MAGTEK®
SECURITY FROM THE INSIDE

Dynamag, DynaMAX, eDynamo, mDynamo, tDynamo, DynaWave, and iDynamo 6

Secure Card Reader Authenticator
Programmer's Reference (C++)

September 2019

Manual Part Number:
D99875725-80

REGISTERED TO ISO 9001:2015

MagTek I 1710 Apollo Court I Seal Beach, CA 90740 I Phone: (562) 546-6400 I Technical Support: (888) 624-8350
www.magtek.com

Table 0.1 – Revisions

Rev Number	Date	Notes
10	9/30/2014	Initial Release
20	1/28/2015	Add helper functions: GetProductID, GetMagnePrintLength, GetEncryptionStatus, GetExpDate, GetExpDateMonth, GetExpDateYear, GetPAN, GetPANLength, GetTrack1DecodeStatus, GetTrack2DecodeStatus, GetTrack3DecodeStatus, GetBatteryLevel, GetSwipeCount, TerminalBLEConnection.
30	02/17/2015	Add eDynamo EMV L2 functions. Add functions GetTLVPayload, GetResultCode, GetCardServiceCode
40	05/17/2016	Added DynaPro format for EMV transaction messages.
50	10/28/2016	Added support for mDynamo.
60	08/22/2018	Added support for tDynamo and DynaWave.
70	01/21/2019	Updated to correctly reference Bluetooth LE.
80	10/08/2019	Updated events for the callback OnTransactionStatus(). Added EMV result codes for GetResultCode(). Updated the parameters for function startTransaction(): cardType, options, transactionType, and mode.

SOFTWARE LICENSE AGREEMENT

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

TERMS, CONDITIONS, AND RESTRICTIONS

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

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

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

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

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

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

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

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

IN NO EVENT WILL LICENSOR BE LIABLE FOR ANY DAMAGES, INCLUDING ANY LOST PROFITS, LOST SAVINGS, OR OTHER INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE, OR INABILITY TO USE, THE SOFTWARE. Licensee's sole remedy in

the event of a defect in material or workmanship is expressly limited to replacement of the Software disk(s) if applicable.

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

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

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

Table of Contents

SOFTWARE LICENSE AGREEMENT 4
Table of Contents 6
1 Introduction 10
1.1 About the MagTek SCRA C++ Demo 10
1.2 Nomenclature 10
1.3 SDK Contents 10
1.4 System Requirements 12
1.5 Interfaces for Operating Systems 12
2 How to Set Up the MagTek SCRA Libraries 13
2.1 How to Setup Up the MagTek SCRA Development Environment 13
2.2 How to Connect DynaMAX or eDynamo to a Windows Host via Bluetooth LE 13
3 MTSCRA Library Functions 18
3.1 MTUSCRAGetDeviceList (LEGACY) 18
3.2 MTUSCRAOpenDevice (LEGACY) 18
3.3 MTUSCRACloseDevice (LEGACY) 18
3.4 MTUSCRASendCommand (LEGACY) 18
3.5 MTUSCRAGetCardData (LEGACY) 19
3.6 MTUSCRAGetCardDataStr (LEGACY) 19
3.7 MTUSCRAClearBuffer (LEGACY) 19
3.8 MTUSCRACardDataStateChangedNotify (LEGACY) 20
3.9 MTUSCRADeviceStateChangedNotify (LEGACY) 20
3.10 MTUSCRAGetDeviceState (LEGACY) 20
3.11 MTUSCRAGetCardDataState (LEGACY) 20
3.12 MTUSCRAGetPID (LEGACY) 21
3.13 GetProductID 21
3.14 GetDeviceList 21
3.15 OpenDevice 21
3.16 CloseDevice 22
3.17 IsDeviceConnected 22
3.18 SendCommand 22
3.19 SendCommandWithLength 22
3.20 GetCardData 23

3.21 ClearCardData 23
3.22 OnDataReceived 23
3.23 OnDeviceConnectionStateChanged 23
3.24 GetMaskedTracks 24
3.25 GetTrack1Masked 24
3.26 GetTrack2Masked 24
3.27 GetTrack3Masked 24
3.28 GetTrack1 24
3.29 GetTrack2 25
3.30 GetTrack3 25
3.31 GetMagnePrint 25
3.32 GetMagnePrintLength 25
3.33 GetMagnePrintStatus 25
3.34 GetEncryptionStatus 25
3.35 GetDeviceSerial 26
3.36 GetSessionID 26
3.37 GetKSN 26
3.38 GetDeviceName 26
3.39 GetCapMagneSafe20Encryption 26
3.40 GetCapMagStripeEncryption 27
3.41 GetCapTracks 27
3.42 GetExpDate 27
3.43 GetCardExpDate 27
3.44 GetExpDateMonth 27
3.45 GetExpDateYear 27
3.46 GetCardLast4 28
3.47 GetCardIIN 28
3.48 GetCardName 28
3.49 GetPAN 28
3.50 GetPANLength 28
3.51 GetCardPANLength 28
3.52 GetFirmware 29
3.53 GetTrackDecodeStatus 29
3.54 GetTrack1DecodeStatus 29

3.55 GetTrack2DecodeStatus 29
3.56 GetTrack3DecodeStatus 29
3.57 GetBatteryLevel 29
3.58 GetSwipeCount 30
3.59 TerminateBLEConnection 30
3.60 GetTLVPayload 30
3.61 GetResultCode 31
3.62 GetCardServiceCode 32
3.63 StartTransaction (EMV Only) 32
3.64 SetUserSelectionResult (EMV Only) 34
3.65 SetAcquirerResponse (EMV Only) 35
3.66 OnTransactionStatus (EMV Only) 35
3.67 OnDisplayMessageRequest (EMV Only) 36
3.68 OnUserSelectionRequest (EMV Only) 37
3.69 OnARQCReceived (EMV Only) 37
3.70 OnTransactionResult (EMV Only) 38
3.71 SendExtendedCommand (EMV Only) 38
3.72 OnDeviceExtendedResponse (EMV Only) 39
4 MTSCRA Library Enumerations and Structures 40
4.1 EErrorValues 40
4.2 EDeviceStateValues 40
4.3 ECardDataStateValues 40
4.4 MTMSRDATA 40
4.5 DYNAMAX MODE 41
5 MTSCRA Library Callback Functions 42
5.1 CallBackCardDataStateChanged 42
5.2 CallBackDeviceStateChanged 42
5.3 CallBackOnCardDataReceived 42
5.4 CallBackOnDeviceResponse 43
5.5 CallBackOnDeviceExtendedResponse 43
Appendix A Sample Code 44
A.1 Open Default Device 44
A.2 Set Up a Callback to Receive Card Data 44
Appendix B ARQC Message Format 45

Appendix C ARQC Response (from online Processing) 46
Appendix D Transaction Result Message – Batch Data Format 47
D.1 DFDF1A Transaction Status Return Codes 47
Appendix E Supported Device Features 49

1 Introduction

This document provides instructions for software developers who want to create software solutions that include a Dynamag, DynaMAX, eDynamo, mDynamo, tDynamo, DynaWave, or iDynamo 6 connected to a Windows-based host via USB. It is part of a larger library of documents designed to assist Dynamag, DynaMAX, eDynamo, mDynamo, tDynamo, DynaWave, and iDynamo 6 implementers, which includes the following documents available from MagTek:

- ***D99875724 Dynamag, DynaMAX, eDynamo, mDynamo, tDynamo, DynaWave Programmer's Reference (Java / Java Applet)***
- ***D99875725 Dynamag, DynaMAX, eDynamo, mDynamo, tDynamo, DynaWave Programmer's Reference (C++)***
- ***D99875728 aDynamo, uDynamo, Dynamag, DynaMAX, eDynamo, Bullet, tDynamo, DynaWave Programmer's Reference (Android)***
- ***D99875475 MagneSafe V5 Communication Reference Manual***

1.1 About the MagTek SCRA C++ Demo

The MTSCRA C++ Demo, available from MagTek, provides demonstration source code and reusable MTSCRA DLLs that provide developers of custom software solutions with an easy-to-use interface for Dynamag, DynaMAX, eDynamo, mDynamo, tDynamo, DynaWave, and iDynamo 6. Developers can include the MTSCRA DLLs 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 (eg. DynaMAX) 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.

1.3 SDK Contents

File	Description
include\mtuscra.h	Header file of library
Win32\MTSCRA.DLL	MagTek SCRA 32 bit Library
Win32\MTSCRA.LIB	MagTek SCRA 32 bit lib file for C/C++ link.
Win32\MTSCRABLE.DLL	MagTek SCRA 32 bit Bluetooth Smart Library.

X64\MTSCRA.DLL	MagTek SCRA 64 bit Library
X64\MTSCRA.LIB	MagTek SCRA 64 bit lib file for C/C++ link.
X64\MTSCRABLE.DLL	MagTek SCRA 64 bit Bluetooth Smart Library.

1.4 System Requirements

Tested operating systems:
Windows 7
Windows 8
Windows 8.1
Windows 10

Tested development environments:
Windows 10 with Microsoft Visual Studio 2017

1.5 Interfaces for Operating Systems

The following table matches the device interface to operating system.

Device	Interface	Operating System
Dynamag	USB	Windows 7, Windows 8 & 8.1, Windows 10
DynaMAX	USB	Windows 7, Windows 8 & 8.1, Windows 10
	Bluetooth LE	Windows 8 & 8.1, Windows 10
eDynamo	USB	Windows 7, Windows 8 and 8.1, Windows 10
	Bluetooth LE	Windows 8 & 8.1, Windows 10
mDynamo	USB	Windows 7, Windows 8 and 8.1, Windows 10
tDynamo	USB	Windows 7, Windows 8 and 8.1, Windows 10
DynaWave	USB	Windows 7, Windows 8 and 8.1, Windows 10
iDynamo 6	USB	Windows 7, Windows 8 and 8.1, Windows 10

2 How to Set Up the MagTek SCRA Libraries

2.1 How to Setup Up the MagTek SCRA Development Environment

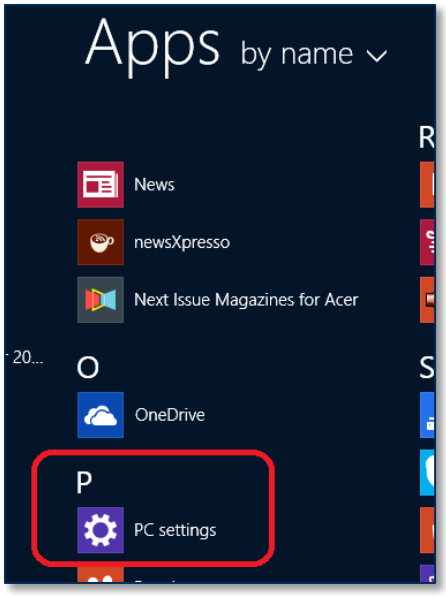
To set up the MTSCRA Libraries, follow these steps:
1) Download the ***Dynamag, DynaMAX, eDynamo, mDynamo, tDynamo, and DynaWave Secure Card Reader Authenticator Windows API Install***, available from MagTek.com
https://www.magtek.com/Content/SoftwarePackages/99510133.exe
2) Right-click **99510133.exe** and select **Run as administrator** . The installer will place all dependencies in appropriate paths.

To build and run the MTSCRA Demo software, follow these steps:
1) For 64-bit machine, launch Visual Studio and open **C:\Program Files (x86)\MagTek\SCRA Windows SDK\Sample Code\CPP\Source\VCDemo.vcxproj**
2) For 32-bit machine, launch Visual Studio and open **C:\Program Files\MagTek\SCRA Windows SDK\Sample Code\CPP\Source\VCDemo.vcxproj**
3) In the **Solution Explorer** , select **VCDemo**.
4) Select **Build** > **Build VCDemo**, or press **Shift-F6** to build the MTSCRA Demo.
5) Select **Debug**->**Start Without Debugging** to run the MTSCRA Demo, or select **Debug**->**Start Debugging** to run the MTSCRA Demo in debug mode.

2.2 How to Connect DynaMAX or eDynamo to a Windows Host via Bluetooth LE

To connect DynaMAX or eDynamo to a host with Windows 8.1 or higher and Bluetooth 4.0 hardware that supports Bluetooth LE, follow these steps:

1) If you are using an external Bluetooth adapter, install any required drivers and connect it to the host.
2) On the host, install and configure the software you intend to use with DynaMAX or eDynamo:
   a) Make sure the host software is configured to look for the device on the proper connection.
   b) Make sure the host software knows which device(s) it should interface with.
   c) Make sure the host software is configured to properly interpret incoming data from the device. This depends on whether the device is configured to transmit data in GATT format or streaming format emulating a keyboard.
3) Make sure the DynaMAX's batteries are installed and have adequate charge. If using eDynamo, make sure the device has an adequate charge.
4) Test the batteries by powering on the DynaMAX or eDynamo device. Provided the device is not already paired, the Bluetooth Status LED will flash blue every two seconds for up to 60 seconds until pairing is complete. If the Bluetooth Status LED is solid blue, the device is already paired with a host. Unpair from the host it is already paired with before continuing.
5) Enter app mode, scroll down to **Apps by name**, and launch the Windows **PC Settings** app.



6) In the left side navigator, select **PC and devices** > **Bluetooth**.
7) Make sure Bluetooth is turned on and close the **PC and devices** app.
8) Launch the Windows **Manage Bluetooth Devices** app by following these steps:
   a) Enter desktop mode by swiping in from the left side of the touchscreen.
   b) Touch the Bluetooth icon in the system tray and select **Add a Bluetooth Device** (see **Figure 2-1**).

Figure 2-1 - Launch Manage Bluetooth Devices App from Desktop Mode

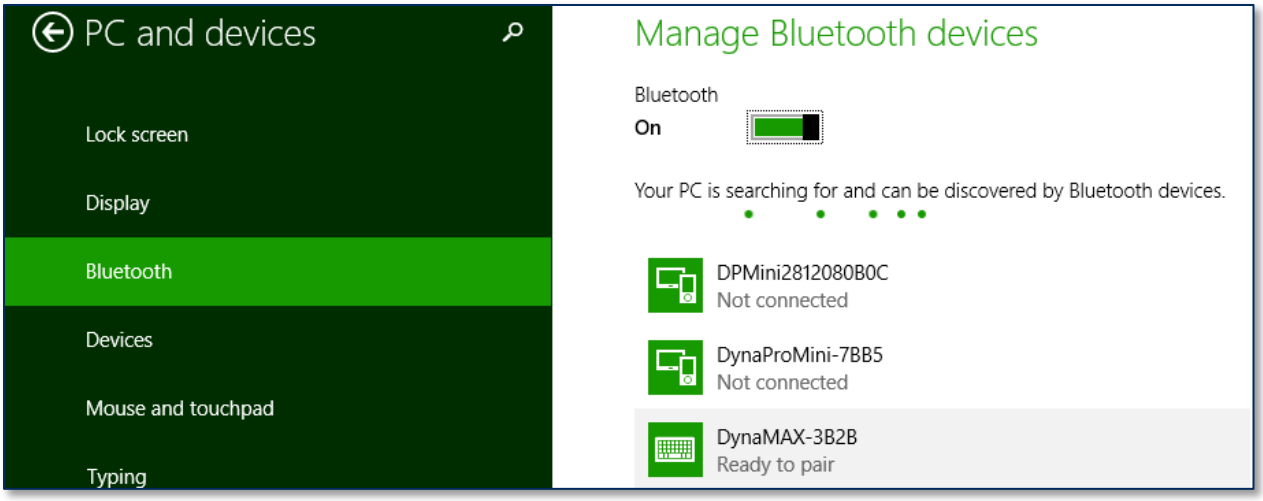
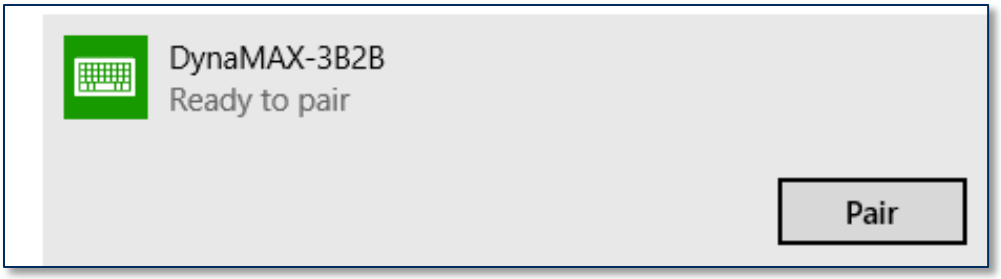
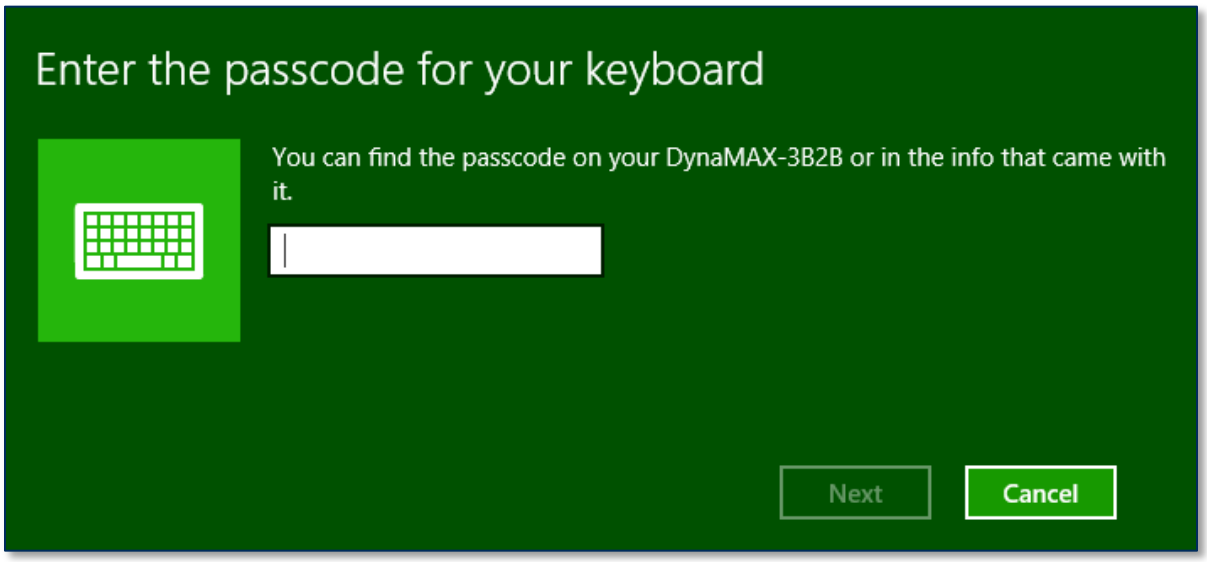


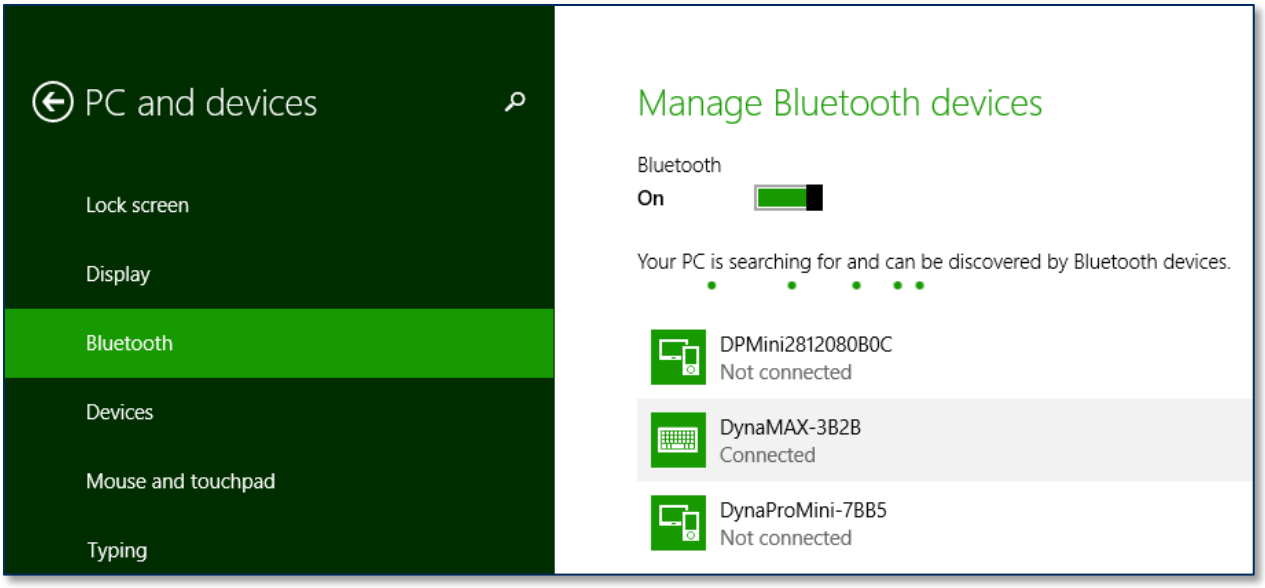
Figure 2-2 – Windows 8 Manage Bluetooth Devices App

9) Locate the serial number on the label on the bottom of the device. Note the final four digits.
10) Read through the list of pairable devices and locate the device called **DynaMAX-nnnn** or **eDynamo-nnnn**, where nnnn is the last four digits of the device's serial number (if the device does not show in the list, power it off then power it back on). Below the device name you should see the text **Ready to pair**.
11) Select the device and press the **Pair** button. If the device is configured to run in KB mode, Windows will prompt you **Enter the passcode for your keyboard**.





12) Enter default passcode **000000** (or the device's actual password if it has been configured differently), then press the **Next** button. Windows will return you to the **Manage Bluetooth devices** page. After a short period of time, you will see the text **Connected** below the device you are pairing with. After a few seconds the device will disconnect, which is normal power-saving behavior.



13) Use the host software to test swiping a card. If you do not yet have host software and the device is configured to run in KB mode, open any text editor and swipe a card. The card contents should appear in the text editor.
14) The device consumes very little power when not transmitting card data, so it is not necessary to power off the device to conserve power. If the device appears as **Not connected** in the Windows list of Bluetooth devices, swiping a card should cause the device to reconnect briefly, transmit the card data, then disconnect.
15) Remember to change the default password. See the DynaMAX Programmer's Reference documents for details.

To unpair from the device:

1) Locate the device in the **Manage Bluetooth devices** window.

Press the **Remove device** button.

3 MTSCRA Library Functions

After creating your C++ project, include MTUSCRA.H and MTSCRA.LIB, then use the functions in this library to communicate with the device.

3.1 MTUSCRAGetDeviceList (LEGACY)

This function gets a list of connected SCRA swipe readers (devices).

```
MTUSCRA_API LPSTR WINAPI MTUSCRAGetDeviceList();
```

Return Value:
Return Null-terminated device paths string, separated by ‘,’.

3.2 MTUSCRAOpenDevice (LEGACY)

This function opens a device.

```
MTUSCRA_API DWORD WINAPI MTUSCRAOpenDevice(LPSTR lpDeviceName);
```

Parameter	Description
lpDeviceName	Path name of the device to open. Passing an empty string will open the first device found. Host software can view the device path name by calling **MTUSCRAGetDeviceList** (LEGACY)**.**

Return Value: See section **4.1 EErrorValues.**

3.3 MTUSCRACloseDevice (LEGACY)

This function closes currently opened device.

```
MTUSCRA_API DWORD WINAPI MTUSCRACloseDevice();
```

Return Value: See section **4.1 EErrorValues.**

3.4 MTUSCRASendCommand (LEGACY)

This function sends a command directly to the device.

```
MTUSCRA_API DWORD WINAPI MTUSCRASendCommand(LPSTR lpCmd, DWORD
lpdwCmdLen, LPSTR lpResult, DWORD lpdwCmdLen);
```

Parameter	Description
lpCmd	Command to send to the device. For example:`0003` (Where “00” is command number, “03” is property ID) will retrieve device serial number.
lpdwCmdLen	Length of command buffer.
lpResult	Buffer to receive result
lpdwCmdLen	Length of the receiving buffer

Return Value:
See section **4.1 EErrorValues.**

3.5 MTUSCRAGetCardData (LEGACY)

This function retrieves card data through a predefined structure.

```
MTUSCRA_API DWORD WINAPI MTUSCRAGetCardData(MTMSRDATA* lpMTMSRDATA);
```

Parameter	Description
lpMTMSRDATA	MSR Data Structure. For more information see section **4.4 MTMSRDATA**

Return Value:
See section **4.1 EErrorValues.**

3.6 MTUSCRAGetCardDataStr (LEGACY)

This function retrieves card data through a predefined string and field separator.

```
MTUSCRA_API DWORD WINAPI MTUSCRAGetCardDataStr(LPSTR lpStrData,LPSTR
lpStrFieldDelimiter);
```

Parameter	Description
lpStrData	Buffer to receive Data 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
lpStrFieldDelimiter	Delimiter to separate the data fields.

Return Value:
See section **4.1 EErrorValues.**

3.7 MTUSCRAClearBuffer (LEGACY)

This function clears the card data buffer.

```
MTUSCRA_API void WINAPI MTUSCRAClearBuffer(void);
```

Return Value:
None

3.8 MTUSCRACardDataStateChangedNotify (LEGACY)

This function sets a callback function to notify the host software when the card data state changes.

```
MTUSCRA_API void  WINAPI
MTUSCRACardDataStateChangedNotify(CallBackCardDataStateChanged
lpFuncNotify);
```

Parameter	Description
lpFuncNotify	Function to call with notifications.

Return Value:
None

3.9 MTUSCRADeviceStateChangedNotify (LEGACY)

This function sets a callback function to notify the host software of device state changes. For details on device states, see section **4.2 EDeviceStateValues**.

```
MTUSCRA_API void WINAPI
MTUSCRADeviceStateChangedNotify(CallBackDeviceStateChanged
lpFuncNotify);
```

Parameter	Description
lpFuncNotify	Function to call with notifications.

Return Value:
None

3.10 MTUSCRAGetDeviceState (LEGACY)

This function retrieves current device state information. For details on device states, see section **4.2 EDeviceStateValues**.

```
MTUSCRA_API void WINAPI MTUSCRAGetDeviceState(DWORD* lpdwDeviceState)
;
```

Parameter	Description
lpdwDeviceState	A pointer to a DWORD value to receive the DeviceState enum value.

Return Value:
None

3.11 MTUSCRAGetCardDataState (LEGACY)

This function retrieves the current card data state. For details on card data states, see section **4.3 ECardDataStateValues**.

```
MTUSCRA_API void WINAPI MTUSCRAGetCardDataState(DWORD*
lpdwCardDataState) ;
```

Parameter	Description
lpdwCardDataState	A pointer to a DWORD value to receive the CardDataState enum value.

Return Value:
None

3.12 MTUSCRAGetPID (LEGACY)

This function retrieves the product ID of the device.

```
MTUSCRA_API void WINAPI MTUSCRAGetPID(DWORD* lpdwPID) ;
```

Parameter	Description
lpdwPID	A pointer to a DWORD value to receive the product ID.

Return Value:
None

3.13 GetProductID

This function retrieves the product ID of the device.

```
MTSCRA_API const char* _WINAPI_ GetProductID();
```

Return Value:
4 hex digit string indicate the product ID. For example – DynaMAX return “0018”

3.14 GetDeviceList

This function gets a list of connected devices.

```
MTSCRA_API LPSTR WINAPI GetDeviceList();
```

Return Value:
Null-terminated device paths string, separated by ‘,’.

3.15 OpenDevice

This function opens a SCRA swipe reader (device).

```
MTSCRA_API DWORD WINAPI OpenDevice(LPSTR lpDeviceName);
```

Parameter	Description
lpDeviceName	Path name of the device to open. Passing an empty string will open the first device found. The host software can get the device path name by calling **GetDeviceList**.

Return Value:
See section **4.1 EErrorValues.**

3.16 CloseDevice

This function closes currently opened device.

```
MTSCRA_API void WINAPI CloseDevice();
```

Return Value:
None

3.17 IsDeviceConnected

This function checks whether the current device is connected.

```
MTSCRA_API BOOL WINAPI IsDeviceConnected();
```

Return Value:
TRUE = connected

3.18 SendCommand

This function sends a direct command to the device using a null-terminated string.

```
MTSCRA_API LPSTR WINAPI SendCommand(LPSTR lpCmd);
```

Parameter	Description
lpCmd	Null terminated hex string to send to device. For example: • "0003" (Where "00" is command number, "03" is property ID) will retrieve device serial number from the device.

Return Value:
Null terminated hex string for the return result. NULL value for failed.

3.19 SendCommandWithLength

This function sends a direct command to the device using a string that includes the data length.

```
MTSCRA_API const LPSTR WINAPI SendCommandWithLength(const LPSTR
lpCmd);
```

Parameter	Description
lpCmd	Null terminated hex string to send to device. Example: • "000103" (Where "00" is command number, "01" is the length and "03" is property ID) will retrieve device serial number from the device.

Return Value:
Null terminated hex string for the return result. NULL value for failed.

3.20 GetCardData

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

```
MTSCRA_API LPSTR WINAPI GetCardData();
```

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

3.21 ClearCardData

This function clears the card data buffer.

```
MTSCRA_API void  WINAPI ClearCardData();
```

Return Value:
None

3.22 OnDataReceived

This function sets a callback function to notify when the device has card data to send to the host.

```
MTSCRA_API void  WINAPI OnDataReceived(CallBackOnCardDataReceived
lpFuncNotify);
```

Return Value:
None

3.23 OnDeviceConnectionStateChanged

This function sets a callback function to notify when the device state changes. See section **4.2 EDeviceStateValues** for possible values.

```
MTSCRA_API void  WINAPI
OnDeviceConnectionStateChanged(CallBackDeviceStateChanged
lpFuncNotify);
```

Parameter	Description
lpFuncNotify	Function to call with notifications.

Return Value:
None

3.24 GetMaskedTracks

This function retrieves masked card track data.

```
MTSCRA_API const char* WINAPI GetMaskedTracks();
```

Return Value:
Masked tracks data in hex string.

3.25 GetTrack1Masked

This function retrieves masked track 1 data.

```
MTSCRA_API const char* WINAPI GetTrack1Masked();
```

Return Value:
Masked track 1 in hex string.

3.26 GetTrack2Masked

This function retrieves masked track 2 data.

```
MTSCRA_API const char* WINAPI GetTrack2Masked();
```

Return Value:
Masked track 2 in hex string.

3.27 GetTrack3Masked

This function retrieves masked track 3 data.

```
MTSCRA_API const char* WINAPI GetTrack3Masked();
```

Return Value:
Masked track 3 in hex string.

3.28 GetTrack1

This function retrieves track 1 data.

```
MTSCRA_API const char* WINAPI GetTrack1();
```

Return Value:
Card track 1 in hex string.

3.29 GetTrack2

This function retrieves track 2 data.

```
MTSCRA_API const char* WINAPI GetTrack2();
```

Return Value:
Card track 2 in hex string.

3.30 GetTrack3

This function retrieves track 3 data.

```
MTSCRA_API const char* WINAPI GetTrack3();
```

Return Value:
Card track 3 in hex string.

3.31 GetMagnePrint

This function retrieves the card's MangePrint score.

```
MTSCRA_API const char* WINAPI GetMagnePrint();
```

Return Value:
Card MagnePrint score.

3.32 GetMagnePrintLength

This function retrieves the card's MangePrint data length.

```
MTSCRA_API const long WINAPI GetMagnePrintLength();
```

Return Value:
Long value indicating the length of MagnePrint data.

3.33 GetMagnePrintStatus

This function retrieves the card MagnePrint status. For more information, see ***99875475***.

```
MTSCRA_API const char* WINAPI GetMagnePrintStatus();
```

Return Value:
Card Magne Print Status.

3.34 GetEncryptionStatus

This function retrieves the encryption status after a cardholder swipe a card. For more information, see ***99875475***.

```
MTSCRA_API const char* WINAPI GetEncryptionStatus();
```

Return Value:
Big endian string containing encryption status. For exsample “0600”.

3.35 GetDeviceSerial

This function retrieves the device serial number.

```
MTSCRA_API const char* WINAPI GetDeviceSerial();
```

Return Value:
A hex string containing the device serial number.

3.36 GetSessionID

This function retrieves the device session ID, which the host can use to uniquely identify a transaction to prevent replay. For more information, see ***99875475*.**

```
MTSCRA_API const char* WINAPI GetSessionID();
```

Return Value:
Device session ID.

3.37 GetKSN

This function retrieves the device KSN.

```
MTSCRA_API const char* WINAPI GetKSN();
```

Return Value:
Device KSN.

3.38 GetDeviceName

This function gets the device product name.

```
MTSCRA_API const char* WINAPI GetDeviceName();
```

Return Value:
String containing the device’s product name.

3.39 GetCapMagneSafe20Encryption

This function will get the device’s capability for MagneSafe 2.0 encryption.

```
MTSCRA_API const char* WINAPI GetCapMagneSafe20Encryption();
```

Return Value:
“1” = MagnePrint 2.0 available

"0" = MagnePrint 2.0 unavailable

3.40 GetCapMagStripeEncryption

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

```
MTSCRA_API const char* WINAPI GetCapMagStripeEncryption();
```

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

3.41 GetCapTracks

This function gets the device's track capability.

```
MTSCRA_API const char* WINAPI GetCapTracks();
```

Return Value:
A hex string for the track capability. See document **99875475 – Track ID Enable Property**

3.42 GetExpDate

This function returns the card expiration date.

```
MTSCRA_API const char* WINAPI GetExpDate();
```

Return Value:
Hex string for YYMM, example : "1201" means January 2012.

3.43 GetCardExpDate

This function returns the card expiration date.

```
MTSCRA_API const char* WINAPI GetCardExpDate();
```

Return Value:
Hex string for YYMM, example : "1201" means January 2012.

3.44 GetExpDateMonth

This function returns month of the card expiration date after a cardholder swipes a card.

```
MTSCRA_API const char* WINAPI GetCardExpDate();
```

Return Value:
Hex string for MM, example : "01" means January.

3.45 GetExpDateYear

This function returns year of the card expiration date after a cardholder swipes a card.

```
MTSCRA_API const char* WINAPI GetCardExpDate();
```

Return Value:
Hex string for YY, example : “12” means year 2012.

3.46 GetCardLast4

This function gets the last 4 digits of the card account number (PAN).

```
MTSCRA_API const char* WINAPI GetCardLast4();
```

Return Value:
Hex string containing the last four digits of the card account number (PAN). Example : “5525”

3.47 GetCardIIN

This function gets the issuer identification number (IIN) of the card number.

```
MTSCRA_API const char* WINAPI GetCardIIN();
```

Return Value:
Hex string containing the card issuer identification number.

3.48 GetCardName

This function gets the cardholder name.

```
MTSCRA_API const char* WINAPI GetCardName();
```

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

3.49 GetPAN

Get card PAN after a cardholder has swiped a card.

```
MTSCRA_API const char* WINAPI_stdcall GetPAN();
```

Return Value:
PAN.

3.50 GetPANLength

This function retrieves the length of the card PAN value after a cardholder has swiped a card.

```
MTSCRA_API long WINAPI GetPANLength();
```

Return Value:
Length of card number or PAN

3.51 GetCardPANLength

This function retrieves the length of the card PAN value after a cardholder has swiped a card.

```
MTSCRA_API int WINAPI GetCardPANLength();
```

Return Value:
Length of card number or PAN

3.52 GetFirmware

This function gets the device's firmware revision number.

```
MTSCRA_API const char* WINAPI GetFirmware();
```

Return Value:
A hex string containing the device's firmware revision number.

3.53 GetTrackDecodeStatus

This function gets the card tracks decoding status.

```
MTSCRA_API const char* WINAPI GetTrackDecodeStatus();
```

Return Value:
Hex string, each 2 digits represent one track's decode status. "00" = success, "01" = Error or not decodable.

3.54 GetTrack1DecodeStatus

This function retrieves track 1 decode status.

```
MTSCRA_API const char* WINAPI GetTrack1DecodeStatus();
```

Return Value:
String value . "0" = success, "1" Error or not decodable.

3.55 GetTrack2DecodeStatus

This function retrieves track 2 decode status.

```
MTSCRA_API const char* WINAPI GetTrack2DecodeStatus();
```

Return Value:
String value . "0" = success, "1" Error or not decodable.

3.56 GetTrack3DecodeStatus

This function retrieves track 3 decode status.

```
MTSCRA_API const char* WINAPI GetTrack3DecodeStatus();
```

Return Value:
String value . "0" = success, "1" Error or not decodable.

3.57 GetBatteryLevel

This function retrieves battery level between 0% to 100%.

```
MTSCRA_API long __stdcall GetBatteryLevel();
```

Return Value:
Long value . from 0 to 100, indicating battery level.

3.58 GetSwipeCount

This function is reserved for future using.

```
MTSCRA_API int __stdcall GetSwipeCount();
```

Return Value:
Integer value. Always returns -1.

3.59 TerminateBLEConnection

This function will turn off DynaMAX or eDynamo Bluetooth LE connection after 1 second. This delay is intented to allow time fo the host to return a reponse if the command is issued over Bluetooth LE. To conserve battery power, the Bluetooth host should terminate the Bluetooth LE connection when it does not need to communicate to the device. Instead of using this command, the Bluetooth host may also directly terminate the Bluetooth LE connection if it is capable.
After this command, device cannot receive any command from Bluetooth LE interface. Call **CloseDevice** and **OpenDevice** to reconnect DynaMAX or eDynamo.

```
MTSCRA_API int __stdcall TerminateBLEConnection();
```

Return Value:
Integer value. 0 for success.

3.60 GetTLVPayload

This function will get the TLV format of the payload after a card swipe.

```
MTSCRA_API const LPSTR __stdcall GetTLVPayload();
```

Return Value:
A hex string representing the payload as follows.

```
FA <len> (Container for Generic Data)
        DFDF25 <len> <val> (Device Serial Number)
        F4 <len> (Container for MSR Data)
                DFDF37 <len> <val> (Encrypted Track 1)
                DFDF39 <len> <val> (Encrypted Track 2)
                DFDF3B <len> <val> (Encrypted Track 3)
                DFDF3C <len> <val> (Encrypted MagnePrint)
                DFDF3D <len> <val> (Encrypted MagnePrint Status)
                DFDF50 <len> <val> (KSN)
```

Tag	Description
FA	Container for generic data
DFDF25	IFD Serial Number
F4	Container for MSR data
DFDF37	Encrypted T1
DFDF39	Encrypted T2
DFDF3B	Encrypted T3
DFDF3C	Encrypted MagnePrint
DFDF3D	Encrypted MagnePrint Status
DFDF50	MSR KSN

3.61 GetResultCode

This function will return the result code of V5 command or V5 extendend command.

```
MTSCRA_API int __stdcall GetResultCode();
```

Return Value:
Integer value. 0 for success. For more information see ***D99875475***.

When the result codes is from an EMV command from an EMV device.

EMV Command 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 • 0x0390 = Device Has No Keys • 0x0391 = Invalid Device Serial Number • 0x0396 = Invalid System Date and Time

3.62 GetCardServiceCode

This function retrieves the card service code and should be called after a cardholder swipes a card and before calling ClearCardData.

```
MTSCRA_API const char* __stdcall GetCardServiceCode();
```

Return Value:
String representing the card service code after a cardholder swipes a card.

3.63 StartTransaction (EMV Only)

This function starts an EMV L2 transaction for smart card.

```
MTSCRA_API const char* __stdcall StartTransaction(
unsigned char timeLimit,
unsigned char cardType,
unsigned char option,
const char* amount,
unsigned char transactionType,
const char* cashBack,
const char* currencyCode
unsigned char mode);
```

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 chip card 0x03 = Magnetic Stripe and Contact chip Card. 0x04 = Contactless chip card 0x05 = Magnetic Stripe and Contactless chip card. 0x06 = Contact chip card and Contactless chip card. 0x07 = Magnetic Stripe, Contact chip card, Contactless chip card. Refere to **5.5Appendix E** for supported devices.

option	0x00 = Normal 0x01 = Bypass PIN 0x02 = Force Online 0x04 = Acquirer not available (Note: prevents long timeout on waiting for host approval) (causes "decline" to be generated internally if ARQC is generated) To use Quick Chip mode, set the most significant bit to '1'. 0x80 = Quick Chip, Normal 0x81 = Quick Chip, Bypass PIN 0x82 = Quick Chip, Force Online Refere to **5.5Appendix E** for supported devices.
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 0x02 or 0x09 = Cash back (0x09 only supported when using 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
mode	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, card insertions, and waiting on user) 0x02 = All Status changes (documents the entire transaction flow)

Return Value:

This function will always returns an empty string. To get the result code of this command, use GetResultCode() function.

3.64 SetUserSelectionResult (EMV Only)

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

```
MTSCRA_API const char* __stdcall SetUserSelectionResult(
unsigned char status,
unsigned char 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:
This function will always returns an empty string. To get the result code of this command, use GetResultCode() function.

3.65 SetAcquirerResponse (EMV Only)

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

```
MTSCRA_API const char* __stdcall SetAcquirerResponse(const char*
response);
```

Parameter	Description
response	Hex string for the response data. First two bytes indicate message length, following TLV response message.

Return Value:
This function will always returns an empty string. To get the result code of this command, use GetResultCode() function.

3.66 OnTransactionStatus (EMV Only)

This function sets a callback function to notify when the device has transaction status update to send to the host.

```
MTSCRA_API void __stdcall OnTransactionStatus(CallBackOnDeviceResponse
notif);
```

This notification will send hex string to represent transaction status.

Offset	Field Name	Value
0	Event	Indicates the event that provoked this notification • 0x00 – No events since start of transaction • 0x01 – Card inserted • 0x02 – Card error • 0x03 – Transaction Progress Change • 0x04 – Notification that device is waiting for user selection • 0x05 – Timeout on user selection • 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.

Offset	Field Name	Value
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 cardholder to present payment • 0x02 = Powering up the card • 0x03 = Selecting the application • 0x04 = Waiting for user language selection (Contact Only) • 0x05 = Waiting for user application selection (Contact Only) • 0x06 = Initiating application (Contact Only) • 0x07 = Reading application data (Contact Only) • 0x08 = Offline data authentication (Contact Only) • 0x09 = Process restrictions (Contact Only) • 0x0A = Cardholder verification (Contact Only) • 0x0B = Terminal risk management (Contact Only) • 0x0C = Terminal action analysis (Contact Only) • 0x0D = Generating first application cryptogram (Contact Only) • 0x0E = Card action analysis (Contact Only) • 0x0F = Online processing • 0x10 = Waiting online processing response • 0x11 = Transaction Complete • 0x12 = Transaction Error • 0x13 = Transaction Approved • 0x14 = Transaction Declined • 0x15 = Transaction Cancelled by MSR Swipe (MSR Only) • 0x16 = EMV error - Conditions Not Satisfied (Contact Only) • 0x17 = EMV error - Card Blocked (Contact Only) • 0x18 = Application selection failed (Contact Only) • 0x19 = EMV error - Card Not Accepted (Contact Only) • 0x1A = Empty Candidate List • 0x1B = Application Blocked
3-4	Final Status	TBD

Return Value:
None

3.67 OnDisplayMessageRequest (EMV Only)

This function sets a callback function to notify when the device has transaction message update for the host to display to user.

```
MTSCRA_API void __stdcall OnDisplayMessageRequest
(CallBackOnDeviceResponse notif);
```

This notification will send a hex string to represent transaction status.
For example:
Hex string : “50524553454e542043415244”
Represent displaying message : “PRESENT CARD”

Return Value:
None

3.68 OnUserSelectionRequest (EMV Only)

This function sets a callback function to notify when the device has user selection message in transaction for the host to present to the user.

```
MTSCRA_API void __stdcall OnUserSelectionRequest
(CallBackOnDeviceResponse notif);
```

This notification will send a hex string to represent user selection request.

Offset	Field Name	Value
0	Selection Type	This field specifies what kind of selection request this is: • 0x00 – Application Selection • 0x01 – Language Selection • Others TBD
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.

Return Value:
None

3.69 OnARQCReceived (EMV Only)

This function sets a callback function to notify when the device has an ARQC message send to the host.

```
MTSCRA_API void __stdcall OnARQCReceived (CallBackOnDeviceResponse
notif);
```

This notification will send a hex string for ARQC of this transaction.

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

Return Value:
None

3.70 OnTransactionResult (EMV Only)

This function sets a callback function to notify when the device has completed the transaction and sends the transaction result to the host.

```
MTSCRA_API void __stdcall OnTransactionResult
(CallBackOnDeviceResponse notif);
```

This notification will send a hex string for result of this transaction.

Offset	Field Name	Value
0	Signature Required	This field indicates whether a card holder signature is required to complete the transaction: • 0x00 – No signature required • 0x01 – Signature required If a signature is required, it is expected that the host will acquire the signature from the card holder as part of the transaction data.
1	Batch Data Length	Two byte binary, most significant byte first. This gives the total length of the ARQC message that follows.
3	Batch Data	See Appendix C. It is expected that the host will save this data as a record of the transaction.

Return Value:
None

3.71 SendExtendedCommand (EMV Only)

This function sends a direct extended command to the device using a binary value.

```
MTSCRA_API int __stdcall SendExtendedCommand (const LPSTR lpCmd);
```

Offset	Field Name	Value
0	Command Number	Two byte binary, most significant byte first. This is the Command number to send to device.
2	Data Length	Two byte binary, most significant byte first. This gives the total length of the Data that follows.
4	Data	Hex data of the command.

Return Value:
Integer value. 0 for success.

3.72 OnDeviceExtendedResponse (EMV Only)

This function sets a callback function to notify when the device has completed processing the command and sends the command result to the host.

```
MTSCRA_API int __stdcall OnDeviceExtendedResponse
(CallBackOnDeviceResponse notif);
```

This notification will send a hex string for result of the extended command.

Offset	Field Name	Value
0	Result Code	Result code of the command sent to device in Hex format.
2	Data Length	Two byte binary, most significant byte first. This gives the total length of the Data that follows.
4	Data	Hex data of the command response.

Return Value:
None

4 MTSCRA Library Enumerations and Structures

The MTSCRA Library uses the following constants and data structures.

4.1 EErrorValues

Function return codes:

```
enum EErrorValues
{
   MTSCRA_ST_OK,
   MTSCRA_ST_FAILED,
   MTSCRA_ST_OPEN,
   MTSCRA_ST_INVALID_PARAM
};
```

4.2 EDeviceStateValues

Device connection states:

```
enum EDeviceStateValues
{
   MTSCRA_STATE_DISCONNECTED,
   MTSCRA_STATE_CONNECTED,
   MTSCRA_STATE_ERROR
};
```

4.3 ECardDataStateValues

Card data states:

```
enum ECardDataStateValues
{
   MTSCRA_DATA_NOTREADY,
   MTSCRA_DATA_READY,
   MTSCRA_DATA_ERROR
};
```

4.4 MTMSRDATA

Card data structure:

```
typedef struct _MTMSRDATA
{
      char m_szCardData[DEF_MSR_DATA_LEN * 3];
      char m_szCardDataMasked[DEF_MSR_DATA_LEN * 3];

      char m_szTrack1Data[DEF_MSR_DATA_LEN];
      char m_szTrack2Data[DEF_MSR_DATA_LEN];
      char m_szTrack3Data[DEF_MSR_DATA_LEN];

      char m_szTrack1DataMasked[DEF_MSR_DATA_LEN];
      char m_szTrack2DataMasked[DEF_MSR_DATA_LEN];
      char m_szTrack3DataMasked[DEF_MSR_DATA_LEN];

      char m_szMagnePrintData[DEF_MSR_DATA_LEN];
```

```
    char m_szCardEncodeType[DEF_MSR_DATA_LEN];

    char m_szMagnePrintStatus[DEF_MSR_DATA_LEN];
    char m_szDUKPTSessionID[DEF_MSR_DATA_LEN];
    char m_szDeviceSerialNumber[DEF_MSR_DATA_LEN];
    char m_szDUKPTKSN[DEF_MSR_DATA_LEN];

    char m_szFirstName[DEF_MSR_DATA_LEN];
    char m_szLastName[DEF_MSR_DATA_LEN];
    char m_szPAN[DEF_MSR_DATA_LEN];
    char m_szMonth[DEF_MSR_DATA_LEN];
    char m_szYear[DEF_MSR_DATA_LEN];


    DWORD m_dwReaderID;
    DWORD m_dwMagnePrintLength;
    DWORD m_dwMagnePrintStatus;
    DWORD m_dwTrack1Length;
    DWORD m_dwTrack2Length;
    DWORD m_dwTrack3Length;
    DWORD m_dwTrack1LengthMasked;
    DWORD m_dwTrack2LengthMasked;
    DWORD m_dwTrack3LengthMasked;
    DWORD m_dwCardEncodeType;

    DWORD m_dwTrack1DcdStatus;
    DWORD m_dwTrack2DcdStatus;
    DWORD m_dwTrack3DcdStatus;
    DWORD m_dwCardSwipeStatus;

    DWORD m_dwBatteryLevel;
} MTMSRDATA, *PMTMSRDATA;
```

4.5 DYNAMAX MODE

```
#define DYNAMAX_MODE_LONGREAD 0
#define DYNAMAX_MODE_NOTIFICATION 1
```

5 MTSCRA Library Callback Functions

The MTSCRA Library uses the following constants and data structures.

5.1 CallBackCardDataStateChanged

The library will call this function when card data is available.

```
typedef void (WINAPI *CallBackCardDataStateChanged)(DWORD
lpdwCardDataState);
```

Parameter	Description
lpdwCardDataState	Reference to **ECardDataStateValues**

Return Value:
None

5.2 CallBackDeviceStateChanged

The library will call this function when the device state changes to Disconnected or Offline.

```
typedef void (WINAPI *CallBackDeviceStateChanged)(DWORD
lpdwDeviceState);
```

Parameter	Description
lpdwDeviceState	Reference to **EDeviceStateValues**

Return Value:
None

5.3 CallBackOnCardDataReceived

The library will call this function when card data becomes available.

```
typedef void (WINAPI *CallBackOnCardDataReceived)(LPSTR lpszCardData);
```

Parameter	Description
lpszCardData	Received card data, all fields are separated by '|'. 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

Return Value:

None

5.4 CallBackOnDeviceResponse

The library will call this function when device has some data to send to the host.

```
typedef void (WINAPI * CallBackOnDeviceResponse)(LPSTR lpszHexData);
```

Parameter	Description
lpszHexData	Hex string representing raw binary data.

Return Value:
None

5.5 CallBackOnDeviceExtendedResponse

The library will call this function when device has some data to send to the host after the host first sends a command using **SendExtendedCommand** (EMV Only).

```
typedef void (__stdcall * CallBackOnDeviceExtendedResponse)(const
LPSTR lpszHexResponse);
```

Parameter	Description
lpszHexRespnse	Hex string representing raw binary data.

Return Value:
None

Appendix A Sample Code

A.1 Open Default Device

This example shows how to use the **OpenDevice** function to open the first device connected to the host.

```
if (OpenDevice("") == MTSCRA_ST_OK)
{
      printf("Success");
}
else
{
      printf("Failed");
}
```

A.2 Set Up a Callback to Receive Card Data

This example shows how to implement a static function to register callback to receive notifications that card data is available.

```
// declare a static function in a class
static void WINAPI OnCardData(const LPSTR CardDataString);
```

```
// call library to setup the callback
if (OpenDevice("") == MTSCRA_ST_OK)
{
      OnDataReceived(this->OnCardData);
}
```

Appendix B 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 C 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 D 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>
```

D.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)
```

Appendix E Supported Device Features

Feature / Product	cDynamo	DynaMAX	eDynamo	iDynamo 5	iDynamo 5 (Gen II)	iDynamo 6	kDynamo	sDynamo	tDynamo	uDynamo
MSR Swipe	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
MSR Insert	N	N	N	N	N	N	N	N	N	N
MSR 3 Tracks	Y	Y	Y	Y	Y	Y	Y	Y	Y	N
MSR Disable	Y	N	N	Y	N	N	N	N	N	N
MSR Swap Tracks 1/3	N	N	N	N	N	N	N	N	N	N
MSR Embedded V5 Head	N	N	N	N	Y	Y	Y	Y	Y	N
MSR Configurabe MSR Variants	Y	Y	Y	Y	Y	Y	Y	Y	Y	N
MSR Configurable MP Variants	N	Y	Y	N	N	Y	Y	N	Y	N
MSR SureSwipe	N	Y	Y	N	N	N	N	N	N	N
MSR JIS Capable	Y	N	N	Y	N	N	N	Y	N	N
SHA-1	N	Y	Y	N	N	N	N	N	N	N
SHA-256	N	N	N	N	N	N	N	N	N	N
Configurable SHA	N	Y	Y	N	N	N	N	N	N	N
Configurable Encryption Algorithm	N	N	N	N	N	Y	N	N	N	N
Set Mask Service Code	N	N	N	N	N	N	N	N	N	N
Never Mask Service Code	N	N	Y	Y	Y	Y	Y	Y	Y	N
MagneSafe 2.0	N	N	Y	N	N	N	N	N	N	N
EMV Contact	N	N	Y	N	N	Y	Y	N	Y	N
EMV Contactless	N	N	N	N	N	Y	Y	N	Y	N
EMV Offline ODA	N	N	Y	N	N	N	N	N	N	N
EMV MSR Flow	N	N	N	N	N	Y	Y	N	Y	N
EMV Contact Quick Chip	N	N	Y	N	N	Y	Y	N	Y	N
EMV Contactless Quick Chip	N	N	N	N	N	Y	Y	N	Y	N
External PIN Accessory Support	N	N	N	N	N	Y	N	N	N	N
Keypad Entry	N	N	N	N	N	N	N	N	N	N
Fixed Key	N	N	N	N	N	N	N	N	N	N
Secondary DUKPT Key	N	Y	Y	N	N	N	N	N	N	Y
Power Mgt Scheme (PM#)	N	2	3	N	N	7	5	N	5	4
Battery-Backed RTC	N	N	Y	N	N	N	N	N	N	N
OEM Features	N	N	N	N	N	N	N	N	N	N
Transaction Validation	N	N	N	N	N	N	N	N	N	N

Appendix E - Supported Device Features

Feature / Product	cDynamo	DynaMAX	eDynamo	iDynamo 5	iDynamo 5 (Gen II)	iDynamo 6	kDynamo	sDynamo	tDynamo	uDynamo
Display	N	N	N	N	N	N	N	N	N	N
Multi-Language	N	N	Y	N	N	Y	Y	N	Y	N
Tamper	N	N	Y	N	N	N	N	N	N	N
Extended Commands	N	N	Y	N	N	Y	Y	N	Y	N
Extended Notifications	N	N	Y	N	N	Y	Y	N	Y	N
Dual USB Ports	N	N	N	N	N	Y	N	N	Y	N
Pairing Modes	N	N	Y	N	N	N	N	N	Y	N
Custom Advertising	N	N	Y	N	N	N	N	N	Y	N
Configurable Lightning FID	Y	N	N	N	Y	Y	Y	N	N	N
Auxiliary Ports	N	N	N	N	N	N	N	N	N	N
External LED Control	N	N	N	N	N	N	N	N	N	N
Encrypt Bulk Data (b)	120	24	24	120	N	N	N	N	N	24