IDTECH™

Value through Innovation

USER MANUAL

SecuRED
Encrypted Magstripe Reader
OPOS Reference Guide

80128503-001

Rev A. 9/9/13

ID Technologies, Inc.
10721 Walker Street, Cypress CA 90630 Voice: (714) 761-6368 Fax: (714) 761-8880

Revision History

Revision	Date	Description	By
A	9/9/2013	Initial Release	CH

Table of Contents

ID TECH SOFTWARE COPYRIGHT NOTICE

ID TECH SOFTWARE LICENSE AGREEMENT

ID TECH ("LICENSOR") IS WILLING TO LICENSE THIS SOFTWARE TO YOU ONLY IF YOU ACCEPT ALL OF THE TERMS IN THIS LICENSE AGREEMENT. PLEASE READ THE TERMS CAREFULLY BEFORE YOU AGREE BECAUSE YOU WILL BE BOUND BY THE TERMS OF THIS AGREEMENT. IF YOU DO NOT AGREE TO THESE TERMS, LICENSOR WILL NOT LICENSE THIS SOFTWARE TO YOU.

Ownership of the Software

1. The Licensor software program ("Software") and any accompanying written materials are owned by Licensor [or its suppliers] and are protected by United States copyright laws, by laws of other nations, and by international treaties.

Grant of License

2. Licensor grants to you the right to use the Software in conjunction with an ID TECH product. You may load one copy into permanent memory of one computer and may use that copy only on that same computer.

Restrictions on Use and Transfer

3. You may not copy the Software, except that (1) you may make one copy of the Software solely for backup or archival purposes, and (2) you may transfer the Software to a single hard disk provided you keep the original solely for backup or archival purposes. You may not copy the written materials.

4. You may permanently transfer the Software and any accompanying written materials (including the most recent update and all prior versions) if you retain no copies and the transferee agrees to be bound by the terms of this Agreement. Such a transfer terminates your license. You may not rent or lease the Software or otherwise transfer or assign the right to use the Software, except as stated in this paragraph.

5. You may not reverse engineer, decompile, or disassemble the Software.

Limited Warranty

6. If used in conjunction with an ID TECH product, Licensor warrants that the Software will perform substantially in accordance with the accompanying written materials for a period of 90 days from the date of your receipt of the Software. Any implied warranties on the Software are limited to 90 days. Some states and territories do not allow limitations on duration of an implied warranty, so the above limitation may not apply to you.

7. LICENSOR DISCLAIMS ALL OTHER WARRANTIES, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT, WITH RESPECT TO THE SOFTWARE AND ANY ACCOMPANYING WRITTEN MATERIALS. This limited warranty gives you specific legal rights. You may have others, which vary from state to state.

8. LICENSOR'S ENTIRE LIABILITY AND YOUR EXCLUSIVE REMEDY SHALL BE REPLACEMENT OF THE SOFTWARE THAT DOES NOT MEET LICENSOR'S LIMITED WARRANTY. Any replacement Software will be warranted for the remainder of the original warranty period or 30 days, whichever is longer.

9. This Limited Warranty is void if failure of the Software has resulted from modification, accident, abuse, or misapplication.

10. IN NO EVENT WILL LICENSOR BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY LOSS OF PROFITS, LOST SAVINGS, OR OTHER INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF YOUR USE OR INABILITY TO USE THE SOFTWARE. Because some states do not allow the exclusion or limitation of liability for consequential or incidental damages, the above limitation may not apply to you.

11. This Agreement is governed by the laws of the state of California.

12. If you have any questions concerning this Agreement or wish to contact Licensor for any reason, please write: ID TECH, 10721 Walker Street, Cypress, CA 90630 or call (714) 761-6368.

13. U.S. Government Restricted Rights. The Software and documentation are provided with Restricted Rights. Use, duplication, or disclosure by the Government is subject to restrictions set forth in subparagraph (c)(1) of The Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 or subparagraphs (c)(1)(ii) and (2) of Commercial Computer Software - Restricted Rights at 48 CFR 52.227-19, as applicable. Supplier is ID TECH, 10721 Walker Street, Cypress, CA 90630.

1. Description

The documentation describes the properties, methods, and events of the ID TECH SecuRED MSR OPOS component. The component includes two parts: a Control Object running on the upper level, which is an ActiveX control, and a Service Control running on the lower level, which is an OLE automation server. The properties, methods, and events are exposed by the Control Object. When the Control Object is imported into your project as an ActiveX control, you will see all the properties, methods, and events.

For different interface devices, OPOS drivers may be different. For USB HID Keyboard interfaces device, the standard keyboard should not be pressed when swiping cards, otherwise the card data will be wrong , MSR OPOS Driver will display a warning dialog and the data will be discarded.

The SecuRED MSR device can’t support hot plug when OPOS driver is in the Open state. If you have already pulled out the device in Open state, close driver and reopen can use again.

For the same interface SecuRED MSR devices, the OPOS supports only one device for use on a computer. In other words: at the same time, the OPOS only allows to connect one device.

If the SecuRED MSR Device has been authenticated, the application should cancel the authentication before close the OPOS. Otherwise, the device will stay in the authentication state 120 seconds.

Target Device:
ID TECH SecuRED
USB-HID, USB-KB interface

Platform:
Microsoft Windows 8,Windows 7,Vista, XP, 2000, 98.
Service Object and Control Object:
Service Object Version: 1.13.309
Control Object Version: 1.13.001
Dll File Version: 3.0.9

2. Methods, Properties and Events of SecuRED

This section describes methods, properties and events for the SecuRED Encrypted MSR.

2.1. Methods of MSR

These function declarations may be different when the Control Object (OPOSMSR.OCX) is imported into your application project. Please refer to the UnifiedPOS Specification for more detailed information on the Control Object.

1) Open

Syntax **LONG Open (BSTR** *DeviceName***);**

DeviceName:

For USB HID interface: "IDTECH_ SECURED _USBHID"

For USB KB interface: "IDTECH_ SECURED _USBKB"

Remarks Call to open a device for subsequent I/O.

Support? Yes

Description This method finds more parameters in the Windows Register Tables on key or sub keys.

For USB HID interface:

HKEY_LOCAL_MACHINE\Software\OLEforRetail\ServiceOPOS\MSR\ IDTECH_ SECURED _USBHID\ CONNECTOR

Key value name: USBHID
Key value: "usbhidConn.dll"
Key value name: CONNECTOR
Key value: "USBHID/0acd/2810"
First field USBHID specify the type of the connector. 0acd is the USB device vendor ID, 2810 is the reader product ID.

For USB KB interface:

HKEY_LOCAL_MACHINE\Software\OLEforRetail\ServiceOPOS\MSR\ IDTECH_ SECURED _USBKB\ CONNECTOR

Key value name: USBKB
Key value: "usbkbConnector.dll"
Key value name: CONNECTOR
Key value: "USBKB/0acd/2820"
First field USBKB specify the type of the connector. 0acd is the USB device vendor ID, 2820 is the reader product ID.

2) ClaimDevice Added in Release 1.5

Syntax **LONG ClaimDevice (LONG** *Timeout***);**

Remarks Call this method to request exclusive access to the device. Many devices require an application to claim them before they can be used. ***Release 1.0 – 1.4*** in releases prior to 1.5, this method is named Claim.

Support? Yes

3) CheckHealth

Syntax **LONG CheckHealth (LONG** *Level***);**

Remarks Called to test the state of a device.

Support? Yes

Description When select CH_INTERNAL, check the SO response, if not it tells that there is something wrong with the device. CheckHealthText property will be “Internal HCheck: Successful”

When select CH_EXTERNAL, SO will return the firmware version of the SecuRED device, if reading the firmware version is successful. CheckHealthText property will be “External HCheck: Successful” + firmware version information. If Not Responding, CheckHealthText property will be “External HCheck: Not Responding”.

When select CH_INTERACTIVE , SO will display a dialog, which include firmware version and swiping card. And it can display the “Real data” of the card; include Start Sentinel and End Sentinel. CheckHealthText property will show “External HCheck:: HCheck: Complete”, after the dialog is closed.

4) ClearInput

Syntax **LONG ClearInput ();**

Remarks Called to clear all device input that has been buffered.

Support? Yes

5) DirectIO

Syntax **LONG DirectIO (LONG** *Command*, **LONG*** *pData*, **BSTR*** *pString***);**

Remarks Call to communicate directly with the Service Object.

Support? No

Description In the current, it implemented incompletely**.** We will improve it in the next release.

6) ReleaseDevice ***Added in Release 1.5***

Syntax **LONG ReleaseDevice ();**

Remarks Call this method to release exclusive access to the device.
Release 1.0 – 1.4
In releases prior to 1.5, this method is named **Release**.

Support? Yes

7) Close

Syntax **LONG Close ();**

Remarks Called to release the device and its resources.

Support? Yes

8) ResetStatistics *Added in Release 1.8*

Syntax **LONG ResetStatistics(BSTR m_StatisticsBuffer);**

Remarks Called to Resets the defined resettable statistics in a device to zero.

Support? No

9) RetrieveStatistics *Added in Release 1.8*

Syntax **LONG RetrieveStatistics(BSTR* m_pStatisticsBuffer);**

Remarks Called to Retrieves the requested statistics from a device.

Support? No

10) UpdateStatistics *Added in Release 1.8*

Syntax **LONG UpdateStatistics(BSTR m_StatisticsBuffer);**

Remarks Called to Updates the defined resettable statistics in a device.

Support? No

11) CompareFirmwareVersion

Syntax **LONG CompareFirmwareVersion(BSTR m_FirmwareFileName, long* m_pResult);**

Remarks Called to compare the firmware version with current firmware version of the device.

Support? No

12) UpdateFirmware

Syntax **LONG UpdateFirmware(BSTR m_FirmwareFileName);**

Remarks Called to update current firmware.

Support? No

13) ClearInputProperties

Syntax **void ClearInputProperties();**

Remarks Sets all data properties that were populated as a result of firing a DataEvent or ErrorEvent back to their default values.

Support? Yes

14) WriteTracks

Syntax **long WriteTracks(LPCTSTR data, long timeout)**

Remarks Sets all data properties that were populated as a result of firing a DataEvent or ErrorEvent back to their default values.

Support? No

15) AuthenticateDevice

Syntax **long AuthenticateDevice (LPCTSTR response)**

Remarks To authenticate a device, the application first calls the retrieveDeviceAuthenticationData method to retrieve a challenge token from the device. The application then typically passes this token to another entity that has special knowledge of a shared secret and is able to create a proper response token. This response token is then passed as the response parameter to this method and the service uses it to validate the authentication request. If this method succeeds, the device enters the authenticated state and the service sets the DeviceAuthenticated property to true.

For SecuRED: The response needs to be 16 bytes in length. And it should be transmitted as a Hex string. Example, 0xAB 0x00 0x09 is converted to "AB0009".

Support? Yes

16) DeauthenticateDevice

Syntax **long DeauthenticateDevice (LPCTSTR response)**

Remarks This method is used to deauthenticate a device that is currently in the authenticated state (DeviceAuthenticated = true). The token is typically generated by passing the challenge retrieved from the retrieveDeviceAuthenticationData method to an entity that has special knowledge of a shared secret. If this method succeeds the service sets DeviceAuthenticated to false and enqueues a StatusUpdateEvent with status value set to MSR_SUE_DEVICE_DEAUTHENTICATED.

For SecuRED: The response needs to be 16 bytes (when Encryption Algorithm is 3DES) or 8 bytes (when Encryption Algorithm is AES) in length. And it should be transmitted as a Hex string. Example, 0xAB 0x00 0x09 is converted to "AB0009".

Support? Yes

17) RetrieveCardProperty

Syntax **long RetrieveCardProperty (BSTR** *Name*, **BSTR ****Value***)**

Remarks Retrieves the value of specific parsed properties from the last card swiped.

Support? Yes

18) RetrieveDeviceAuthenticationData

Syntax **long RetrieveDeviceAuthenticationData (LPCTSTR** *challenge***)**

Remarks Applications call this method to retrieve a challenge token that will subsequently be used to generate response tokens that will be passed to the authenticateDevice and deauthenticateDevice methods. The challenge token is typically sent to another entity that has special knowledge of a shared secret that is required to generate the proper response token(s).

For SecuRED: The challenge is always 26 bytes in length. And it will be transmitted as a Hex string. Example, 0xAB 0x00 0x09 is converted to "AB0009".

Support? Yes

19) UpdateKey

Syntax **long UpdateKey (BSTR** *Key*, **BSTR** *KeyName***)**

Remarks Provides a new encryption key to the device. It is used only for those encryption algorithms in which new key values are sent to the terminal as a field in standard messages from the host.

Support? NO

2.2. Properties of MSR

Please refer to the UnifiedPOS Specification for detailed information.

NOTE: CO --- Control Object
SO --- Service Object
AP or App --- the abbreviation of Application.

Property Group1---Description

Name	*Type*	*Mutability*	*Use After*	*Description*	*Support?*
DeviceControlDescription	*String*	read-only	--	Identify the Control Object and the company that produced it	Yes
DeviceControlVersion	*int32*	read-only	--	Hold the Control Object version number.	Yes
DeviceServiceDescription	*String*	read-only	open	Identify the Service Object supporting the device and the company that produced it	Yes
DeviceServiceVersion	*int32*	read-only	open	Hold the Service Object version number.	Yes
PhysicalDeviceDescription	*string*	read-only	open	Identify the device and any pertinent information about it.	Yes
PhysicalDeviceName	*string*	read-only	open	Identify the device and any pertinent information about it.	Yes

Property Group2---Control

Name	*Type*	*Mutability*	*Use After*	*Description*	*Support?*
Claimed	*Boolean*	read-only	open	SecuRED must be claimed for exclusive use before access its methods and properties, and before any events to be fired. It is initialized to FALSE by the **Open** method. It is set to TRUE after the method **Claim** is successfully called.	Yes
AutoDisable	*Boolean*	read-write	open	When TRUE, as soon as an event **DataEvent** is received, then **DeviceEnabled** is automatically to FALSE. It is initialized to FALSE by the **Open** method.	Yes
DeviceEnabled	*Boolean*	read-write	open& claim	When FALSE, SecuRED has been disabled and any subsequent input will be discarded (No DataEvent could be received even if the card is swiped). It is initialized to FALSE by the **Open**	Yes

				method.	
FreezeEvents	*boolean*	read-write	open	When TRUE, events are not required to be delivered and will be held by SO until events are unfrozen. It is initialized to FALSE by the **Open** method.	Yes
DataEventEnabled	*boolean*	read-write	open	When TRUE, a **DataEvent** or **ErrorEvent** will be delivered immediately when had. (Of course, ***FreezeEvents=FALSE and DeviceEnabled=TRUE*** is a prerequisite). It is initialized to FALSE by the **Open** method.	Yes
CapPowerReporting	*int32*	read-only	open	Identifies the reporting capabilities of the device about Power. It seems that SecuRED doesn't support in the hardware.	No
PowerNotify	*int32*	read-write	open	Contains the type power notification selection made by the Application. is initialized to OPOS_PN_DISABLED by the **Open** method.	No
PowerState	*int32*	read-only	open	Contains the current power condition. It seems that SecuRED doesn't support in the hardware.	No
State	*int32*	Read-only	--	Contains the current state of the Control. It can be set to one of the four Values: Closed, Idle, Busy, or Error.	Yes
DataCount	*int32*	Read-only	open	Holds the number of queued **DataEvent**s remained in the queue.	Yes
CheckHealthText	*string*	read-only	open	Holds the results of the most recent call to the **CheckHealth** method. Before the first **CheckHealth** method call, its value is uninitialized.	Yes

Property Group3---Track Control

Name	*Type*	*Mutability*	*Use After*	*Description*	*Support?*
CapISO	*boolean*	read-only	open	If TRUE, SecuRED supports ISO cards.	Yes
CapJISOne	*boolean*	read-only	open	If TRUE, SecuRED supports JIS Type-I cards. JIS-I cards are a superset of ISO cards. Therefore, if **CapJISOne** is true, it is implied that **CapISO** is also TRUE.	No
CapJISTwo	*boolean*	read-only	open	If TRUE, SecuRED supports JIS type-II cards.	No
CapTransmitSentinels	*boolean*	read-only	open	If TRUE, SecuRED is able to transmit the start and end sentinels. e.g. start sentinel could be '%' or ';', and stop sentinel could be '?'.	Yes
DecodeData	*boolean*	read-write	open	If TRUE, each byte of track data properties is mapped from its original encoded bit sequence (as it exists on the magnetic card) to its corresponding decoded ASCII bit sequence.	Yes
ParseDecodeData	*boolean*	read-write	open	When TRUE, the decoded data contained within the **Track1Data** and **Track2Data** properties is further separated into fields for access via various other properties. If ***DecodeData=FALSE***, ***ParseDecodeData must be false.***	Yes
TransmitSentinels	*boolean*	read-write	open	If TRUE, the	Yes

				Track1Data, **Track2Data**, **Track3Data**, and **Track4Data** properties contain start and end sentinel values. Otherwise only the track data between these sentinels.	
TracksToRead	*int32*	read-write	open	Indicate which track data that the App wishes to get following a card sweep**.**	Yes
ErrorReportingType	*int32*	Read-write	open	Holds the type of errors to report via **ErrorEvent**s. This property has one of the following values: MSR_ERT_CARD or MSF_ERT_TRACK	Yes

Property Group4---TrackData

Name	***Type***	***Mutability***	***Use After***	***Description***	***Support?***
Track1Data	*binary*	read-only	open	Holds the track 1 data obtained from the most recently swept card. If **DecodeData** is true, then it has been decoded from the "raw" format. it may also be parsed into other properties when the **ParseDecodeData** property is set.	Yes
Track1DiscretionaryData	*binary*	read-only	open	Holds the track 1 discretionary data obtained from the most recently swept card. It may be NULL when: **1)** The field was not included in the track data obtained, or, **2)** The track data format was	Yes

				not supported, **3) ParseDecodeData** is false.	
Track2Data	*binary*	read-only	open	Holds the track 2 data obtained from the most recently swept card. If **DecodeData** is true, then it has been decoded from the “raw” format. it may also be parsed into other properties when the **ParseDecodeData** property is set.	Yes
Track2DiscretionaryData	*binary*	read-only	open	Holds the track 2 discretionary data obtained from the most recently swept card. It may be NULL when: **1)** The field was not included in the track data obtained, or, **2)** The track data format was not supported, **3) ParseDecodeData** is false.	Yes
Track3Data	*binary*	read-only	open	Holds the track 3 data obtained from the most recently swept card.	Yes
Track4Data	*binary*	read-only	open	Holds the track 4 data (JIS-II) obtained from the most recently swept card.	No
Track1EncryptedData	*binary*	read-only	Open	Holds the encrypted track 1 data obtained from the most recently swiped card. The start and end sentinel values are contained in it, and appear only after data is decrypted. Encrypted data is always a multiple of 8 bytes (when Encryption Algorithm is 3DES) or 16 bytes (when Encryption Algorithm is AES) in length. And it will be transmitted as a Hex string. Example, 0xAB 0x00 0x09 is converted to	Yes

				"AB0009".	
Track1EncryptedData Length	*int32*	read-only	Open	Holds the length of the raw track 1 data before it was encrypted.	Yes
Track2EncryptedData	*binary*	read-only	Open	Holds the encrypted track 2 data obtained from the most recently swiped card. The start and end sentinel values are contained in it, and appear only after data is decrypted. Encrypted data is always a multiple of 8 bytes (when Encryption Algorithm is 3DES) or 16 bytes (when Encryption Algorithm is AES) in length. And it will be transmitted as a Hex string. Example, 0xAB 0x00 0x09 is converted to "AB0009".	Yes
Track2EncryptedData Length	*int32*	read-only	Open	Holds the length of the raw track 2 data before it was encrypted.	Yes
Track3EncryptedData	*binary*	read-only	Open	Holds the encrypted track 3 data obtained from the most recently swiped card. The start and end sentinel values are contained in it, and appear only after data is decrypted. Encrypted data is always a multiple of 8 bytes (when Encryption Algorithm is 3DES) or 16 bytes (when Encryption Algorithm is AES) in length. And it will be transmitted as a Hex string. Example, 0xAB 0x00 0x09 is converted to "AB0009".	Yes
Track3EncryptedData Length	*int32*	read-only	Open	Holds the length of the raw track 3 data before it was	Yes

				encrypted.	
Track4EncryptedData	*binary*	read-only	Open	Holds the encrypted track 4 data obtained from the most recently swiped card.	No
Track4EncryptedDataLength	*binary*	read-only	Open	Holds the length of the raw track 4 data before it was encrypted.	No
AdditionalSecurityInformation	*binary*	read-only	Open	Holds additional security/encryption information when a DataEvent is delivered. For example “DUKPT sequence number” in it. This data is always 10 bytes in length. And it will be transmitted as a Hex string. Example, 0xAB 0x00 0x09 is converted to "AB0009".	Yes
CardAuthenticationData	*binary*	read-only	Open	Holds card authentication information when a **DataEvent** is delivered.	No
CardAuthenticationDataLength	*int32*	read-only	Open	This property will be zero if CapCardAuthentication is an empty string.	No
DeviceAuthenticated	*boolean*	read-only	Open & Claim & Enable	If the device supports authentication the service must keep the value of this property up to date when the device is enabled. MSR_SUE_DEVICE_AUTHENTICATED or MSR_SUE_DEVICE_DEAUTHENTICATED.	Yes
CardType	*string*	read-only	open	Holds the card type identifier for the most recently swiped card. Value is one of them (“BANK”,”AAMVA” and empty).	Yes
CardTypeList	*string*	read-only	open	Holds a comma separated list of string names of card types supported by the Service. Value is BANK and AAMVA.	Yes

CardPropertyList	*string*	read-only	open	Holds a comma separated list of the names of the properties parsed from the most recently swiped card.	Yes

Property Group5---ParsedData

Name	***Type***	***Mutability***	***Use After***	***Description***	***Support?***
AccountNumber	*string*	read-only	Open	Holds the account number obtained from the most recently swept card. it is initialized to NULL if: **1)** The field was not included in the track data obtained, or, **2)** The track data format was not supported, or, **3) ParseDecodeData** is false.	Yes
ExpirationData	*string*	read-only	Open	Holds the expiration date obtained from the most recently swept card. Others are same as ***AccountNumber.***	Yes
FirstName	*string*	read-only	Open	Holds the first name obtained from the most recently swept card. Others are same as ***AccountNumber.***	Yes
MiddleInitial	*string*	read-only	Open	Holds the middle initial obtained from the most recently swept card. Others are same as ***AccountNumber.***	Yes
Surname	*string*	read-only	Open	Holds the surname obtained from the most recently swept card. Others are same as ***AccountNumber.***	Yes
Title	*string*	read-only	Open	Holds the title obtained from the most recently swept card.. Others are same as ***AccountNumber.***	Yes
Suffix	*string*	read-only	Open	Holds the suffix obtained from the most recently swept card.. Others are same as ***AccountNumber.***	Yes

ServiceCode	*string*	read-only	Open	Holds the service code obtained from the most recently swept card. Others are same as ***AccountNumber.***	Yes

Property Group6--- Statistic

Name	***Type***	***Mutability***	***Use After***	***Expected Result***	***Test Result***
CapStatisticsReporting	*boolean*	read-write	Open	If true ,the SO can get device information to a XML statistics	No
CapUpdateStatistics	*boolean*	read-write	Open	If true ,the SO can update the XML statistics	No

Property Group7---Firmware

Name	***Type***	***Mutability***	***Use After***	***Expected Result***	***Test Result***
CapCompareFirmwareVersion	*boolean*	read-write	Open	If true ,the SO can compare the Firmware version	No
CapUpdateFirmware	*boolean*	read-write	Open	If true ,the SO can update the firmware of the device	No
CapWritableTracks	*Int32*	read_only	Open	This capability indicates if the SecuRED device supports the writing of track data - and which tracks are supported.	No
EncodingMaxLength	*Int32*	read_only	Open	The maximum length of data that can be written by the SecuRED to the track(s).	No
TracksToWrite	*Int32*	Read-Write	Open	Holds the SecuRED track(s) that will be written.	No
CapDataEncryption	*Int32*	read_only	Open	Holds a bitwise indication of the encryption algorithms supported by the device and selectable via the	Yes

				DataEncryptionAlgorithm property. MSR_DE_NONE: Data encryption is not enabled. MSR_DE_3DEA_DUKPT: Triple DES **D**erived **U**nique **K**ey **P**er **T**ransaction. MSR_DE_AES_DUKPT (value:3): Advanced Encryption Standard **D**erived **U**nique **K**ey **P**er **T**ransaction.	
DataEncryptionAlgorithm	*Int32*	Read-Write	Open & Claim	Holds the encryption algorithm that will be used to encrypt the track data. This property may be set to one of the supported encryption algorithms as defined in the CapDataEncryption property. MSR_DE_NONE: Data encryption is not enabled.	Yes
CapTrackDataMasking	*boolean*	Read_only	Open	This value will be true if the Service is capable of masking track data.	Yes
CapCardAuthentication	*string*	Read_only	Open	Holds the type, if any, of card authentication data that is supported by the device.	No
CapDeviceAuthentication	*Int32*	Read_only	Open	Holds the level of device authentication supported by the service. MSR_DA_NOT_SUPPORTED: The service does not support device authentication. MSR_DA_OPTIONAL: The service supports device authentication but does not require it. MSR_DA_REQUIRED: The service requires device authentication.	Yes
DeviceAuthenticationProtocol	*Int32*	Read_only	Open	Holds the device authentication protocol supported by the device. MSR_AP_NONE: The service does not support device	Yes

				authentication. MSR_AP_CHALLENGERESPONSE: The service supports the challenge response protocol.	
WriteCardType	*string*	Read-Write	Open	Holds the card type to be used the next time the write Tracks method is called.	No

2.3. Events of MSR

These events are fired by the Service Object when it is necessary. The following functions are, in fact, the event-handlers that can be added into the applications. Then the applications can receive these events and do some processing accordingly. Please refer to the UnifiedPOS Specification for detailed information.

1) DataEvent

Syntax **void DataEvent (LONG** *Status***);**

The *Status* parameter contains the input status. Its value is Control-dependent. And it may describe the type or qualities of the input.

Remarks Fired to present input data from the device to the application.

Description a **DataEvent** can be received when a magnetic card is swiped if the three conditions are all met:

1) **DeviceEnabled** = TRUE
2) **FreezeEvents** = FALSE
3) **DataEventEnabled** = TRUE.

The track data can be obtained, and the parsed data can also be obtained if ParseDecodeData is TRUE.

Support? Yes

2) DirectIO Event

Syntax **void DirectIOEvent (LONG** *EventNumber,* **LONG*** *pData*, **BSTR*** *pString***);**

Parameter Description

EventNumber Event number. Specific values are assigned by the Service Object.

pData Pointer to additional numeric data. Specific values vary by *EventNumber* and the Service Object.

pString Pointer to additional string data. Specific values vary by *EventNumber* and the Service Object.

Remarks Fired by a Service Object to communicate directly with the application.

Description The event **DirectIOEvent** is used for some special communication between one SO and an application. Currently, this event is not fully implemented.

Support? No

3) Error Event

Syntax **void ErrorEvent (LONG** *ResultCode*, **LONG** *ResultCodeExtended*,
LONG *ErrorLocus***, LONG*** *pErrorResponse***);**

Parameter Description

ResultCode	Result code causing the error event. See **ResultCode** for values.
ResultCodeExtended	Extended result code causing the error event. See **ResultCodeExtended** for values.
ErrorLocus	Location of the error. See values below.
PErrorResponse	Pointer to the error event response. See values below.

When **ErrorReportingType** property is MSR_ERT_TRACK, and *ErrorCode* is E_EXTENDED, then *ErrorCodeExtended* contains Track-level status, broken down as follows:

Byte3	Byte2	Byte1	Byte0
Track 4	Track 3	Track 2	Track 1

Remarks Fired when an error is detected and the Control's **State** transitions into the error state.

NOTICE: The error type is only one E_FAILURE (Other or general error) while any error is raised from reading card of SecuRED device. Because the SecuRED hardware cannot support discerning wrong type.

Support? Yes

4) StatusUpdate Event

Syntax **void StatusUpdateEvent (LONG** *Status***);**

The *Status* parameter is for device class-specific data, describing the type of status change.

Remarks Fired when a Control needs to alert the application of a device status change.

Note The SecuRED hardware cannot support the notification of power status change.

Description It is not implemented by the SO for the power status cannot be inquired from the SecuRED.

Support? No

3. Programming Examples

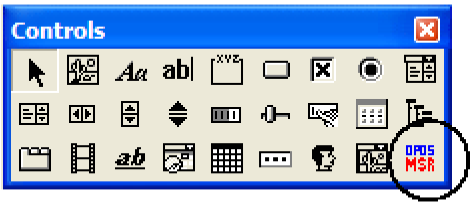
There are three simple programming simple examples provided in this section including VC++6.0, VB6.0, and VS2005/2008 C#. The examples include basic operations and event handling.

In general, there are two steps to work with the OPOS control object:
1. Insert the OPOS Control Object (CO) into the project
2. Add an event handle

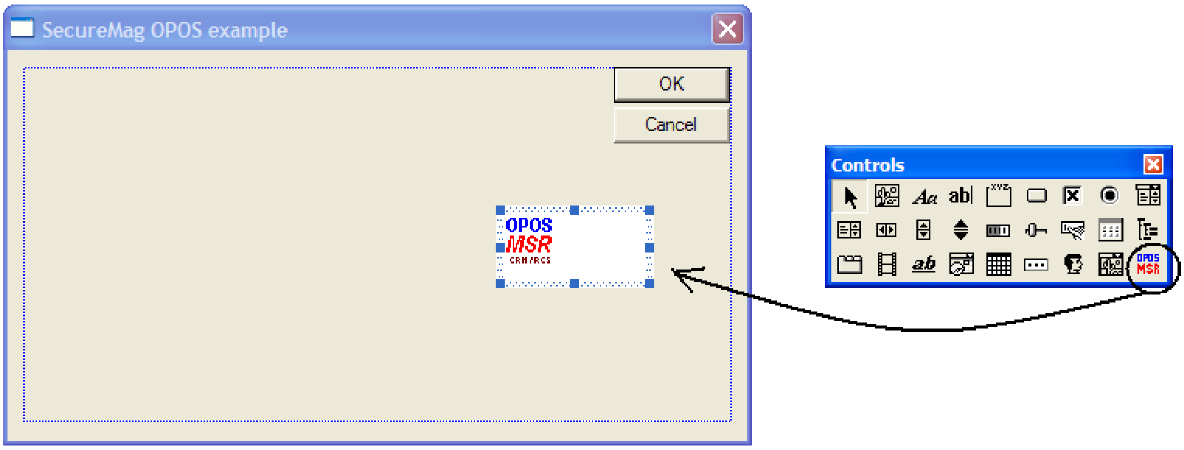
3.1. Visual C++ 6.0 Programming Example

Programming Environment:
Windows XP Pro, Visual C++ 6.0, OPOS CO 1.13. ID TECH SO 1.13.307

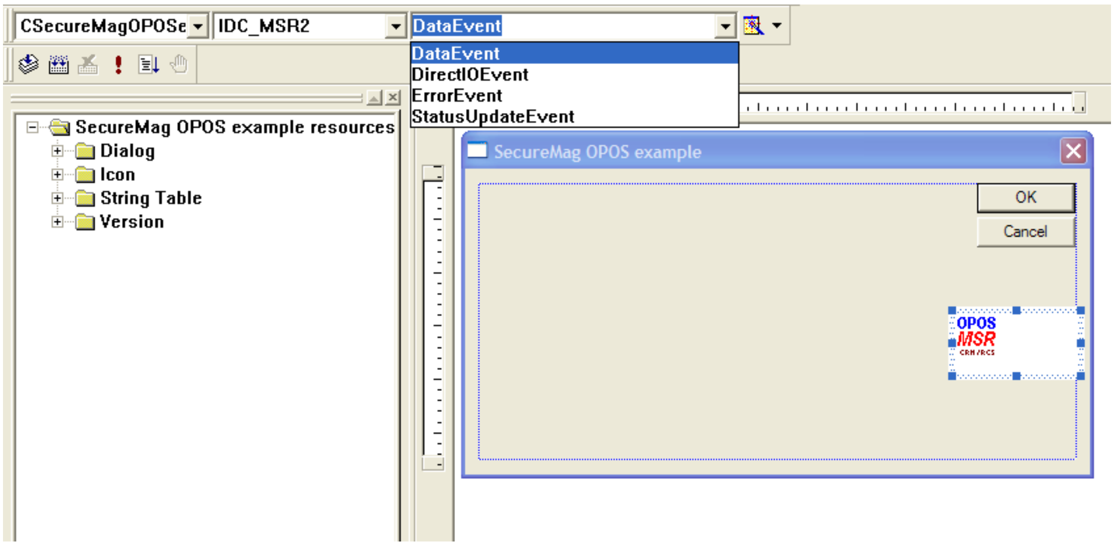
1. Download the OPOS driver and demo from the IDTECH website www.idtechproducts.com. Install the driver and make sure the OPOS demo is functioning.
2. In Visual C++ 6.0, create a Dialog Based MFC application using MFC Application Wizard with ActiveX supports.
3. Go to Project → Add to Project → Components and Controls. From the “Registered ActiveX Controls” folder, select “OPOS MSR Control 1.13.001”, and insert this ActiveX control into the project. An icon for OPOS MSR will be added to the Controls toolbar.



4. Add the OPOS MSR CO to the project dialog.

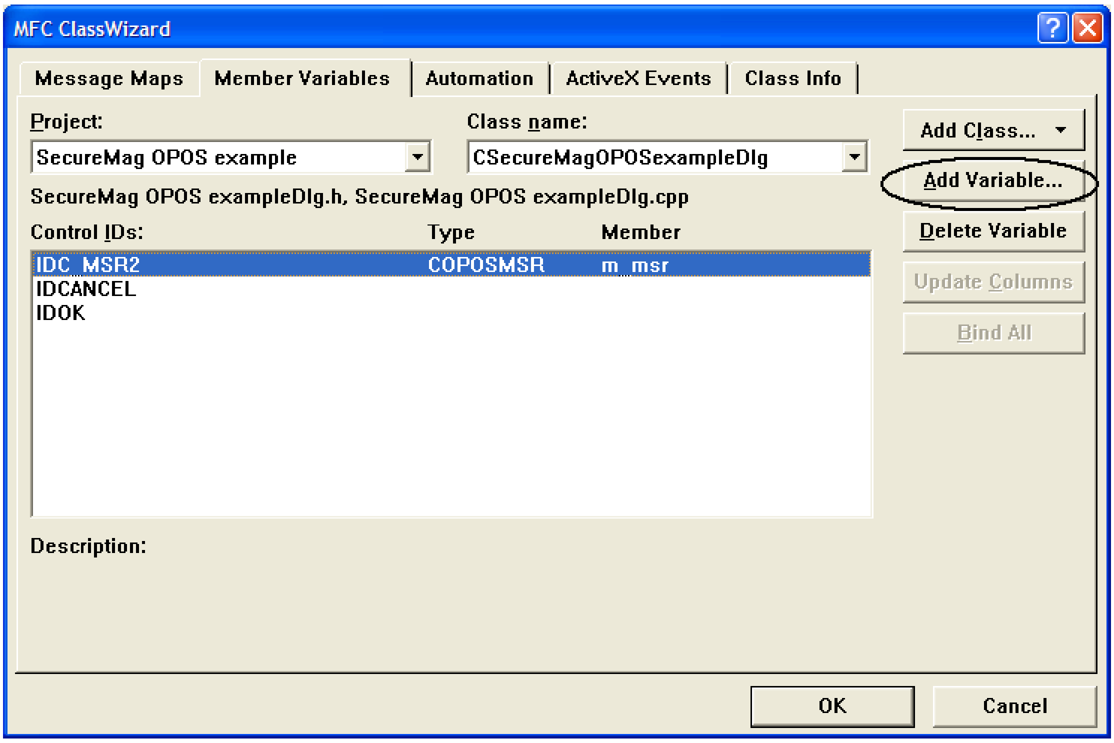


5. Add DataEvent and ErrorEvent handle



```
void CMfc_diagDlg::OnDataEventMsr1(long Status)
```

6. Go the View->ClassWizard and select the “Member Variables” tab. Select IDC_MSR and add a member variable of type “COPOSMSR”, name it *m_msr*.



Project: SecureMag OPOS example

Class name: CSecureMagOPOSexampleDlg

SecureMag OPOS exampleDlg.h, SecureMag OPOS exampleDlg.cpp

Control IDs:	Type	Member
IDC_MSR2	COPOSMSR	m_msr
IDCANCEL		
IDOK		

Description:

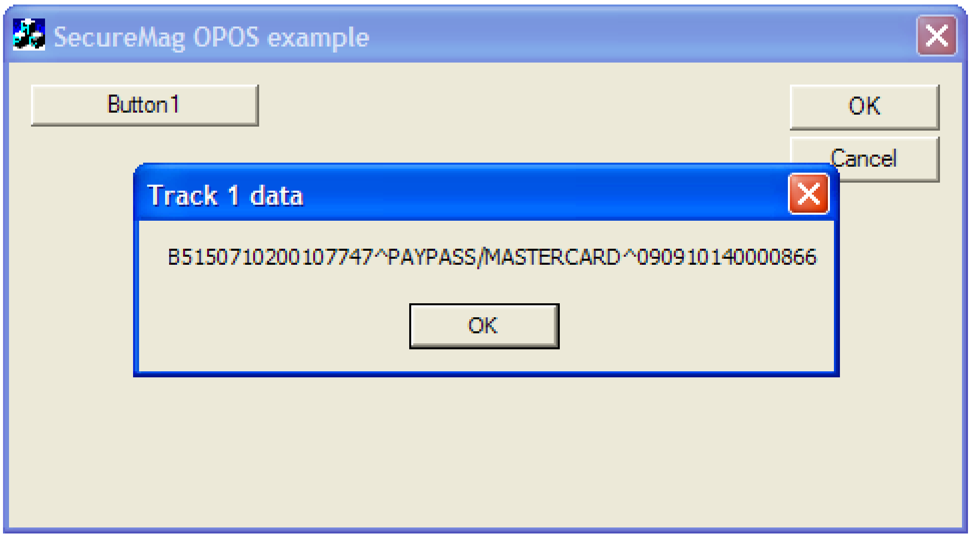
7. Create a button on the form and add the following initialization code:

```
void CMfc_diagDlg::OnButton1()
{
if (m_msr.Open("IDTECH_SecuRED_USBKB") == 0)
{
    m_msr.ClaimDevice(100);
    m_msr.SetDeviceEnabled(TRUE);
    m_msr.SetDataEventEnabled(TRUE);
    }
    else {
         // something wrong ...
    }
}
```

8. Add code for DataEvent handle

```
  void CMfc_diagDlg::OnDataEventMsr1(long Status)
  {
    MessageBox(m_msr.GetTrack1Data(), "Track 1 data");
    m_msr.SetDataEventEnabled(TRUE); // prepare the next event.
  }
```

9. Compile and run the program. Compile and run the program. Click on “Button1” to initialize the reader and swipe a card. Track 1 data will show up in a message box.

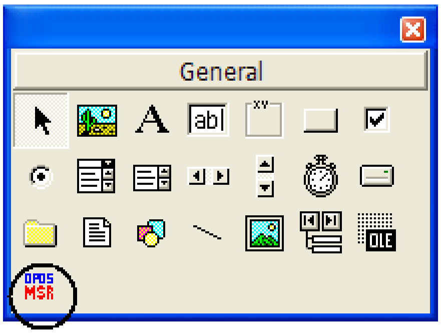


B5150710200107747^PAYPASS/MASTERCARD^090910140000866

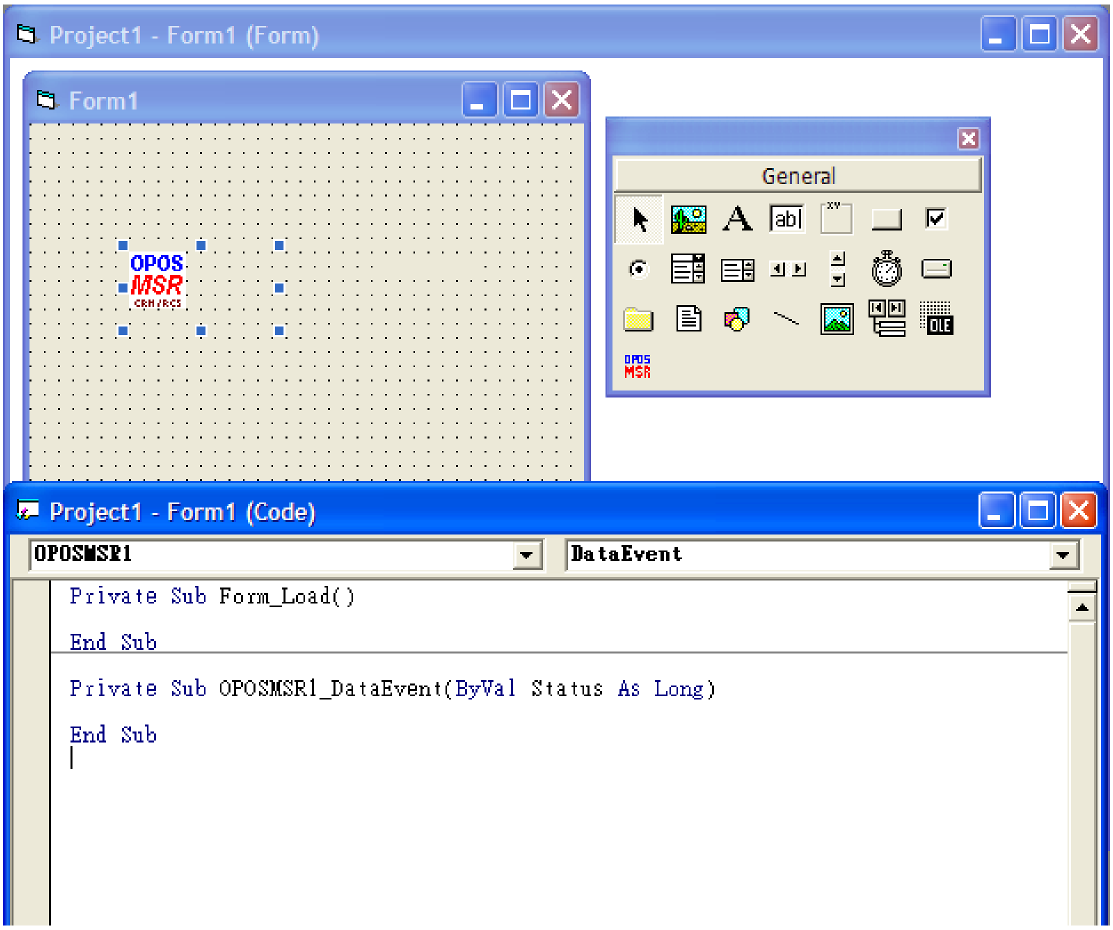
3.2. Visual Basic 6.0 Programming Example

Programming Environment:
Windows XP Professional, Visual Basic 6.0, OPOS CO 1.13. ID TECH SO 1.13.307

1. Create a new project of type “Standard EXE”.
2. From Project->Components, select “OPOS MSR Control 1.13.001” and click “apply”. The OPOS MSR icon will be added to the control toolbar.



3. Add an OPOS MSR control to the form. Double click on the control to add "DataEvent" handle.



```
Private Sub Form_Load()
End Sub
Private Sub OPOSMSR1_DataEvent(ByVal Status As Long)
End Sub
```

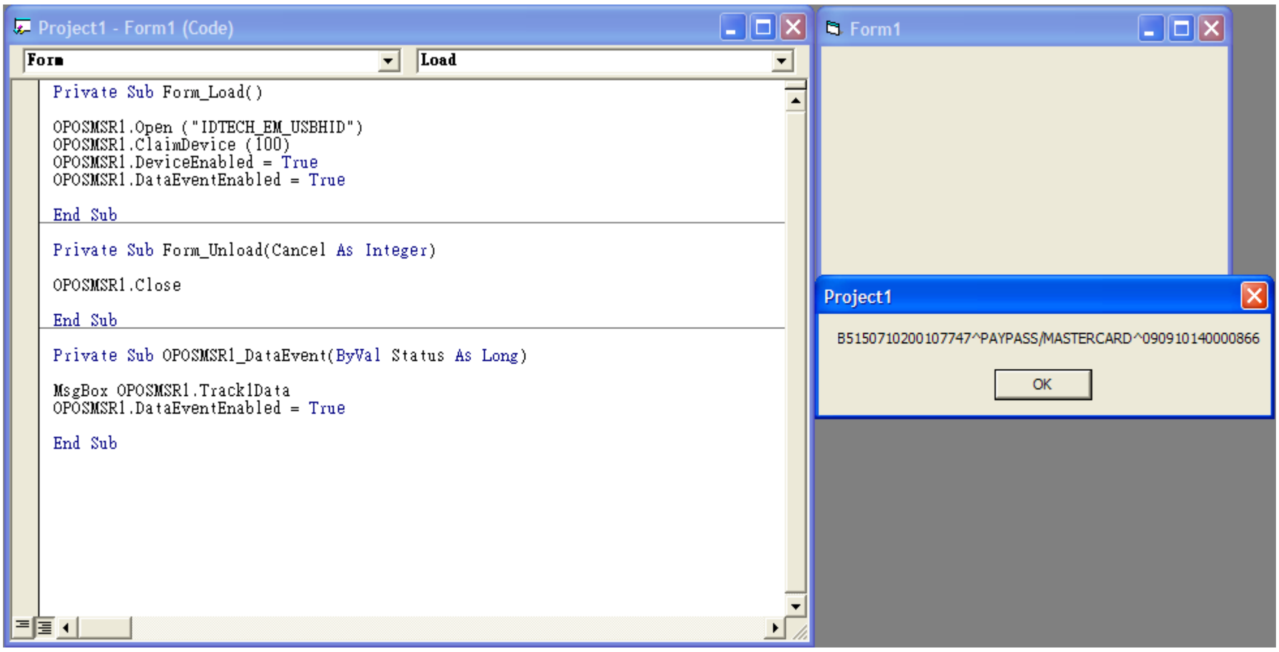
4. Add the initialization code:

```
Private Sub Form_Load()
OPOSMSR1.Open("IDTECH_SecuRED_USBKB")
OPOSMSR1.ClaimDevice (100)
OPOSMSR1.DeviceEnabled = True
OPOSMSR1.DataEventEnabled = True
End Sub
```

5. Add the code for Event Handle

```
Private Sub OPOSMSR1_DataEvent(ByVal Status As Long)
MsgBox OPOSMSR1.Track1Data
OPOSMSR1.DataEventEnabled = True
End Sub
```

6. Run program and swipe a card. The track 1 data will show up in a message box.



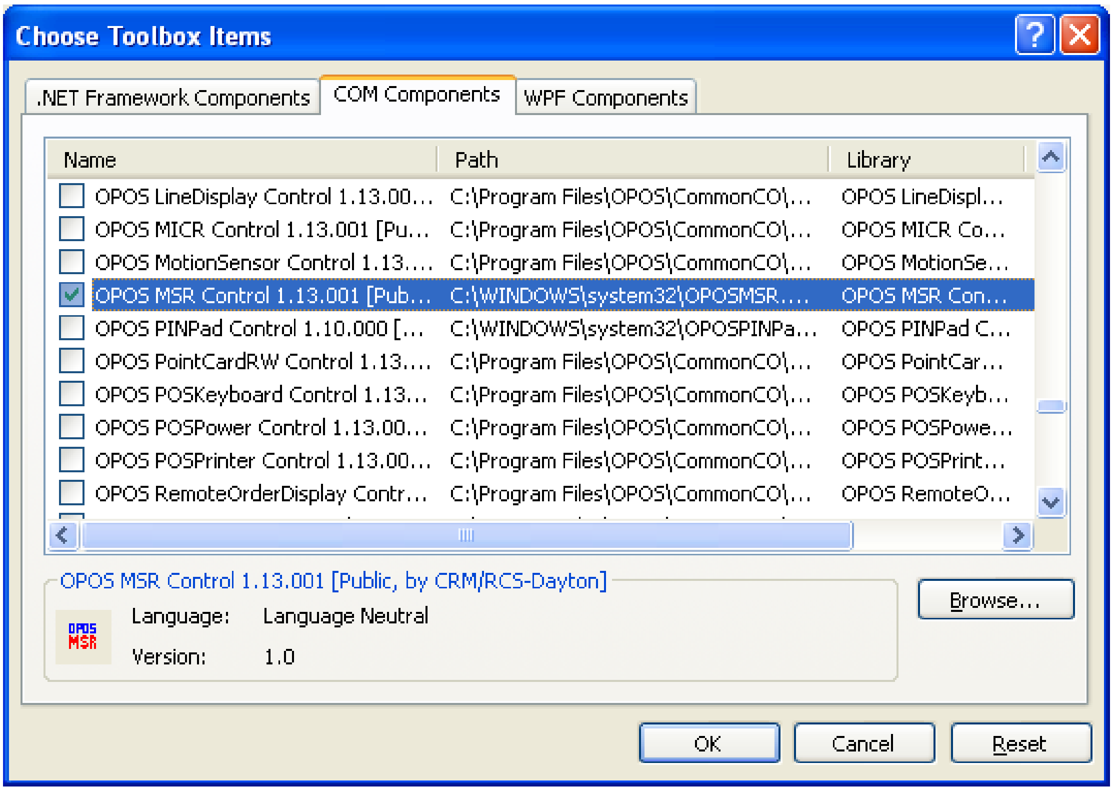
```
Private Sub Form_Load()

OPOSMSR1.Open ("IDTECH_EM_USBHID")
OPOSMSR1.ClaimDevice (100)
OPOSMSR1.DeviceEnabled = True
OPOSMSR1.DataEventEnabled = True

End Sub

Private Sub Form_Unload(Cancel As Integer)

OPOSMSR1.Close

End Sub

Private Sub OPOSMSR1_DataEvent(ByVal Status As Long)

MsgBox OPOSMSR1.Track1Data
OPOSMSR1.DataEventEnabled = True

End Sub
```

```
B5150710200107747^PAYPASS/MASTERCARD^090910140000866
```

3.3. Visual Studio 2005/2008 C# Programming Example

Programming Environment:
Windows XP Professional, Visual Studio 2005/2008 C#, OPOS CO 1.13.001 ID TECH SO 1.13.307

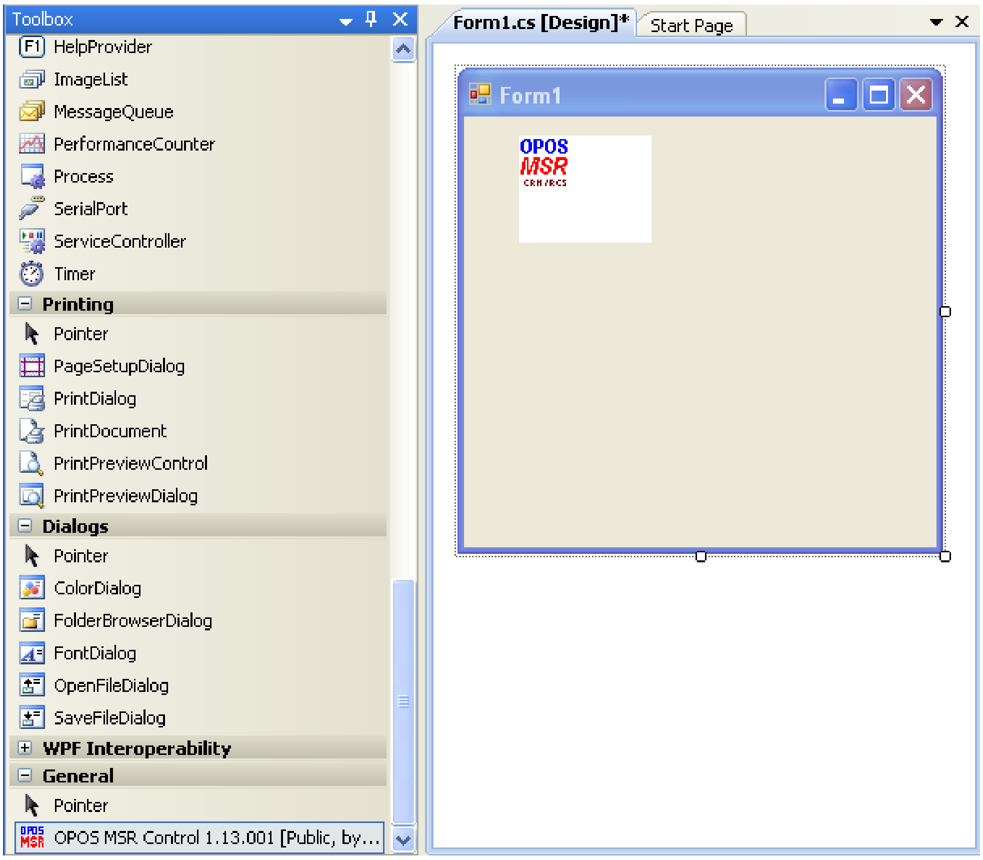
1. Create a "Windows Application" Project.
2. Right click on the "Toolbox" tool bar, select "Choose item …". Under "COM Components" tab, select "OPOS MSR Control 1.13.001" and click okay.



	Name	Path	Library
□	OPOS LineDisplay Control 1.13.00…	C:\Program Files\OPOS\CommonCO\…	OPOS LineDispl…
□	OPOS MICR Control 1.13.001 [Pu…	C:\Program Files\OPOS\CommonCO\…	OPOS MICR Co…
□	OPOS MotionSensor Control 1.13.…	C:\Program Files\OPOS\CommonCO\…	OPOS MotionSe…
☑	OPOS MSR Control 1.13.001 [Pub…	C:\WINDOWS\system32\OPOSMSR.…	OPOS MSR Con…
□	OPOS PINPad Control 1.10.000 […	C:\WINDOWS\system32\OPOSPINPa…	OPOS PINPad C…
□	OPOS PointCardRW Control 1.13.…	C:\Program Files\OPOS\CommonCO\…	OPOS PointCar…
□	OPOS POSKeyboard Control 1.13…	C:\Program Files\OPOS\CommonCO\…	OPOS POSKeyb…
□	OPOS POSPower Control 1.13.00…	C:\Program Files\OPOS\CommonCO\…	OPOS POSPowe…
□	OPOS POSPrinter Control 1.13.00…	C:\Program Files\OPOS\CommonCO\…	OPOS POSPrint…
□	OPOS RemoteOrderDisplay Contr…	C:\Program Files\OPOS\CommonCO\…	OPOS RemoteO…

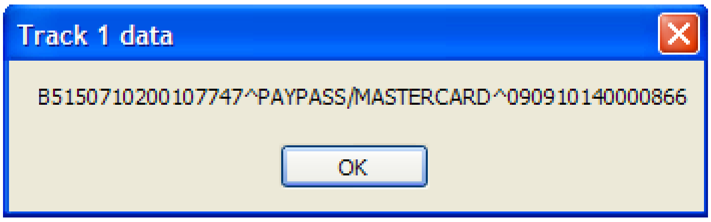
OPOS MSR Control 1.13.001 [Public, by CRM/RCS-Dayton]
Language: Language Neutral
Version: 1.0

3. Add "OPOS MSR Control" to "Form1". Double click on the OPOS MSR Control to add DataEvent handler code. Notice that the device name might need to be changed for different interface.



```
        private void Form1_Load(object sender, EventArgs e)
        {
            if (axOPOSMSR1.Open("IDTECH_SecuRED_USBKB") == 0)
              //0 is OPOS_SUCCESS
            {
                axOPOSMSR1.ClaimDevice(100);
                axOPOSMSR1.DeviceEnabled = true;
                axOPOSMSR1.DataEventEnabled = true;
            }
        }
        private void axOPOSMSR1_DataEvent(object sender,
AxOposMSR_1_13_Lib._IOPOSMSREvents_DataEventEvent e)
        {
            MessageBox.Show(axOPOSMSR1.Track1Data, "Track 1 data");
            axOPOSMSR1.DataEventEnabled = true;
        }
```

4. Run the program and swipe a card. Track 1 data will be displayed in a window.



```
B5150710200107747^PAYPASS/MASTERCARD^090910140000866
```

4. Result Code/Error Code List

```
const LONG OPOS_SUCCESS          =   0;
const LONG OPOS_E_CLOSED         = 101;
const LONG OPOS_E_CLAIMED        = 102;
const LONG OPOS_E_NOTCLAIMED     = 103;
const LONG OPOS_E_NOSERVICE      = 104;
const LONG OPOS_E_DISABLED       = 105;
const LONG OPOS_E_ILLEGAL        = 106;
const LONG OPOS_E_NOHARDWARE     = 107;
const LONG OPOS_E_OFFLINE        = 108;
const LONG OPOS_E_NOEXIST        = 109;
const LONG OPOS_E_EXISTS         = 110;
const LONG OPOS_E_FAILURE        = 111;
const LONG OPOS_E_TIMEOUT        = 112;
const LONG OPOS_E_BUSY           = 113;
const LONG OPOS_E_EXTENDED       = 114;
```