MAGTEK®
SECURITY FROM THE INSIDE

Dynamag, DynaMAX, and eDynamo

Secure Card Reader Authenticator
Programmer’s Reference (WCF)

November 2015

Manual Part Number:
D9982000105-10

REGISTERED TO ISO 9001:2008

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	11/25/2015	Initial Release

SOFTWARE LICENSE AGREEMENT

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

TERMS, CONDITIONS, AND RESTRICTIONS

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

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

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

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

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

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

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

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

IN NO EVENT WILL LICENSOR BE LIABLE FOR ANY DAMAGES, INCLUDING ANY LOST PROFITS, LOST SAVINGS, OR OTHER INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE, OR INABILITY TO USE, THE SOFTWARE. Licensee's sole remedy in the event of a defect in material or workmanship is expressly limited to replacement of the Software disk(s) if applicable.

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

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

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

Table of Contents

1 Introduction

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

- ***D99875724 Dynamag, DynaMAX, and eDynamo Programmer's Reference (Java and Java Applet)***
- ***D99875725 Dynamag, DynaMAX, and eDynamo Programmer's Reference (C++)***
- ***D99875475 MagneSafe V5 Communication Reference Manual***

1.1 About the MagTek SCRA WCF Demo

The MTSCRA WCF Demo, available from MagTek, provides demonstration source code and reusable MTSCRAWCF DLLs that provide developers of custom software solutions with an easy-to-use interface for Dynamag, DynaMAX, and eDynamo. Developers can include the MTSCRAWCF 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
WCF\MTDevice.DLL	MagTek SCRA Device constance library
WCF\MTLIB.DLL	MagTek SCRA interface library
WCF\MTSCRANET.DLL	MagTek SCRA library for .Net
WCF\MTSCRAWCF.DLL	MagTek SCRA library for WCF
WCF\MTServiceNet.DLL	MagTek SCRA connection service library for .Net

1.4 System Requirements

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

Microsoft .Net Framework 4.5 installed.

Tested development environments:
Windows 8.1 with Microsoft Visual Studio 2013

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
	BLE	Windows 8, 8.1 & 10
eDynamo	USB	Windows 7, Windows 8, 8.1 & Windows 10
	BLE	Windows 8, 8.1 & 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, and eDynamo Secure Card Reader Authenticator Windows API Install***, available from MagTek.com
(**Support** > **Software** > **Programming Tools** > **Dynamag, DynaMAX, and eDynamo Secure Card Reader Authenticator Windows API**>
Dynamag, DynaMAX, and eDynamo SCRA Windows API)

2) Right-click **99510133.exe** and select **Run as administrator** . The installer will place all dependencies in appropriate paths.

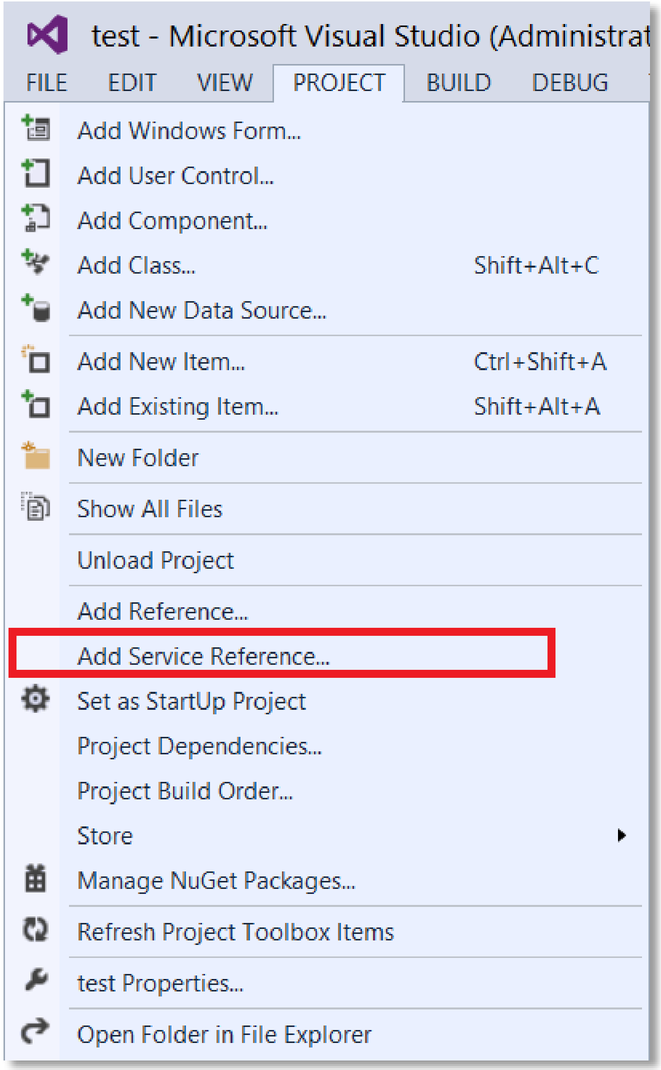
To build and run the MTSCRA Demo software, follow these steps:
1) For 64-bit machine, launch Visual Studio 2013 and open **C:\Program Files (x86)\MagTek\SCRA Windows SDK\Sample Code\CPP\Source\VCDemo.vcxproj**
2) For 32-bit machine, launch Visual Studio 2013 and open **C:\Program Files\MagTek\SCRA Windows 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 MTSCRA WCF Service to WCF Host Demo

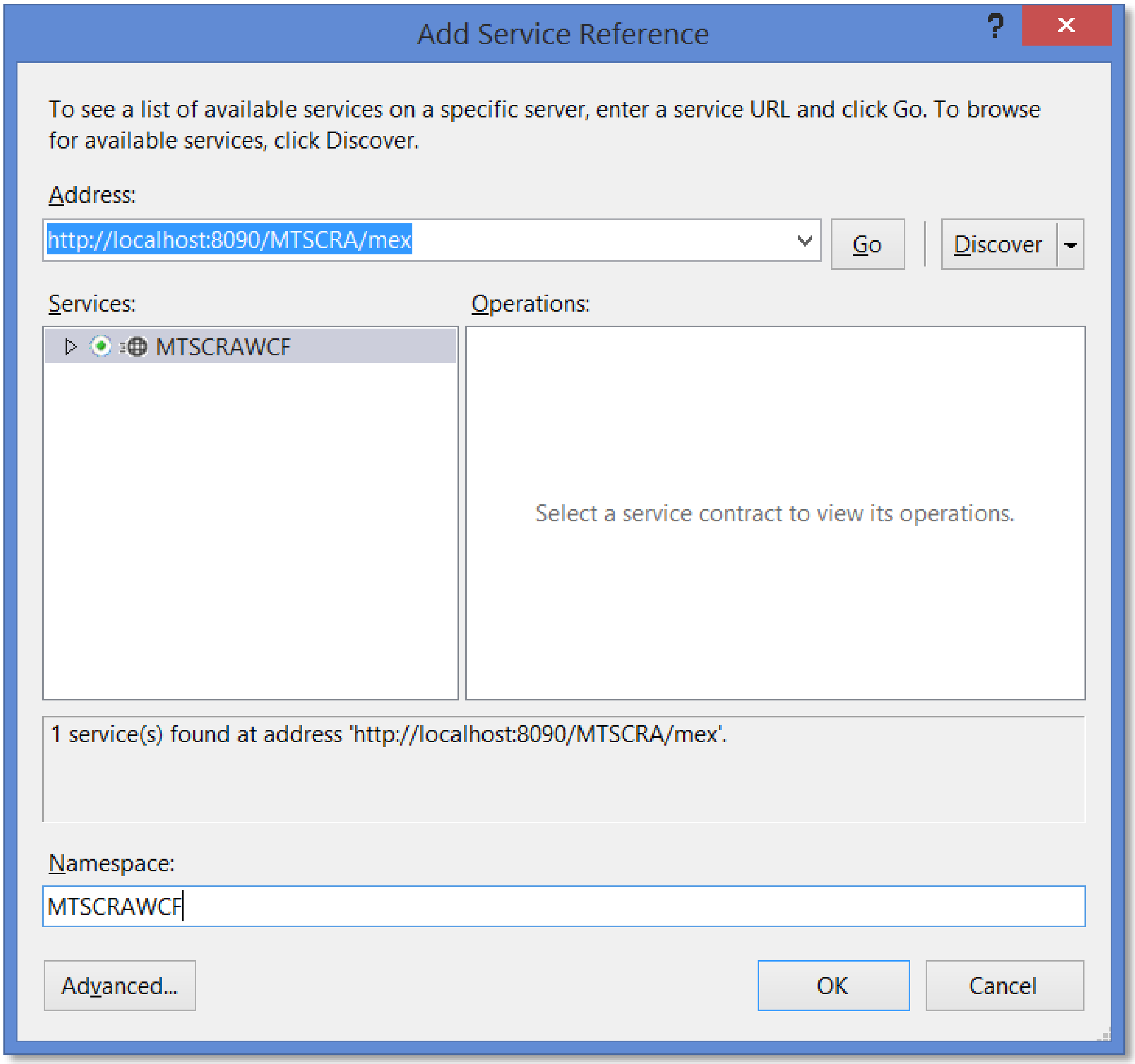
To use the WCF Host Demo (MTSCRAWCFHost.exe) use base address http://localhost:8090/MTSCRA and add two endpoints on it.
<endpoint address="ajax" binding="webHttpBinding" behaviorConfiguration="ajaxBehavior" contract="MTSCRAWCF.IMTSCRA" />
<endpoint address="mex" binding="mexHttpBinding" contract="IMetadataExchange" />

2.2.1 Connect web service in C# Project

1) Open your C# project
2) Select **PROJECT**->**Add Service Reference**



3) Input the base address to reference the WCF service, then click OK.



To see a list of available services on a specific server, enter a service URL and click Go. To browse for available services, click Discover.

Address: http://localhost:8090/MTSCRA/mex

1 service(s) found at address 'http://localhost:8090/MTSCRA/mex'.

Namespace: MTSCRAWCF

4) Access class MTSCRA in yourproject.MTSCRAWCF namespace.

2.2.2 Connect ajax service in web page javascript

1) Open a web page project.
2) Create a page clientside.aspx.
3) Insert ScriptManager object:

```
        <asp:ScriptManager ID="ScriptManager1" runat="server">
        </asp:ScriptManager>
```

This step will enable the microsoftajax.js module.

4) Insert javascript referent to WCF:

```
    <script src="http://localhost:8090/MTSCRA/ajax/js"></script>
```

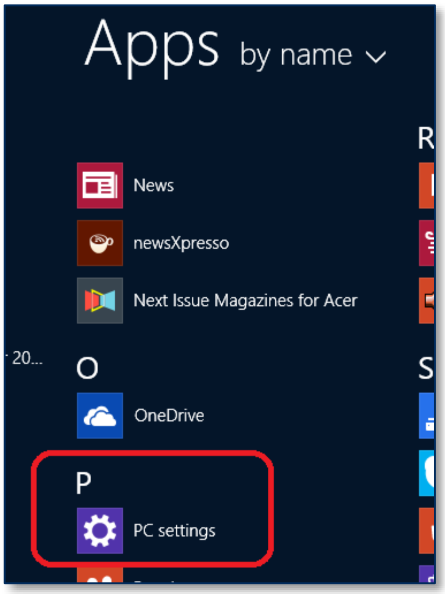
5) Access class IMTSCRA in tempuri.org namespace:

```
            var msr = new tempuri.org.IMTSCRA();
```

2.3 How to Connect DynaMAX or eDynamo to a Windows Host via BLE

To connect DynaMAX or eDynamo to a host with Windows 8.1 or higher and Bluetooth 4.0 hardware that supports BLE, 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**).

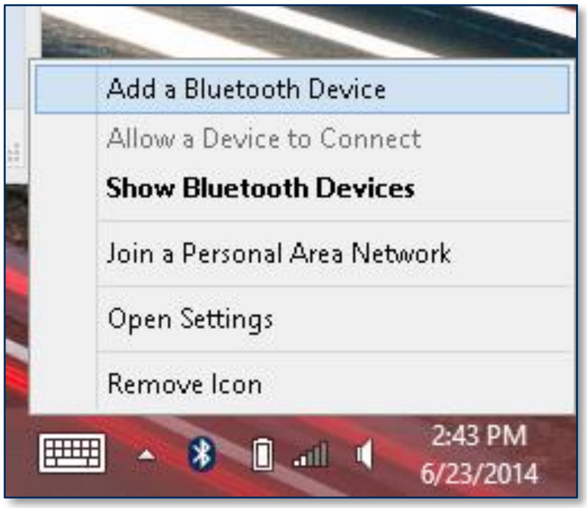
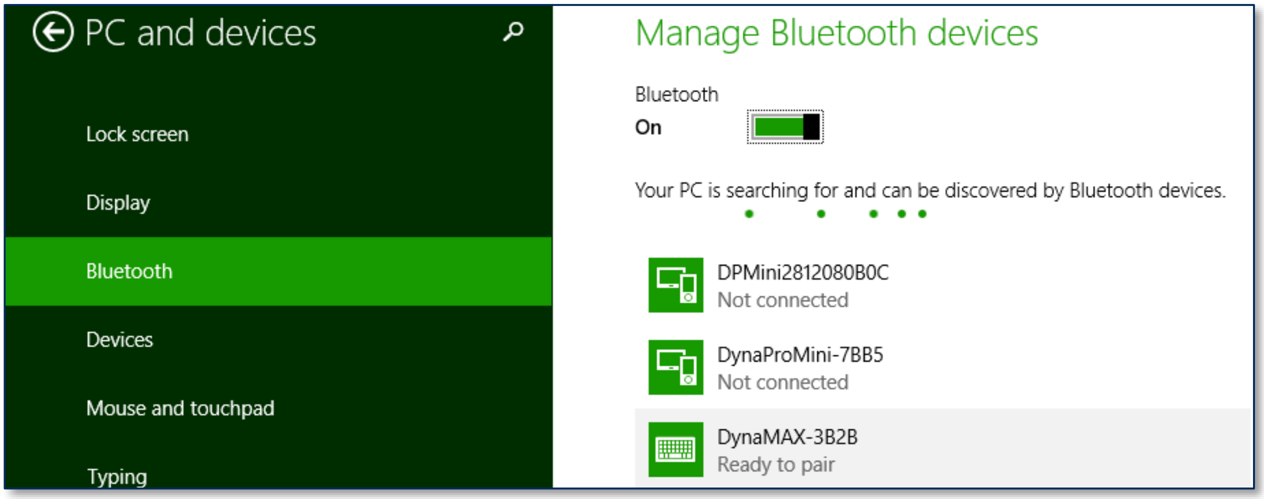


Figure 2-1 - Launch Manage Bluetooth Devices App from Desktop Mode



Manage Bluetooth devices

Bluetooth
On

Your PC is searching for and can be discovered by Bluetooth devices.

DPMini2812080B0C
Not connected

DynaProMini-7BB5
Not connected

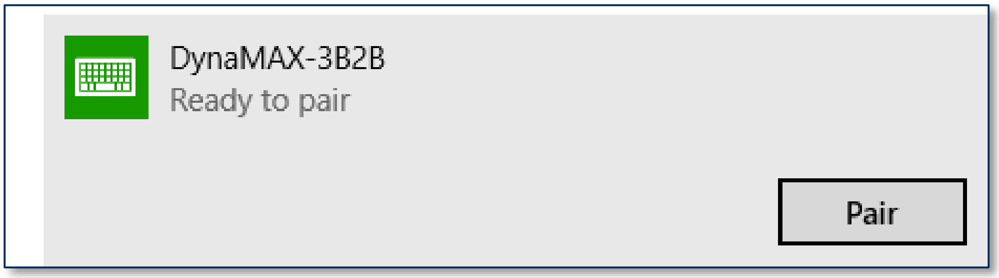
DynaMAX-3B2B
Ready to pair

Figure 2-2 – Windows 8 Manage Bluetooth Devices App

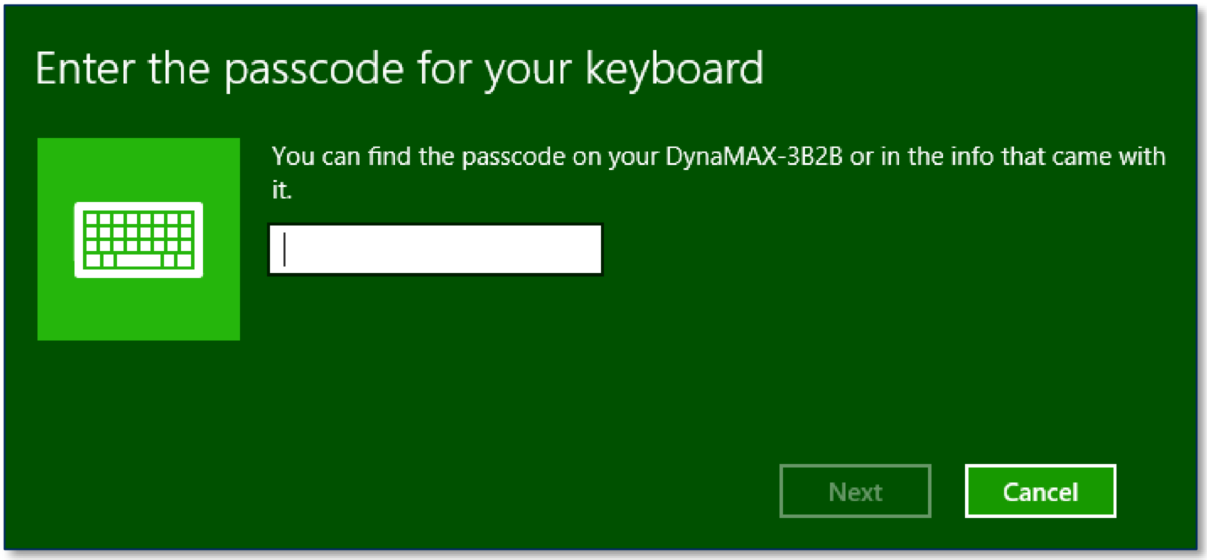
9) Locate the serial number on the label on the bottom of the device. Note the final four digits.

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**.



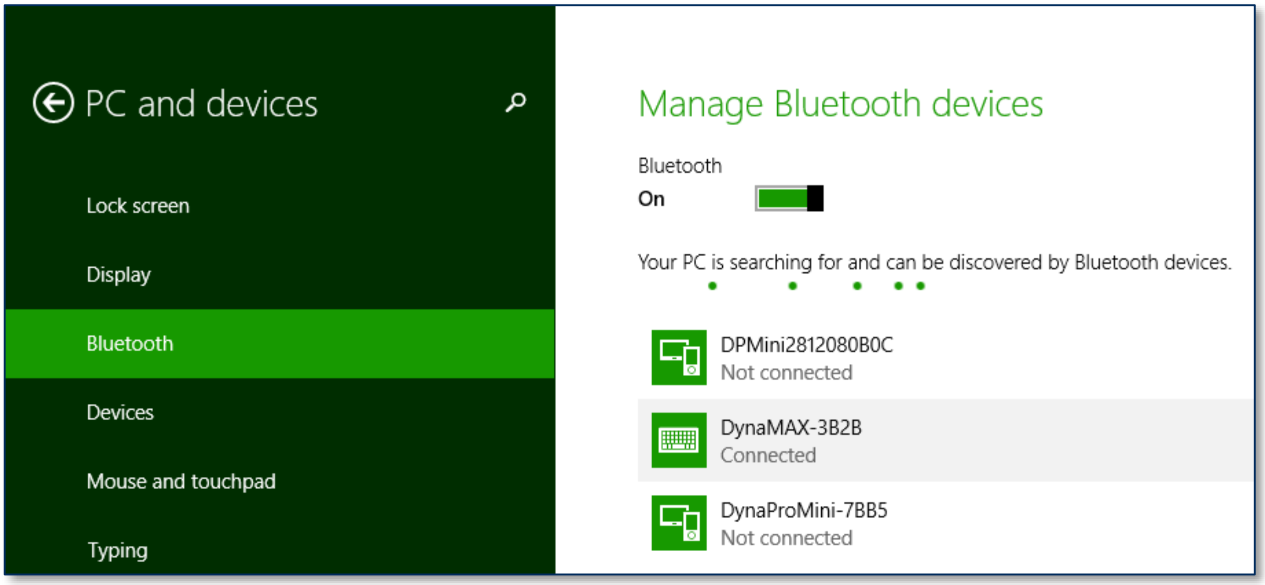
DynaMAX-3B2B
Ready to pair



Enter the passcode for your keyboard

You can find the passcode on your DynaMAX-3B2B or in the info that came with it.

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.



Bluetooth
On

Your PC is searching for and can be discovered by Bluetooth devices.

DPMini2812080B0C
Not connected

DynaMAX-3B2B
Connected

DynaProMini-7BB5
Not connected

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

MagTek SCRA WCF Library can be hosted by service, IIS or any other application and it also can be referenced just like a normal .Net dll.

3.1 getDeviceList

This function enumerate devices and return device URIs.

```
string getDeviceList(string deviceType = "");
```

Parameter	Description
deviceType	Specifies the device connection type to search. Can be “usb”, “ble” or “bleemv” or multiple types delimited by “,”. If input is empty string or null, means search all types.

Return Value:
A string with Devices URIs, delimited by “,”.

3.2 getSDKVersion

Retrieves SDK version.

```
string getSDKVersion();
```

Return Value:
The version information of the SDK.

3.3 openDevice

Open an MTSCRA device.

```
int openDevice(string DeviceName);
```

Parameter	Description
DeviceName	URI for a device, it can be retrieved by function getDeviceList. If this parameter is null or empty, will open the first device that can be found.

Return Value:
Zero for success, other for failed.

3.4 closeDevice

Close an MTSCRA device. It will release all resources attach to this device. If device is disconnected, please call this function to release resource.

```
void closeDevice();
```

Return Value:
None.

3.5 isDeviceConnected

Checks whether the opened device is connected or not.

```
bool isDeviceConnected ();
```

Return Value:
True for connected and opened, false for disconnected or closed.

3.6 isDeviceEMV

Checks whether the opened device support EMV function or not.

```
bool isDeviceEMV ();
```

Return Value:
True for device supports EMV. Otherwise device does not support EMV.

3.7 sendCommand

Sends a command to device and returns the raw response from device.

```
string sendCommand(string Command);
```

Parameter	Description
Command	Hex string for command. It does not include the length of command data. Reference to V5 command for detail.

Return Value:
Hex string of device raw response for this command.

3.8 sendCommandWithLength

Sends a command to device and returns the raw response from device.

```
string sendCommandWithLength(string Command);
```

Parameter	Description
Command	Hex string for command. It includes the length of command data. Reference to V5 command for detail.

Return Value:
Hex string of device raw response for this command.

3.9 getCardData

Retrieves last swipe card data.

```
string getCardData();
```

Return Value:

A string including card data.

3.10 getResultCode

Get last EMV command result code.

```
int getResultCode();
```

Return Value:
Zero for success, otherwise for failed.

3.11 startTransaction (eDynamo Only)

This function starts an EMV L2 transaction for smart card.

```
byte[] startTransaction(
     byte timeLimit,
     byte cardType,
     byte option,
     byte[] amount,
     byte transactionType,
     byte[] cashBack,
     byte[] currencyCode,
     byte 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 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) For example: [0,0,0,0,0x10,0], means 10.00 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 (we need to support this, returns PAN only, see EMV website for guidance) 0x40 = International Cash Advance or Cash Back 0x80 = Domestic Cash Advance or Cash Back
cashBack	Cash back Amount (EMV Tag 9F03, format n12, 6 bytes). For example: [0,0,0,0,0x10,0], means 10.00 dollars.
currencyCode	Transaction Currency Code (EMV Tag 5F2A, format n4, 2 bytes) Sample Valid values: [0x08,0x40] – US Dollar [0x09,0x78] – Euro
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 plus card insertions and waiting on user) 0x02 = All Status changes (documents the entire transaction flow)

Return Value:
This function will always returns null. To get the result code of this command, use getResultCode() function.

3.12 cancelTransaction (eDynamo Only)

This function cancels the current transaction.

```
byte[] cancelTransaction();
```

Return Value:
This function will always returns null. To get the result code of this command, use getResultCode() function.

3.13 setUserSelectionResult (eDynamo 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.

```
byte[] setUserSelectionResult(
     byte status,
     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:
This function will always returns null. To get the result code of this command, use getResultCode() function.

3.14 setAcquirerResponse (eDynamo Only)

This function informs eDynamo to process transaction decision from acquirer.

```
byte[] setAcquirerResponse(byte[] response);
```

Parameter	Description
response	Hex string for the response data. First two bytes indicate message length, following TLV response message. Reference to A.2 ARQC Response

Return Value:
This function will always returns null. To get the result code of this command, use getResultCode() function.

3.15 getTransactionProgress

This function get the device notification data.

```
string getTransactionProgress();
```

Return Value:
JSON string for progress. It contains two properties – type (integer) which references the event type and data (data attach to this event type). Reference Data Type and Structure for detail.

4 MTSCRA Library Data Type and Structures

The MTSCRA Library uses the following constants and data structures.

4.1 Device Connection State

getTransactionProgress will return the connection state after open/close a device or disconnect a device.
type : 1
data : 0 for disconnected, 1 for connected

Example:

```
 {"type":1,"data":"1"};
```

4.2 CardData

getTransactionProgress will return the carddata after a card swipe.
type : 2
data : string delimited by "|" for card data

Example :

```
{"type":2,"data":"0019|B306F96092815AA|00|00|00|00|00|61401000|144|80|
0|68|34|0|112|e1e90da02b0a4672b97f428fa1f8ca7104c52f248f992fdea89211d8
33e10070ba62f8d274fe55abee8855971364f716f4fa7d6fae47d078c7c98e7a220b84
4b127cc769344dc76bc14e3cf1638da8c4428daff38163b429037e8f80bc099e123f42
8fc06fb47743f8eeb633ba523a07|%B9812000080006827^LAST/FIRST I
^18050000000000000000?;9812000080006827=180500000000000?|8f9f09b9f68a
bc66|9010010b306f9600001a|LAST/FIRST|I|981200|18|05|e1e90da02b0a4672b9
7f428fa1f8ca7104c52f248f992fdea89211d833e10070ba62f8d274fe55abee885597
1364f716f4fa7d6fae47d078c7c98e7a220b844b127cc769344dc76b|c14e3cf1638da
8c4428daff38163b429037e8f80bc099e123f428fc06fb47743f8eeb633ba523a07||%
B9812000080006827^LAST/FIRST I
^18050000000000000000?|;9812000080006827=180500000000000?||f58763ab6c
ed3f331e4fbb3e92d84bec8e867e1c8861f647a13f6ab02778eacb483ee4a906d7feba
4124540bab017a6ce65dcfeada5774f8|100"}
```

4.3 Transaction Status

getTransactionProgress will return the transaction status after device has status changed.
type : 768 (0x300)
data : Hex string for 5 bytes

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 using 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.
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 – cardcard action analysis • 0x0F – online processing • 0x10 – waiting online processing response • 0x11 – transaction completion • 0x12 – transaction error • 0x13 – transaction approved • 0x14 – transaction declined
3-4	Final Status	TBD

Example :

```
{"type":768,"data":"0100020000"}
```

4.4 Display Message Request

getTransactionProgress will return this request to the host to display a message for the card holder. The host should display the message.
type : 769 (0x301)
data : Hex string can be decode to ANSI string

```
{"type":769,"data":"504C454153452057414954"}
// PLEASE WAIT
```

4.5 User Selection Request

getTransactionProgress will return this request to inform the host that a user selection is needed for the reader to continue processing the transaction. The host should prompt the card holder to select an item from the menu then send the setUserSelectionResult command to inform the reader that the transaction can proceed with the selected result.
type : 770 (0x302)
data : Hex string contains byte array for following structure

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.

```
{"type":770,"data":"011E53656C656374206C616E67756167653A00656E00646500
"}

/*
Select language:
en
de
```

```
*/
```

4.6 ARQC

getTransactionProgress will return the ARQC after device request a acquirer decision for a transaction.
type : 771 (0x303)
data : hex string contains byte array for following structure

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.

```
{"type":771,"data":"01ABFD8201A7DFDF250F423330364639363039323831354141
FA8201909A030000009F02060000000009999F10120110A00001220000000000000000
000000FF9F1607303030303030309F4E0730303030303030820258008E100000000000
00000042015E0342031F035F24031401315F25031201019F0607A00000000410109F07
02FF009F0D05F0500408009F0E0500008800009F0F05F0700498009F260816A2BD071A
D5DC1E9F2701809F360200C59C01009F33032028C89F34035E03009F37049FA41BDA9F
3901059F4005720000B00195054220008000 9B02E8009F1E084233303646393620 9F1A
0208405F2A0208409F01060000000000019F2103183917DFDF4D263B35343133303030
3034303030313531333D3031313430303030303030303030303030303030303FDFDF520105
F8820078DFDF560A9010010B306F9600001BDFDF570100FA820061DF31182E6ECDA60D
8CB9D382EE13942C18B6DF82EE13942C18B6DFDF321040A47A243903C9DA71DBF06996
3D1D64DF3B30E5B2E54BF31D8DC0BA5C217F756F0CC3D24553AE6B25BD4FFA7C8D9824
3DC66D8C43FEA072300044F6278B858108B825"};
```

4.7 Transaction Result

getTransactionProgress will return the transaction data after device complete a transation
type : 772 (0x304)
data : hex string contains byte array for following structure

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.

```
{"type":772,"data":"010214FE820210DFDF250F42333036463936303932383135 41
41FA8201F9F08201F5F182000FDFDF1A0100DFDF1B0100DFDF520105F28200E39A0300
00009F02060000000009999F10120110600001220000000000000000000000FF9F1607
303030303030309F4E0730303030303030820258008E1000000000000000042015E03
42031F035F24031401315F25031201019F0607A00000000410109F0702FF009F0D05F0
500408009F0E0500008800009F0F05F0700498009F2608C4751D62AEA184A59F270140
9F360200C59C01009F33032028C89F34035E03009F37049FA41BDA9F3901059F400572
0000B001950542200080009B02E8009F1E0842333036463936209F1A0208405F2A0208
409F01060000000000018A023030F782007B5F25031201015F24031401315F2A020840
9F02060000000009999F03060000000000009F0607A00000000410109F1C0831313232
333334349F3901059C01009F34035E03005F2009544553542F43415244DFDF4D263B35
3431333030303034303030313531333D303131343030303030303030303030303030 30
3FF8820078DFDF560A9010010B306F9600001BDFDF570100FA820061DF31182E6ECDA6
0D8CB9D382EE13942C18B6DF82EE13942C18B6DFDF321040A47A243903C9DA71DBF069
963D1D64DF3B30E5B2E54BF31D8DC0BA5C217F756F0CC3D24553AE6B25BD4FFA7C8D98
243DC66D8C43FEA072300044F6278B858108B825"}
```

Appendix A TLV Data Format

A.1 ARQC Message Format

This section gives the format of the ARQC Message delivered in the ARQC Message notification. It is a TLV object with the following contents:

```
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)
                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>
```

A.2 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. It is a TLV object with the following contents:

```
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
                         .
                         .
                         .
```

A.3 Transaction Result Message – Batch Data Format

This section gives the format of the data the device uses to do completion processing

```
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 */

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>
```

Appendix B Cryptography

Appendix C Sample Code

C.1 Query devices, openDevice and closeDevice.

This example shows how to use the getDeviceList function to get devices and open the first device connected to the host.

```
            MTSCRAWCF.MTSCRAWCF wcf = new MTSCRAWCF.MTSCRAWCF();

            var devList = wcf.getDeviceList();

            Console.WriteLine(devList);

            var devs = devList.Split(',');

            foreach (var dev in devs)
                Console.WriteLine(dev);

            var open = wcf.openDevice(devs[0]);

            Console.WriteLine("open " +  devs[0] + " -> " + open);

            if (open == 0)
                wcf.closeDevice();

/*
  Output in Console :
USB://MagTek SCRA 1,BLEEMV://eDynamo-B306F96
USB://MagTek SCRA 1
BLEEMV://eDynamo-B306F96
open USB://MagTek SCRA 1 -> 1
 */
```

C.2 SendCommand to device and get response.

This example shows how to use the SendCommand function to send command to device and get device response.

```
            var response = wcf.sendCommand("0001");
            Console.WriteLine(response);


/*
  Output in Console :
*/
```