

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
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

3.53 cancelTransaction

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

```
(int) cancelTransaction
```

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

The status of this function will be returned in the delegate method **onEMVCommandResult** (EMV Device Only).

Parameter	Description
status	**Result codes**: 0x0000 = Success, the transaction was cancelled 0x038D = Failure, no transaction currently in progress 0x038F = Failure, transaction in progress, card already inserted

3.54 setAcquirerResponse (EMV Device Only)

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

```
(int) setAcquirerResponse:(Byte*)response length:(int)length
```

Parameter	Description
response	See **4.17Appendix E** Hex string for the response data following TLV response message.
length	Two byte binary, most significant byte first. This gives the total length of the Acquirer Response message that follows.

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

3.55 sendExtendedCommand (EMVDevice Only)

Send extended command to device.

```
(int) sendExtendedCommand:(NSString *)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

3.56 requestDeviceList

Start request to retrieve Device list from BLE or USB.

```
(void) requestDeviceList:(UInt32 *)type
```

Parameters:

Parameter	Description
type	MTSCRADeviceType for which the device will be requested.

Return value: Void

3.57 setAddress

Address of desired device to connect to.

```
(void) setAddress:(NSString *)address
```

Parameters:

Parameter	Description
address	NSString of address.

Return value: Void

3.58 getProductID

Mac USB only. Get current connected Product ID.

```
(int) getProdID
```

Return value: Integer of product ID.

4 MTSCRA Delegate Method

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:

- ***D99875475 MagneSafe V5 Programmer's Reference (Commands)***
- ***D998200176 DYNAMAG / MAGNESAFE V5 INTELLIHEAD USB / MAGNESAFE V5 READERS USB PROGRAMMER'S MANUAL (COMMANDS)***
- ***D998200175 DYNAMAX PROGRAMMER'S MANUAL (COMMANDS)***
- ***D998200215 DYNAWAVE PROGRAMMER'S MANUAL (COMMANDS)***
- ***D998200151 MDYNAMO PROGRAMMER'S MANUAL (COMMANDS)***
- ***D998200324 IDYNAMO 6 PROGRAMMER'S MANUAL (COMMANDS)***
- ***D998200115 EDYNAMO PROGRAMMER'S MANUAL (COMMANDS)***
- ***D998200151 MDYNAMO PROGRAMMER'S MANUAL (COMMANDS)***
- ***D998200226 TDYNAMO PROGRAMMER'S MANUAL (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

BLE Reader was connected.

4.8 bleReaderDidDiscoverPeripheral

BLE Reader was discovered.

4.9 bleReaderStateUpdated

BLE State did change.

4.10 onTransactionStatus (EMV Device Only)

The SDK sends this notification when the transaction status has changed.

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

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 • 0x15 – transaction canceled by MSR Swipe

3-4	Final Status	TBD

4.11 onDisplayMessageRequest (EMV Device Only)

Device request for displaying information to user.

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

4.12 onUserSelectionRequest (EMV Device Only)

Device request for application to display a User Selection Menu.

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.

4.13 onARQCReceived (EMV Device Only)

This notification is sent from the device for ARQC data.

0	Message Length	Two byte binary, most significant byte first. This gives the total length of the ARQC message that follows.
2	ARQC Message	See **4.17Appendix D**. It is expected that the host will use this data to process a request.

4.14 onTransactionResult (EMV Device Only)

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

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 ARQC message that follows.
3	Batch Data	See **4.17Appendix F**. It is expected that the host will save this data as a record of the transaction.

4.15 onEMVCommandResult (EMV Device Only)

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

Result Code Description
• 0x0000 = Success, the transaction process has been started • 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 in progress, card already inserted

4.16 onDeviceExtendedResponseReceived

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

Parameter	Description
Command	Hexadecimal string 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 the following data in bytes.

4.17 deviceNotPaired

This message occurs when a command is sent to an unpaired BLE device.

Appendix A Enums

A.1 MTSCRADeviceType

MAGTEKAUDIOREADER //iOS only
MAGTEKIDYNAMO
MAGTEKDYNAMAX = BLE reader DynaMAX.
MAGTEKEDYNAMO = BLE Reader eDynamo
MAGTEKUSBMSR = USB reader uDynamo //macOS only
MAGTEKKDYNAMO
MAGTEKTDYNAMO
MAGTEKDYNAWAVE
MAGTEKMDYNAMO
MAGTEKNON

A.2 ConnectionType

BLE
BLE_EMV
USB
Lightning
NONE

A.3 MTSCRACapabilities

CAP_MASKING
CAP_ENCRYPTION
CAP_CARD_AUTH
CAP_DEVICE_AUTH
CAP_SESSION_ID
CAP_DISCOVERY

A.4 MTSCRATransactionStatus

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

A.5 MTSCRATransactionEvent

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

A.6 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

A.7 MTSCRABLEState

OK
OFF
RESETTING
DISCONNECTED
UNSUPPORTED
UNAUTHORIZED
UNKNOWNBLE

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

```
// To open device
// step 1. setDeviceType
// step 2. setConnectionType
// step 3. setDeviceAddress
// step 4. openDevice
// step 5. a connected or disconnected will call to
onDeviceConnectionDidChange

    self.scra.delegate = self;

    if([self.sltConnectionType.selectedItem.title
isEqualToString:@"Bluetooth LE"])
    {
        [self.scra setDeviceType:MAGTEKDYNAMAX];
    }

    else if([self.sltConnectionType.selectedItem.title
isEqualToString:@"Bluetooth LE EMV"])
    {
        [self.scra setDeviceType:MAGTEKEDYNAMO];
    }

    else if([self.sltConnectionType.selectedItem.title
isEqualToString:@"Bluetooth LE EMVT"])
    {
        [self.scra setDeviceType:MAGTEKTDYNAMO];
    }

    else
    {
        [self.scra setDeviceType:MAGTEKUSBMSR];
    }

    [self.scra setConnectionType:device.connectionType];
    [self.scra setAddress:device.Address];


    dispatch_after(dispatch_time(DISPATCH_TIME_NOW, .5f *
NSEC_PER_SEC),

                   dispatch_get_main_queue(), ^{

        [self.scra 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";
```

```
            }
}
```

Appendix D ARQC Message Format

This section gives the format of the ARQC Message delivered in the ARQC Message notification. The output is controlled by Property 0x68 – EMV Message Format. There are currently 2 selectable formats: Original and DynaPro. It is a TLV object with the following contents.

Original Format:

```
FD<len>/* container for generic data */
       DFDF25(IFD Serial Number)<len><val>
       FA<len>/* container for generic data */
              <tags defined by DFDF02 >
                . Note: Sensitive Data cannot be defined in DFDF02
                .
              DFDF4D(Masked T2 ICC Data)
              DFDF52 - Card Type Used
              F8<len>/* container tag for encrypted data */
                     DFDF56(Encrypted Transaction Data KSN)<len><val>
                     DFDF57(Encrypted Transaction Data Encryption Type)<val>

                     FA<len>/* container for generic data */
                            DF30(Encrypted Tag 56 TLV, T1 Data)<len><val>
                            DF31(Encrypted Tag 57 TLV, T2 Data)<len><val>
                            DF32(Encrypted Tag 5A TLV, PAN)<len><val>
                            DF35(Encrypted Tag 9F1F TLV, T1 DD)<len><val>
                            DF36(Encrypted Tag 9F20 TLV, T2, DD)<len><val>
                            DF37(Encrypted Tag 9F61 TLV, T2 CVC3)<len><val>
                            DF38(Encrypted Tag 9F62 TLV, T1,PCVC3)<len><val>
                            DF39(Encrypted Tag DF812A TLV, T1 DD)<len><val>
                            DF3A(Encrypted Tag DF812B TLV, T2 DD)<len><val>
                            DF3B(Encrypted Tag DFDF4A TLV, T2 ISO Format)<len><val>
                            DF40(Encrypted Value only of DFDF4A, T2 ISO Format)<len><val>
```

DynaPro Format:

```
F9<len>/* container for MAC structure and generic data */
       DFDF54(MAC KSN)<len><val>
       DFDF55(MAC Encryption Type)<len><val>
       DFDF25(IFD Serial Number)<len><val>
       FA<len>/* container for generic data */
              70<len>/*container for ARQC */
                     DFDF53<len><value>/*fallback indicator */
                     5F20<len><value>/*cardholder name */
                     5F30<len><value>/*service code */
                     DFDF4D<len><value>/* Mask T2 ICC Data */
                     DFDF52<len><value>/* card type */
                     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>
(Buffer if any to be a multiple of 8 bytes)
CBC-MAC (4 bytes, always set to zeroes)
```

The Value inside tag DFDF59 is encrypted and contains the following after decryption:

```
              FC<len>/* container for encrypted generic data */
                     <tags defined by DFDF02 >
                   .
                   .
```

Appendix E ARQC Response (from online Processing)

This section gives the format of the data for the Online Processing Result / Acquirer Response message. This request is sent to the reader in response to an ARQC Message notification from the reader. The output is controlled by Property 0x68 – EMV Message Format. There are currently 2 selectable formats: Original and DynaPro. It is a TLV object with the following contents.

Original format:

```
F9<len>/* container for ARQC Response data */
       DFDF25 (IFD Serial Number)<len><val>
       FA<len>/* Container for generic data */
              70<len>/* Container for ARQC */
              8A<len> approval
              Further objects as needed...
```

DynaPro format:

```
F9<len>/* container for MAC structure and generic data */
       DFDF54 (MAC KSN)<len><val>
       DFDF55 (Mac Encryption Type><len><val>
       DFDF25 (IFD Serial Number)<len><val>
       FA<len>/* Container for generic data */
              70<len>/* Container for ARQC */
              8A<len> approval
(ARQC padding, if any, to be a multiple of 8 bytes)
CBC-MAC (4 bytes, use MAC variant of MSR DUKPT key that was used in ARQC request, from
message length up to and including ARQC padding, if any)
```

Appendix F Transaction Result Message – Batch Data Format

This section gives the format of the data the device uses to do completion processing. The output is controlled by Property 0x68 – EMV Message Format. There are currently 2 selectable formats: Original and DynaPro. It is a TLV object with the following contents.

Original Format:

```
FE<len>/* container for generic data */
       DFDF25(IFD Serial Number)<len><val>
       FA<len>/* container for generic data */
              F0<len>/* Transaction Results */
                     F1<len>/* container for Status Data */
                     … /* Status Data tags */
                            DFDF1A - Transaction Status (See DFDF1A descriptions)
                            DFDF1B - Additional Transaction Information (always 0)
                            DFDF52 - Card Type Used

                     F2<len>/* container for Batch Data */
                     … /* Batch Data tags defined in DFDF17 */
                     …/* Note: Sensitive Data cannot be defined in DFDF17*/

                     F3<len>/* container for Reversal Data, if any */
                     … /* Reversal Data tags defined in DFDF05 */
                     …/* Note: Sensitive Data cannot be defined in DFDF05*/

                     F7<len>/* container for Merchant Data */
                     … /* < Merchant Data tags */

                     F8<len>/* container tag for encrypted data */
                            DFDF56(Encrypted Transaction Data KSN)<len><val>
                            DFDF57(Encrypted Transaction Data Encryption Type)<val>

                     FA<len>/* container for generic data */
                            DF30(Encrypted Tag 56 TLV, T1 Data)<len><val>
                            DF31(Encrypted Tag 57 TLV, T2 Data)<len><val>
                            DF32(Encrypted Tag 5A TLV, PAN)<len><val>
                            DF35(Encrypted Tag 9F1F TLV, T1 DD)<len><val>
                            DF36(Encrypted Tag 9F20 TLV, T2, DD)<len><val>
                            DF37(Encrypted Tag 9F61 TLV, T2 CVC3)<len><val>
                            DF38(Encrypted Tag 9F62 TLV, T1,PCVC3)<len><val>
                            DF39(Encrypted Tag DF812A TLV, T1 DD)<len><val>
                            DF3A(Encrypted Tag DF812B TLV), T2 DD<len><val>
                            DF3B(Encrypted Tag DFDF4A TLV, T2 ISO Format)<len><val>
                            DF40(Encrypted Value only of DFDF4A, T2 ISO
Format)<len><val>
```

F.1 DFDF1A Transaction Status Return Codes

0x00 = Approved
0x01 = Declined
0x02 = Error
0x10 = Cancelled by Host
0x1E = Manual Selection Cancelled by Host
0x1F = Manual Selection Timeout
0x21 = Waiting for Card Cancelled by Host
0x22 = Waiting for Card Timeout
0x23 = Cancelled by Card Swipe
0xFF = Unknown

DynaPro Format:

```
F9<len>/* container for MAC structure and generic data */
        DFDF54(MAC KSN)<len><val>
        DFDF55(MAC Encryption Type)<len><val>
        DFDF25(IFD Serial Number)<len><val>
        FA<len>/* container for generic data */
                F0<len>/* Transaction Results */
                        F1<len>/* container for Status Data */
                                … /* Status Data tags */
                        F8<len>/* container tag for encryption */
                                DFDF59(Encrypted Data Primative)<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>
                        F7<len>/* container for Merchant Data */
                                … /* < Merchant Data tags */
(Buffer if any to be a multiple of 8 bytes)
CBC-MAC (4 bytes, always set to zeroes)
```