OMTD
OPOS FOR MAGTEK DEVICES
PROGRAMMING REFERENCE MANUAL

Manual Part Number: 99875144 Rev 3

JUNE 2003

MAGTEK®

REGISTERED TO ISO 9001:2000

1710 Apollo Court
Seal Beach, CA 90740
Phone: (562) 546-6400
FAX: (562) 546-6301
Technical Support: (651) 415-6800
www.magtek.com

REVISIONS

Rev Number	Date	Notes
1	5 Oct 99	Initial Release
2	17 Feb 00	Section 2: Added "Registering an OLE Programmatic ID Alias for OPOS Control Object." Added Figure 2-1; Section 3: Added "Registering an OLE Programmatic ID Alias for OPOS Control Object."
3	13 Jun 03	Engineering Update for MTD Software Release 1.12. Front Matter: Added ISO line to logo, changed Tech Support phone number, and replaced Limited Warranty with generic software license

MAGTEK®

REGISTERED TO ISO 9001:2000

1710 Apollo Court, Seal Beach, CA 90740
Voice: 562-546-6400 Fax: 562-546-6301

MagTek Part Number 99875144-2
13 June 2003

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 ABOVE ADDRESS 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 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 to be free from defects in material or workmanship under normal use. THE SOFTWARE IS PROVIDED AS IS WITHOUT 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.

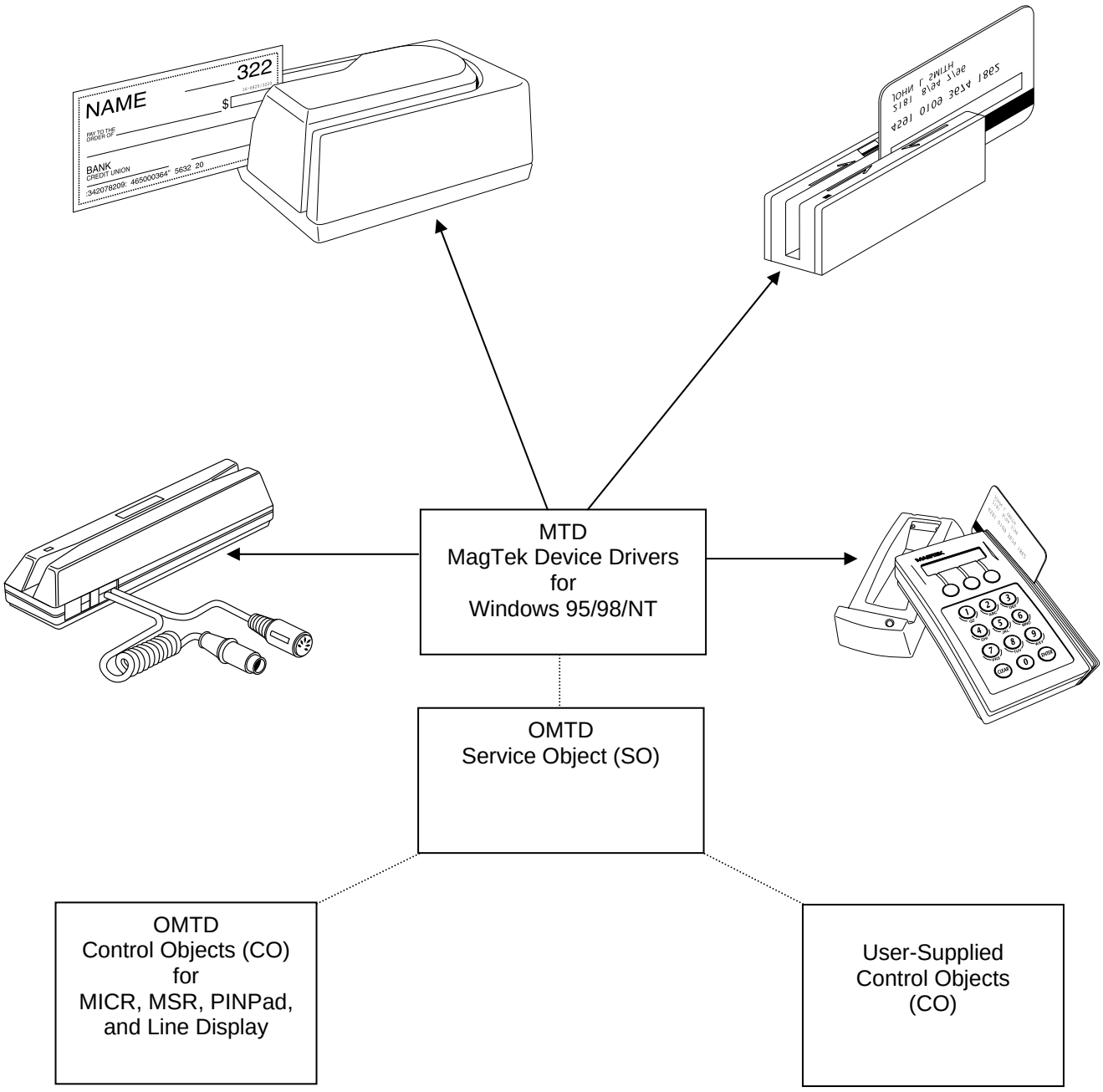
GOVERNING LAW: If any provision of this Agreement is found to be unlawful, void or unenforceable, that provision shall be removed from consideration under this Agreement and will not affect the enforceability of any of the remaining provisions. This Agreement shall be governed by the laws of the State of California and shall insure 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 ABOVE ADDRESS OR E-MAILED TO support@magtek.com.

TABLE OF CONTENTS

FIGURE



OMTD Program Controlled Devices

SECTION 1. FEATURES AND REQUIREMENTS

The OMTD program is an extension of the MagTek Device Driver Program (MTD). The OMTD Program provides an OPOS-compliant application interface to MagTek products and works on Windows 95, 98, ME, NT, 2000, and XP. OMTD is in the form of a set of ActiveX controls (COs) and a Service Object (SO).

OMTD is the OPOS (OLE for Retail POS) Support for MagTek Devices. It contains the Control Objects for the MICR, MSR, PINPad, and Line Display device classes, and a Service Object.

The OMTD extensions are installed as part of the MTD installation package. When installing the MTD drivers, you have the option to include OPOS support. Refer to the MTD Programming Manual (99875125) for complete installation information.

Part Number	Medium
30037385	MTD/OMTD Installation Package for Windows 95/98/ME/NT/2000/XP (CD)
99510030	MTD/OMTD Installation Package for Windows 95/98/ME/NT/2000/XP (WEB)

FEATURES

- OMTD is a set of COM objects that act as an extension to the MTD driver and provide an OPOS-compliant application interface to MagTek products.

- Each control consists of a control object (CO) and a service object (SO). The CO is an ActiveX control that exposes to applications a set of properties, methods and events defined by OPOS for a specific device class (e.g., MICR devices, PINPads, line displays, Magnetic Stripe Readers (MSR), etc.).

- The SO is a local in-process OLE Automation server (in a DLL) that is used by a CO to implement the OPOS-prescribed functionality for a specific device. The SO performs this functionality by sending one or more commands to the MTD driver using the Win32 File IO.

- In addition to the OPOS-specific functionality, each object exposes the OLE-specific interfaces that make the control Microsoft compliant.

REQUIREMENTS

Each control provides the following functionality, to the degree supported by the particular device class.

- Each CO and SO provides self-registration functionality.

- Each CO implements the following standard OLE control interfaces thereby making the control Microsoft OLE compliant: *IDispatch*, *IViewObject2*, *IOleInPlaceObject*, *IOleObject*, *IPersistStorage*, *IDataObject*, and *IProvideClassInfo*.

- Exposes OPOS-specific device class methods and properties through the *IDispatch* interface.

- Each CO implements all OPOS-specific functionality for a specific device class as defined by the OPOS Specification.

- A CO will only support MagTek SO(s).

- The SO implements the following standard OLE interfaces: *IUnknown*, *IClassFactory*, and *IDispatch*.

- The SO implements a union of all supported OPOS device class interfaces.

- The SO implements the OPOS functionality by sending commands to the MTD driver using the Win32 File IO API.

Interfacing to Applications

The OPOS Specification defines a common architecture for interfacing Windows applications with a variety of POS devices. The CO is loaded and created by the host application or container class whenever it needs to interact with the supported device. If an application needs to access a combo device (one that supports multiple device classes), the application is required to create and open several different control objects, one for each base class functionality that the device supports.

Interfacing to the Device

The CO loads and creates a SO in response to an application requesting a device to be opened. The CO invokes methods on the SO to request implementation of OPOS-specific functionality. In response, the SO sends one or more commands to the MTD driver using the Win32 File IO API to access the device functionality.

Supported Device Classes

The following OPOS device classes are supported by the SO:

- MSR (Magnetic Stripe Reader, e.g., Port Powered Swipe Reader, MiniWedge Swipe Reader)

- MICR (Magnetic Ink Character Recognition Check Reader, e.g., MICR Plus, Mini MICR)
- PINPad (e.g., IntelliPIN)
- Line Display (e.g., IntelliPIN)

Operation

The requirements for operation are as follows:

- Computer system: Intel Pentium-based PC or better
- Operating system: Microsoft Windows 95/98/ME/NT/2000/XP
- Environment: Win32
- Packaging: ActiveX control (.OCX) for OMTD CO, COM in-process server (.DLL) for OMTD SO
- Device access: OMTD accesses the device through existing MTD drivers.

COMPATIBILITY

Each of the OMTD controls is designed to be compatible with release 1.04 of the OPOS Specification and work with any containing application or container that implements the standard OLE mechanisms of creation and binding.

The common service object is designed to be compatible with the 1.05 version of the MTD driver or later. The SO will function with earlier versions of MTD, but not all functionality may be supported (e.g., data parsing).

It must be noted that the MTD driver itself does not have any knowledge about OPOS. This gives the ability to modify the OMTD controls without having to modify the MTD drivers and visa versa.

MTD will auto-configure OPOS parameters when it is installed and when any changes are made to the device (e.g., add, remove, friendly name change, etc.).

CONFIGURATION

The OPOS specification does not provide support for or prescribe mechanisms for configuration of OPOS controls. To provide a service to the user, the configuration of the OMTD controls will be integrated with the MTD class installer.

The OPOS configuration for MagTek devices is provided in a separate DLL that is invoked by both the OMTD setup application and the MTD class installer when they want to modify the OPOS configuration for a device.

REFERENCE DOCUMENTS

1. *MagTek Device Drivers for Windows, Technical Reference Manual* (P/N 99875125)

2. *OLE for Retail POS, Application Programmer's Guide*, Release 1.4, September 23, 1998, International Standard, other OLE/ActiveX compliant 32-bit operating system

3. *OLE for Retail POS, Control Programmer's Guide*, Release 1.4, International Standard, September 24, 1998, Windows 95/98, Windows NT, or other OLE/ActiveX compliant 32-bit operating system

4. Header files for VC++ and VB

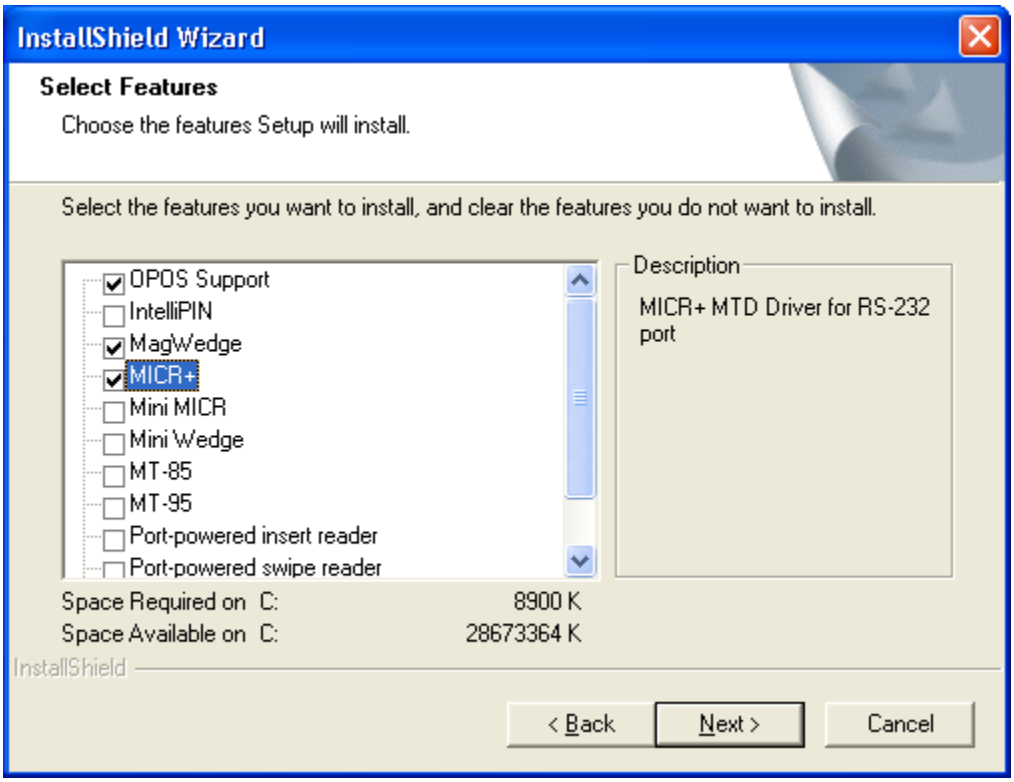
5. *OLE Controls Inside Out*, Adam Denning, Microsoft Press, 1995

SECTION 2. INSTALLATION

MagTek Device Drivers (MTD) must be installed before installing the OMTD. Refer to *MagTek Device Drivers for Windows, Technical Reference Manual*, P/N 99875125.

OMTD INSTALLATION NOTES

The OMTD extensions to the MTD Drivers are installed as part of the MTD installation package. When installing the MTD drivers, check the ***OPOS Support*** option at the top of the list.



Selecting the ***OPOS Support*** option triggers the installation of all of the OMTD controls. It also copies all CO and SO packages, and adds registry entries under the OPOS service provider key. Additionally, all of the installed MTD devices will be configured for OPOS operation. Each MTD driver will be mapped to the service object for all device classes supported by the particular device.

Note

When installing the **OPOS Support** *on Windows 2000 and XP, you will need to manually configure the devices by using the OMTDCFG configuration utility. See* Modifying and Removing Devices *below for more information.*

REGISTERING OLE PROGRAMMATIC ID ALIAS FOR OPOS CONTROL OBJECT

The OPOS specification states that the OLE Programmatic IDs for OPOS control objects should be OPOS.xxx where xxx specifies the device class of the control object (e.g., MSR, MICR, PINPad, etc). The result of this requirement is that only one control object of a specific device class can be registered at a single time.

By default, the OMTD installation registers aliases for the control objects' OLE programmatic ID, as required by the OPOS specification. The registered aliases are:

- MSR Control Object = OPOS.MSR
- MICR Control Object = OPOS.MICR
- PINPad Control Object = OPOS.PINPad
- Line Display Control Object = OPOS.LineDisplay

If it is desired to use more than one control object of a specific device class at a time, the alias registration can be bypassed by typing a:\setup c in step #4 below.

When the alias registration is bypassed, the control objects are registered under the following OLE programmatic IDs:

- MSR Control Object = MAGTEK.OPOS.MSR
- MICR Control Object = MAGTEK.OPOS.MICR
- PINPad Control Object = MAGTEK.OPOS.PPAD
- Line Display Control Object = MAGTEK.OPOS.LDSP

MODIFYING AND REMOVING DEVICES

After the OMTD Controls are installed, the configuration utility (OMTDCFG.EXE) can be used to update/remove OPOS configuration for installed MagTek devices. It is copied to the C:\ProgramFiles\MagTek\MTDInstall Folder when the OMTD controls are installed. OMTDCFG.EXE is a Win32 console application that has the following usage:

omtdcfg [/|-][option] [class flag]

The [**option**] parameter can be one of the following:

- c , C = Configure OPOS support for installed MagTek devices
- d , D = Remove OPOS support for configured MagTek devices
- l , L = List currently configured MagTek devices
- ? = Display usage help

The [**class flag**] parameter gives the option to select specific device classes to configure, remove, or list. It is a bit flag that can be one or more of the following:

- <empty> = Perform operation for all device classes listed below
- 0x01 = Perform operation for MSR device class
- 0x02 = Perform operation for MICR device class
- 0x04 = Perform operation for PIN Pad device class
- 0x08 = Perform operation for Line Display device class

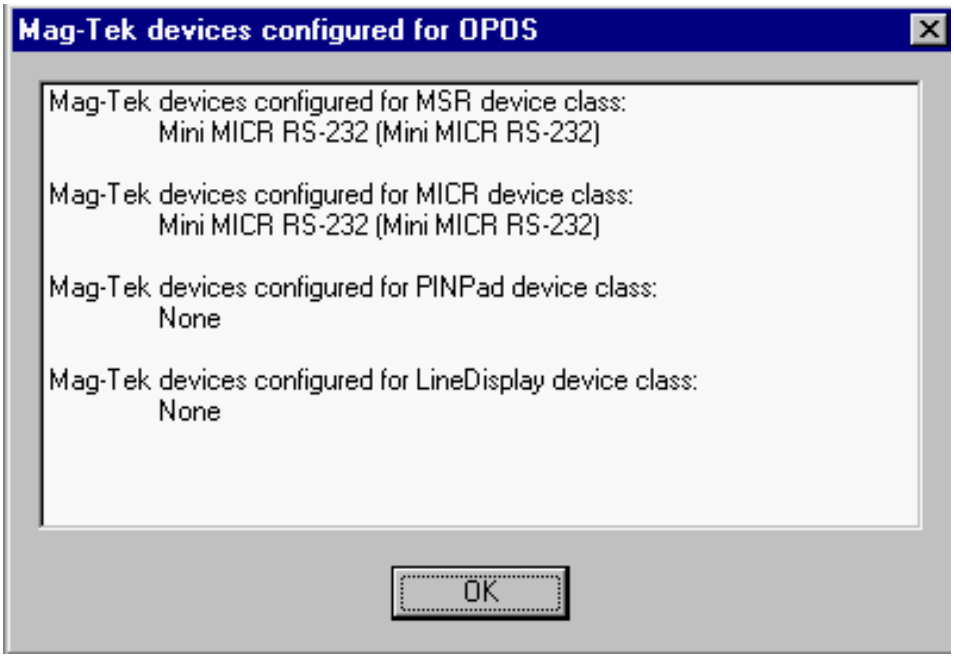
The configuration utility should be run after MagTek devices are installed, removed, or modified in Windows NT/2000/XP.

The configuration utility should be run from the DOS command line so that its output may be seen.

Examples:

omtdcfg -C 0x02 [configure MICR devices]
omtdcfg -L 0x03 [list MICR & MSR devices]

After this, a window listing all of the configured devices will be shown, as indicated below.



SECTION 3. IMPLEMENTATION NOTES

The following sections define specifics of the OMTD implementation of the OPOS controls.

SINGLE SERVICE OBJECT

The OMTD product implements a slight deviation from the OPOS Specification. The OPOS Specification specifies that there is a separate control object for each device class and separate service object for each device model. OMTD conforms to the idea of a separate control object per device class, but it diverges from the specification in that OMTD contains a single service object that is used for all device models.

The reasoning behind the separate service object idea is that each device has its own interface (i.e., set of commands) and communication channel (e.g., RS-232 or keyboard) which requires different implementations of the OPOS functionality.

The OMTD product contains only one service object, the common service object, because the MTD driver handles the data medium and the transport protocol to the device, providing uniform access to the device regardless of the type of connection and device model.

POTENTIAL PROBLEMS IN WINDOWS 95/98

The MTD property sheet (settings) in Windows 95/98 has an additional check box that is used to enable or disable OMTD for that device. When a new device is installed, the box is checked by default if OMTD is installed.

When the box is checked, the class installer creates and maintains a registry entry that maps the device name to the SO for each device class supported by the device. The registry entry is made up of the friendly name for the device (e.g., "micr+") and the device class (e.g., msr) separated by a period. An example is: "Micr+.msr".

When the state of the box is changed from checked to unchecked or the device is removed from the system, the class installer removes the device's registry entries for each supported device class.

MODIFYING OMTD DEVICE CONFIGURATION

The OMTD Service Object accepts per-device parameterization from the registry under the following path:

HKLM\Software\OLEForRetail\ServiceInfo\MagTek\Devices\<device friendly name>

The following value is currently supported for each device:

Name	Type	Description
msr.enable_sentinels	REG_DWORD	When set to 1, the OMTD MSR Control includes start and end sentinels with the TrackXData properties. The default value is 0 (disabled).

When OMTD is installed or reconfigured, the device configuration, as described above, is reset to the default values for each device.

OPENING THE DEVICE

The device name that should be provided on the Open method for the control is made up of the friendly name for the device and the device class for which the open method is being called. It has the following format:

<device friendly name>.<device class>

The table below shows the legal combinations of <device friendly names> and <device class>.

	Device Classes			
Default Friendly Name	Msr	Micr	Pinp	Ldsp
IntelliPIN RS-232	X		X	X
IntelliPIN Wedge	X		X	X
IntelliPIN MICR+ Aux	X		X	X
Mag-Wedge	X			
MiniWedge	X			
MICR+	X	X		
Mini MICR RS-232	X	X		
Mini MICR Wedge	X	X		
MT-85	X			
MT-95	X			
Port-Powered Swipe reader	X			
Port-Powered Insert reader	X			

For example, the OPOS device names for MICR+ with an MSR is "micr+.micr" and "micr+.msr".

REGISTERING OLE PROGRAMMATIC ID ALIAS FOR OPOS CONTROL OBJECT

The OPOS specification states that the OLE Programmatic IDs for OPOS control objects should be OPOS.xxx where xxx specifies the device class of the control object (e.g., MSR, MICR, PINPad, etc). The result of this requirement is that only one control object of a specific device class can be registered at a single time.

By default, the OMTD installation registers aliases for the control objects' OLE programmatic ID, as required by the OPOS specification. The registered aliases are:

- MSR Control Object = OPOS.MSR
- MICR Control Object = OPOS.MICR
- PINPad Control Object = OPOS.PINPad
- Line Display Control Object = OPOS.LineDisplay

When the alias registration is bypassed, the control objects are registered under the following OLE programmatic IDs:

- MSR Control Object = MAGTEK.OPOS.MSR
- MICR Control Object = MAGTEK.OPOS.MICR
- PINPad Control Object = MAGTEK.OPOS.PPAD
- Line Display Control Object = MAGTEK.OPOS.LDSP

When the OMTD controls are uninstalled, the registered aliases are removed only if they still refer to an OMTD control object meaning that another OPOS control has not been installed after the OMTD controls were installed.

DEVICE POWER REPORTING

The OMTD controls do not support the optional power reporting capability described in the OPOS Specification. This is due to the fact that MagTek devices and MTD do not report power events.

SAVING DEVICE SETTINGS ACROSS CLAIM/RELEASE

The OPOS Specification defines that settable device characteristics are saved and restored for each application that accesses the device. The complexity of supporting this functionality does not justify the marginal benefit provided to applications. If MagTek foresees frequent simultaneous access of a device from multiple applications, this feature may be added later.

ERROR PROCESSING

Reporting

The SO will display a message box if it receives a property request containing an invalid *PropIndex*. These types of errors should be found during development, testing, or staging of the containing application prior to rollout to a customer. This type of error reporting is recommended by the OPOS Specification.

Incompatibility Issues

If the control object determines that the specified service object is not supplied by MagTek or does not implement the required device class methods, it will fail the **Open** method. The CO determines if the SO is supplied by MagTek by querying a private property on the SO. However, the SO should work with other COs by specification.

The minimally required device class methods are those methods that are common to all device classes. Based on the service object version, one or more device class-specific methods may also be required.

The PINPad CO will fail the **Open** method if the OPOS version of the SO is less than 1.3. This is because the PINPad device class did not exist with prior versions of OPOS.

The SO will fail its **OpenService** method if it fails to bind to the **SOxxx** event handlers of the CO.

SECTION 4. CONTROL OBJECTS

This section defines the interfaces, properties, and methods exposed or used by the OMTD control objects. In the tables that follow, in the "Access" column, "R" indicates that the property is Read Only, "R/W" indicates that the property can be modified.

COMMON INTERFACES

The following sections describe the interfaces, methods, and properties that are common to all OMTD control objects.

OLE Interfaces

Each OMTD control object implements the following standard OLE interfaces: *IUnknown*, *IClassFactory*, *IDispatch*, *IViewObjectEx*, *IViewObject2*, *IViewObject*, *IOleInPlaceObjectWindowless*, *IOleInPlaceObject*, *IOleWindow*, *IOleInPlaceActiveObject*, *IOleControl*, *IOleObject*, *IPersistStreamInit*, *IPersist*, *IConnectionPointContainer*, *ISpecifyPropertyPages*, *IQuickActivate*, *IPersistStorage*, *IDataObject*, *IProvideClassInfo*, and *IProvideClassInfo2*. See the OLE documentation for detailed descriptions of these interfaces. (See Section 1, Reference Documents, numbers 2, 3, and 4.)

Event Methods

The following event methods are implemented by the containing application via its *IDispatch* interface. These event methods are also implemented by the CO to receive event notifications from the SO except they are preceded with *SO* and do not have *Event* in their name (e.g., **DataEvent** = **SOData**)

void DataEvent (LONG Status)

Fired to present input data from the device to the application. The actual input data is placed in one or more device-specific properties.

The *Status* parameter contains the input status. Its value is device class-dependent, and may describe the type or qualities of the input.

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

Fired by a Service Object to communicate directly with the application. This event provides a means for a Service Object to provide events to the application that are not otherwise supported by the Control Object.

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

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

void OutputCompleteEvent (LONG OutputID)

Fired when a previously started asynchronous output request completes successfully. The *OutputID* parameter indicates the ID number of the asynchronous output request that is complete.

void StatusUpdateEvent (LONG Status)

Fired when a Control needs to alert the application of a device status change. The *Status* parameter is for device class-specific data, describing the type of status change. When a device is enabled, the Control may fire initial **StatusUpdateEvents** to inform the application of the device state.

Common Properties

The following properties are common to all OMTD control objects. Most properties are initialized to their default values after a successful call to the **Open** method. If a property has not been initialized, its default value will be "[Error]" for BSTR properties, 0 for numeric properties and FALSE for Boolean properties. See the OPOS Specification for more information.
It is the application's responsibility to determine what properties the CO supports by querying the corresponding capability (e.g. CapXXX) property. If an unsupported property is accessed that does not have a corresponding capability property, the return status will be OPOS_E_ILLEGAL.

Property Name	Access	Type	Description
AutoDisable	R/W	BOOL	Set to TRUE if application wants to receive and process only one input at a time. The default is FALSE.
BinaryConversion	R/W	LONG	Format in which characters are placed into BString arguments. The default is OPOS_BC_NONE.
CapPowerReporting	R	LONG	Identifies the reporting capabilities of the device.
CheckHealthText	R	BSTR	Holds the results of the most recent call to the CheckHealth method.
Claimed	R	BOOL	If TRUE, the device is claimed for exclusive access. The device must be claimed before accessing the device methods and properties and before events are fired to the application. The default is FALSE.
ControlObjectDescription	R	BSTR	String identifying the CO and the company that produced it (e.g., "MSR OPOS Control, © 1999 MagTek, Inc.").
ControlObjectVersion	R	LONG	Control object version number.

Property Name	Access	Type	Description
DataCount	R	LONG	Number of enqueued data events at the control. The default is 0.
DataEventEnabled	R/W	BOOL	When TRUE, a **DataEvent** is delivered as soon as input data is enqueued. The default is FALSE.
DeviceDescription	R	BSTR	String identifying the device (e.g., MICR Plus).
DeviceEnabled	R/W	BOOL	When TRUE, the device is placed in an operational state. When FALSE, any input is discarded and output operations are disallowed. The default is FALSE.
DeviceName	R	BSTR	Identifies the device (e.g., friendly name of the device).
FreezeEvents	R/W	BOOL	When TRUE, the control will not deliver events. The default is FALSE.
OutputID	R	LONG	Holds the identifier of the most recently started asynchronous output.
PowerNotify	R/W	LONG	Contains the type power notification selection made by the application. The default is OPOS_PN_DISABLED.
PowerState	R	LONG	Contains the current power condition, if it can be determined. The default is OPOS_PS_UNKNOWN.
ResultCode	R	LONG	Result code of the most recently invoked method.
ResultCodeExtended	R	LONG	Contains extended result information when **ResultCode** is set to OPOS_S_EXTENDED.
ServiceObjectDescription	R	BSTR	String identifying the service object supporting the device and the company that produced it (e.g., "OPOS Common Service Object, © 1999 MagTek, Inc.").
ServiceObjectVersion	R	LONG	Service object version number.
State	R	LONG	Contains the current state of the control.

Common Methods

The following methods are common to all OMTD control objects.

LONG Open (BSTR DeviceName)

Call to open a device for subsequent I/O.

When the **Open** method is successful, it sets the properties **Claimed**, **DeviceEnabled**, and **FreezeEvents**, as well as descriptions and version numbers of the OPOS software layers (i.e. CO and SO). Additional class-specific properties may also be initialized.

LONG Close 0

Called to release the device and its resources. If the **DeviceEnabled** property is TRUE, then the device is first disabled. If the **Claimed** property is TRUE, then exclusive access to the device is first released.

LONG Claim (LONG Timeout)

Called to acquire exclusive access to the device. The *Timeout* parameter gives the maximum number of milliseconds to wait for exclusive access to be satisfied. If zero, the method attempts to claim the device, then returns the appropriate status immediately. If OPOS_FOREVER (-1), the method waits as long as needed until exclusive access is satisfied.

LONG Release ()

Call this method to release exclusive access to the device. If the **DeviceEnabled** property is TRUE and the device is an exclusive-use device, then the device is first disabled.

LONG CheckHealth (LONG Level)

Called to test the state of a device. A text description of the results of this method is placed in the **CheckHealthText** property. The **CheckHealth** method is always synchronous.

LONG ClearInput ()

Called to clear all device input that has been buffered. Any data events or input error events that were enqueued – usually waiting for **DataEventEnabled** to be set to TRUE and **FreezeEvents** to be set to FALSE – are also cleared.

LONG ClearOutput ();

Called to clear all device output that has been buffered. Also, when possible, halts outputs that are in progress. Any output error events that were enqueued – usually waiting for **FreezeEvents** to be set to FALSE – are also cleared.

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

Call to communicate directly with the Service Object.

This method provides a means for a Service Object to provide functionality to the application that is not otherwise supported by the standard Control Object for its device class. Depending upon the Service Object's definition of the command, this method may be asynchronous or synchronous.
Use of **DirectIO** will make an application non-portable (e.g., it will not work with service objects developed by other companies). The application may, however, maintain portability by performing **DirectIO** calls within conditional code. This code may be based upon the value of the **ServiceObjectDescription**, **DeviceDescription**, or **DeviceName** property.

MAGNETIC STRIPE READER (MSR) DEVICE CLASS

The following sections describe those properties and methods that are specific to the MSR device class.

MSR Properties

Property Name	Access	Type	Description
AccountNumber	R	BSTR	Account number obtained from the most recently swiped card.
CapISO	R	BOOL	If TRUE, the device supports ISO cards.
CapJISOne	R	BOOL	If TRUE, the device supports JIS Type-I cards.
CapJISTwo	R	BOOL	If TRUE, the device supports JIS Type-II cards.
DecodeData	R/W	BOOL	If TRUE, each byte of track data contained within the **Track1Data**, **Track2Data**, and **Track3Data** properties is mapped from its raw format to its corresponding decoded ASCII bit sequence. The default is TRUE.
ErrorReportingType	R/W	LONG	Specifies when an error is reported by an **ErrorEvent** when a card is swiped and one or more of the tracks specified by the **TracksToRead** property contains data with errors. The default is MSR_ERT_CARD – report errors at a card level.
ExpirationDate	R	BSTR	Expiration date as retrieved from the most recently swiped card, as four ASCII decimal characters in the form YYMM.
FirstName	R	BSTR	First name obtained from the most recently swiped card.
MiddleInitial	R	BSTR	Middle initial obtained from the most recently swiped card.
ParseDecodedData	R/W	BOOL	If TRUE, the decoded data contained within the **Track1Data** and **Track2Data** properties is further separated into fields for access via various other properties. The default is TRUE.
ServiceCode	R	BSTR	Service code obtained from the most recently swiped card.
Suffix	R	BSTR	Suffix obtained from the most recently swiped card.
Surname	R	BSTR	Surname obtained from the most recently swiped card.
Title	R	BSTR	Title obtained from the most recently swiped card.
Track1Data	R	BSTR	Contains either the track 1 data from the previous card swipe or an empty string. This property contains the track data between but not including the start and end sentinel characters (refer to *msr.enable_sentinels*).
Track1DiscretionaryData	R	BSTR	Track 1 discretionary data obtained from the most recently swiped card.
Track2Data	R	BSTR	Contains either the track 2 data from the previous card swipe or an empty string.

Property Name	Access	Type	Description
Track2DiscretionaryData	R	BSTR	Track 2 discretionary data obtained from the most recently swiped card.
Track3Data	R	BSTR	Contains either the track 3 data from the previous card swipe or an empty string.
TracksToRead	R/W	LONG	Indicates the track data that the application wishes to have placed into the **Track1Data**, **Track2Data** and **Track3Data** properties following a card swipe. The default is MSR_TR_1_2_3.

MSR Methods

The MSR device class does not define any additional methods.

MICR Device Class

The following sections describe those properties and methods that are specific to the MICR device class.

MICR Properties

Property Name	Access	Type	Description
AccountNumber	R	BSTR	A string containing the account number parsed from the most recently read MICR data.
Amount	R	BSTR	A string containing the amount field parsed from the most recently read MICR data.
BankNumber	R	BSTR	A string containing the bank number portion of the transit field parsed from the most recently read MICR data.
CapValidationDevice	R	BOOL	Indicates if this device also performs validation printing via the POS Printer Control's slip station.
CheckType	R	LONG	A number that represents the type of check parsed from the most recently read MICR data.
CountryCode	R	LONG	A number that represents the country of origin of the check parsed from the most recently read MICR data.
EPC	R	BSTR	A string containing the Extended Processing Code ("EPC") field parsed from the most recently read MICR data. The string will contain a single character 0 though 9 if the field is present. If not, the string will be empty ("").
RawData	R	BSTR	A string containing the MICR data from the most recent MICR read.
SerialNumber	R	BSTR	A string containing the serial number of the check parsed from the most recently read MICR data.
TransitNumber	R	BSTR	A string containing the transit field of the check parsed from the most recently read MICR data.

MICR Methods

The MagTek MICR devices do not require processing of the following methods, so they will always return OPOS_SUCCESS.

BeginInsertion(LONG Timeout)

Called to initiate check insertion processing.

EndInsertion()

Called to end check insertion processing.

BeginRemoval(LONG Timeout)

Called to initiate check removal processing.

EndRemoval()

Called to end check removal processing.

PINPAD DEVICE CLASS

The following sections describe those properties and methods that are specific to the PINPad device class.

PINPad Properties

Property Name	Access	Type	Description
AccountNumber	R/W	BSTR	The account number to be used for the current EFT transaction.
AdditionalSecurity Information	R	BSTR	This property may contain additional security/encryption information after a **DataEvent** event.
Amount	R/W	CURRENCY	The amount of the current EFT transaction.
AvailableLanguagesList	R	BSTR	This property is a comma-separated string of the languages supported by the pre-defined prompts in the PINPad.
AvailablePromptsList	R	BSTR	This property is a comma-separated string of supported values for the **Prompt** property.
CapDisplay	R	LONG	Defines the operations that the Application may perform on the PINPad display.
CapKeyboard	R	BOOL	Defines whether the application can obtain input from the PINPad keyboard.
CapLanguage	R	LONG	Defines the capabilities that the application has to select the language of pre-defined messages (e.g., English, French, and Arabic).
CapMACCalculation	R	BOOL	If TRUE, the PINPad supports MAC calculation.
CapTone	R	BOOL	If TRUE, the PINPad has a Tone Indicator.
EncryptedPIN	R	BSTR	This property will contain the value of the Encrypted PIN after a **DataEvent** event.
MaximumPINLength	R/W	LONG	The application should set this property to the maximum acceptable number of digits in a PIN.
MerchantID	R/W	BSTR	The Merchant ID, as it is known to the EFT Transaction Host.
MinimumPINLength	R/W	LONG	The application should set this property to the minimum acceptable number of digits in a PIN.
PINEntryEnabled	R	BOOL	The PINPad Control object sets this property to TRUE when an **EnablePINEntry** method is executed.
Prompt	R/W	LONG	This property identifies a pre-defined message to be displayed on the PINPad.
PromptLanguage	R/W	LONG	This property specifies the language of the message to be displayed (as specified by the **Prompt** property).
TerminalID	R/W	BSTR	The terminal ID, as it is known to the EFT Transaction Host.
Track1Data	R/W	BSTR	Contains either the track 1 data from the previous card swipe or an empty string.
Track2Data	R/W	BSTR	Contains either the track 2 data from the previous card swipe or an empty string.
Track3Data	R/W	BSTR	Contains either the track 3 data from the previous card swipe or an empty string.
TransactionType	R/W	BSTR	The type of the current EFT transaction.

PINPad Methods

BeginEFTTransaction(BSTR PINPadSystem, LONG TransactionHost)

This method must be called by the application to inform the PINPad Control of the beginning of an EFT Transaction. The PINPad Control will perform initialization functions (such as computing session keys). No other PINPad functions can be performed until this method is called. Set the *PINPadSystem* to "MDK" or "DUKPT".

EndEFTTransaction ()

This method must be called by the application to inform the PINPad Control of the end of an EFT Transaction.

EnablePINEntry ()

This method is called by the application to enable PIN Entry at the PINPad device.
When the user has completed the PIN entry operation (either by entering their PIN or by hitting Cancel), a **DataEvent** event will be fired to provide the encrypted PIN to the application.

ComputeMAC(BSTR InMsg, BSTR* pOutMsg)

This method is called by the application to have the PINPad compute a MAC value and append it to the designated message. Depending on the selected PINPad Management System, the PINPad may also insert other fields into the message. Note that the **ComputeMAC** method cannot be used while PINPad input (PIN Entry) is enabled.

VerifyMAC(BSTR Message)

This method is called by the application to have the PINPad verify the MAC value in a message received from an EFT Transaction Host.

UpdateKey(LONG KeyNum, BSTR Key)

This method is used to provide a new encryption key to the PINPad. It is used only for those PINPad Management Systems in which new key values are sent to the terminal as a field in standard messages from the EFT Transaction Host.

LINE DISPLAY DEVICE CLASS

The following sections describe those properties and methods that are specific to the Line Display LDSP device class.

LDSP Properties

Property Name	Access	Type	Description
CapBlink	R	LONG	Holds the character blink capability of the device.
CapBrightness	R	BOOL	If TRUE, the brightness control is supported; otherwise it is FALSE.
CapCharacterSet	R	LONG	Holds the default character set capability.
CapDescriptors	R	BOOL	If TRUE, then the display supports descriptors; otherwise it is FALSE.
CapHMarquee	R	BOOL	If TRUE, the display supports horizontal marquee windows; otherwise it is FALSE.
CapICharWait	R	BOOL	If TRUE, the display supports intercharacter wait; otherwise it is FALSE.
CapVMarquee	R	BOOL	If TRUE, the display supports vertical marquee windows; otherwise it is FALSE.
CharacterSet	R/W	LONG	Contains the character set for displaying characters.
CharacterSetList	R	BSTR	A string of character set numbers.
Columns	R	LONG	Holds the number of columns for this window.
CurrentWindow	R/W	LONG	Holds the current window to which text is displayed.
CursorColumn	R/W	LONG	Holds the column in the current window to which the next displayed character will be output.
CursorRow	R/W	LONG	Holds the row in the current window to which the next displayed character will be output.
CursorUpdate	R/W	BOOL	If TRUE when characters are displayed by the **DisplayText** or **DisplayTextAt** method, then **CursorRow** and **CursorColumn** will be updated to point to the character beyond the last character output. If FALSE when characters are displayed, then the cursor properties will not be updated.
DeviceBrightness	R/W	LONG	Holds the device brightness value, expressed as a percentage between 0 and 100.
DeviceColumns	R	LONG	Holds the number of columns on this device.
DeviceDescriptors	R	LONG	Holds the number of descriptors on this device. If the capability **CapDescriptors** is TRUE, then **DeviceDescriptors** is non-zero; otherwise it is zero.
DeviceRows	R	LONG	Holds the number of rows on this device.
DeviceWindows	R	LONG	Holds the maximum window number supported by this device. A value of zero indicates that only the device window is supported, and that no windows may be created.
InterCharacterWait	R/W	LONG	Holds the wait time between displaying each character with the **DisplayText** and **DisplayTextAt** methods. This timer gives a “teletype” appearance when displaying the text.
MarqueeFormat	R/W	LONG	Holds the marquee format for the current window.
MarqueeRepeatWait	R/W	LONG	Holds the wait time between scrolling the final character or row of the window into its viewport and restarting the marquee with the first or last character or row. The wait time is the specified number of milliseconds. (Note that the timer resolution may reduce the precision of the wait time.)
MarqueeType	R/W	LONG	Holds the marquee type for the current window. When not DISP_MT_NONE, the window is in Marquee Mode.

Property Name	Access	Type	Description
MarqueeUnitWait	R/W	LONG	Holds the wait time between marquee scrolling of each column or row in the window. The wait time is the specified number of milliseconds. (Note that the timer resolution may reduce the precision of the wait time.)
Rows	R	LONG	Holds the number of rows for this window.

LDSP Methods

DisplayText(BSTR Data, LONG Attribute)

The characters in *Data* are processed beginning at the location specified by **CursorRow** and **CursorColumn**, and continue in succeeding columns.

DisplayTextAt(LONG Row, LONG Column, BSTR Data, LONG Attribute)

The characters in *Data* are processed beginning at the window location specified by the *Row* and *Column* parameters, and continuing in succeeding columns. This method has the same effect as setting the **CursorRow** to *Row*, setting **CursorColumn** to *Column*, and calling the **DisplayText** method.

ClearText()

Clears the current window to blanks, sets **CursorRow** and **CursorColumn** to zero, and resynchronizes the beginning of the window with the start of the viewport.

ScrollText(LONG Direction, LONG Units)

Scrolls the current window.

SetDescriptor(LONG Descriptor, LONG Attribute)

Sets the state of one of the descriptors, which are small indicators with a fixed label.

ClearDescriptors()

Turns off all descriptors.

CreateWindow (LONG ViewportRow, LONG ViewportColumn, LONG ViewportHeight, LONG ViewportWidth, LONG WindowHeight, LONG WindowWidth)

Creates a viewport over the portion of the display given by the first four parameters. The window size is given by the last two parameters. Valid window row values range from (0) to (*WindowHeight*-1) and column values range from (0) to (*WindowWidth*-1).

DestroyWindow()

Destroys the current window. The characters displayed in its viewport are not changed. **CurrentWindow** is set to window 0. The device window and the associated window properties are updated.

RefreshWindow(LONG Window)

Changes the current window to *Window*, then redisplays its viewport. Neither the mapping of the window to its viewport nor the window's cursor position is changed.

USE CASES

Using an OMTD Control

Once an OMTD control has been installed and an application has created and linked to it, the first action an application must take on the CO is to call its **Open** method passing it a device name that is associated with the control object. The CO locates the corresponding SO and using the system registry and calls **CoCreateInstance** passing the SO's Programmatic GUID to create the object. The CO then binds to the SO's interfaces and invokes its **OpenService** method.

The CO initializes its properties as well as descriptions and version numbers of the OPOS control layers. Additional class-specific properties may also be initialized.

Since several applications may have the same device open at the same time, an application must gain exclusive access to the device by calling the **Claim** method. This method must be called before some methods and properties may be accessed.

Before the application can use the device, it must set the **DeviceEnabled** property to TRUE. This value brings the device to an operational state, while FALSE disabled the device. When a device is disabled, any physical input from the device will be discarded until the device is enabled.

An application uses the device by invoking various methods on the CO via the *IDispatch* interface.

When an application has finished using the device, the **Close** method should be called to release the device and associated resources. If the **DeviceEnabled** property is TRUE, then **Close** disables the device. If the **Claimed** property is TRUE, the **Close** releases the lock.

Before exiting, an application should close all open OMTD Controls.

SUPPORTED DEVICE FEATURES

The following sections describe the OPOS features that are not supported or partially supported for each device class. The restrictions are based on the functionality of the supported MagTek devices. In some cases, features not supported by the device (e.g., multiple windows in the Line Display) can be implemented by OMTD in software at a later time.

Common Features

Common Properties

Property Name	Comments
CapPowerReporting	This property returns OPOS_PR_NONE for all devices because no MagTek devices have power reporting capabilities.
Claimed	All devices must be claimed before access will be granted by the control.
PowerNotify	This property is initialized to OPOS_PN_DISABLED and the control will fail any attempts to enable this property because it is not supported by any devices.
PowerState	This property will always return OPOS_PS_UNKNOWN because **CapPowerReporting** is OPOS_PR_NONE.

Common Methods

Method Name	Comments
CheckHealth	Only OPOS_CH_INTERNAL is supported by the SO. All other levels are mapped to OPOS_CH_INTERNAL.
DirectIO	At this time, no commands have been defined for this method. See the *Discretionary Features* section later in this document for more details.

Common Events

Event Name	Comments
DirectIOEvent	Used only if asynchronous DirectIO method is supported.
OutputCompleteEvent	This event is not used because all output is performed synchronously for the device classes supported by OMTD.
StatusUpdateEvent	This event is not used because the power state reporting capability is not supported and no status update events are defined for the device classes supported by OMTD.

MSR Device Class

MSR Properties

Property Name	Comments
TracksToRead	This property modifies the “trk_enable” property of the MTD driver.
DecodeData	This property cannot be set to FALSE because the devices send all track data in decoded format.

MSR Methods

There are no device class-specific methods defined for the MSR class.

MICR Device Class

MICR Properties

Property Name	Comments
CapValidationDevice	This property will always be FALSE because the MICR devices do not support validation printing.
RawData	This property will contain the check data as returned by the MTD driver (e.g., 6500 format).

The **CheckType** and **CountryCode** properties are set based upon the MICR check status that is returned by the device and provided in the response of the /read command. The following table describes the values that are stored in the **CheckType** and **CountryCode** properties based on the MICR status. The MICR status values are listed in order of priority that they are returned by the device.

MICR Status	Description	CheckType Value	CountryCode Value
00	No error, check OK	MICR_CT_PERSONAL	MICR_CC_USA
01	No MICR Data	MICR_CT_UNKNOWN	MICR_CC_UNKNOWN
03	Low MICR signal, good read	MICR_CT_PERSONAL	MICR_CC_USA
04	Check # Error, Bad character in check #, or no check #.	MICR_CT_PERSONAL	MICR_CC_USA
05	Transit Error	MICR_CT_PERSONAL	MICR_CC_USA
07	Account Error	MICR_CT_PERSONAL	MICR_CC_USA
08	Canadian Check	MICR_CT_PERSONAL	MICR_CC_CANADA
09	Mexican Check	MICR_CT_PERSONAL	MICR_CC_MEXICO
10	Business Check	MICR_CT_BUSINESS	MICR_CC_USA
11	Amount field present	MICR_CT_PERSONAL	MICR_CC_USA

MICR Methods

Method Name	Comments
BeginInsertion	This method either always returns success or causes a **/read check** command to be sent to the MTD driver.
EndInsertion	This method always returns success. Optionally, it may wait until **/read check** command completes or the specified timeout expires.
BeginRemoval	This method always returns success because the MICR devices do not support its functionality.
EndRemoval	This method always returns success.

PINPad Device Class

PINPad Properties

Property Name	Comments
AvailableLanguagesList	This property is not supported because **CapDisplay** is PPAD_DSP_UNRESTRICTED.
AvailablePromptsList	This property is not supported because **CapDisplay** is PPAD_DSP_UNRESTRICTED.
CapDisplay	This property is initialized to PPAD_DSP_UNRESTRICTED, which specifies that the Line Display CO is the interface to the PINPad display.
CapLanguage	This property is initialized to PPAD_LANG_NONE because **CapDisplay** is PPAD_DSP_UNRESTRICTED.
CapMACCalculation	This property is always FALSE. MAC calculation is not supported due to security concerns.
CapKeyboard	This property is always FALSE. At a later time, access to the keyboard functionality may be added through a POS Keyboard control.
CapTone	This property is always FALSE because the IntelliPIN does not have a Tone Indicator.
MerchantID	This property is supported, but has no effect on the operation of the device (i.e., no commands are sent to the device).
MinimumPINLength	This property is supported, but cannot be modified. The minimum PIN length is based on the current encode mode (M/S or DUKPT).
Prompt	This property is not supported because **CapDisplay** is PPAD_DSP_UNRESTRICTED.
PromptLanguage	This property is not supported because **CapDisplay** is PPAD_DSP_UNRESTRICTED.
TerminalID	This property is supported, but has no effect on the operation of the device.
TransactionType	Only PPAD_TRANS_DEBIT and PPAD_TRANS_CREDIT options are supported because these are the transaction types supported by IntelliPIN.

PINPad Methods

Method Name	Comments
BeginEFTTransaction	The only supported PINPad Management Systems are “M/S” and “DUKPT” – the ones supported by IntelliPIN.
ComputeMAC	This function is not supported due to security concerns.
VerifyMAC	This function is not supported due to security concerns.

Line Display Device Class

LDSP Properties

The following table lists those **properties that are supported** by the Line Display CO. All other properties are **not** supported.

Property Name	Comments
CapBlink	CAP_CB_NOBLINK
CapBrightness	FALSE
CapCharacterSet	DISP_CCS_ASCII
CapDescriptors	FALSE
CapHMarquee	FALSE
CapICharWait	FALSE
CapVMarquee	FALSE
DeviceWindows	0 (only device window supported)
DeviceRows	2
DeviceColumns	16
DeviceDescriptors	0
DeviceBrightness	100 only
CharacterSet	DISP_CS_ASCII only
CharacterSetList	998
CurrentWindow	0 only
Rows	2
Columns	16
CursorRow	0..1
CursorColumn	0..16
CursorUpdate	TRUE, FALSE
MarqueeType	DISP_MT_NONE only
MarqueeFormat	DISP_MF_WALK (not used)
MarqueeUnitWait	0 (not used)
MarqueeRepeatWait	0 (not used)
InterCharacterWait	0 (not used)

LDSP Methods

Based on the device features and the capabilities defined above, only the following methods are supported:

Method Name	Comments
ClearText	Supported
DisplayText	Supported
DisplayTextAt	Supported

APPENDIX A. GLOSSARY

CO	OPOS Control Object – ActiveX Control that exposes a set of properties, methods, and events to an application for its device class
COM	Component Object Module
Device	Physical MagTek device (e.g., Port Powered Swipe Reader, MICR+, IntelliPIN, etc.)
Device class	Category of POS devices that share a consistent set of properties, methods, and events (e.g., MSR, MICR, PINPad, etc.)
DLL	Dynamic-Link Library
LDSP	Line Display Device (e.g., LCD on IntelliPIN)
MICR	Magnetic Ink Character Recognition (Check Reader)
MSR	Magnetic Stripe Reader
MTD	MagTek Device Driver
OLE	Object Linking and Embedding
OPOS	OLE for Retail Point of Sale
PINPad	PINPad Device Class (PIN Entry Device)
SO	OPOS Service Object – COM in-process server that provides the device-specific OPOS functionality

INDEX