QVS TERMINAL CONCENTRATOR

RPAMTerminal Object Programmer's Guide
Version 1.0

Draft

Notes:

- For further information regarding how to program and use the IBM POS peripherals refer to the IBM 4690 Store System Programming Guide and the CBASIC Language Reference.

Change Sheet			
Rev.	Date	Section	Change
C	2/6/2004		General update of document to include _QVSRPAMEvents documentation.
B	4/4/2000		Second review changes
A	3/31/2000		Initial version with a date

Chapter 1. Introduction

The purpose of this document is to explain the high level RPAMTerminal object programming interface to a 4690 sales application running under the QVS Terminal Concentrator/NT execution environment.

1.1 Concepts

The QVS Terminal Concentrator/NT allows a user to run one or more native 4690 terminal sales applications on Windows/NT. In this environment, TC/NT intercepts all OS service requests from the 4690 executable and carries them out on behalf of the application. This concept extends to device I/O as well. When an application requests device I/O from the OS, TC/NT remotes that request out to a network terminal using QVS's Remote Peripheral Access Method (RPAM) protocol.

Likewise, when inbound device data is received from a network terminal, TC/NT takes care of translating the RPAM message to native 4690 format and then forwarding it to the proper 4690 application.

In this way it is possible to decouple the device I/O aspect of the application execution from the business logic aspect. This allows you to run (virtually) a fairly fat-client application in a thin-client device as the thin client is only burdened with managing the I/O devices.

The RPAM communication protocol is a fairly low level protocol requiring a fair amount of effort on behalf of the application program to implement correctly. The RPAMTerminal object concept was developed to allow application developers to communicate with the 4690 sales application at a higher level.

The RPAMTerminal object is implemented as a COM component allowing the user to employ mainstream applications development tools to develop applications that take advantage of the benefits of a RPAM terminal communicating with 4690 sales applications running under TC/NT.

The component communicates with TC/NT using Windows sockets API. This allows the control to execute in the remote device or on the same machine as the concentrator. When running in the same machine as the concentrator it is possible to run multiple instances of the component, each representing a different virtual RPAM terminal device. The socket API uses the internet domain (AF_INET) as a means of communicating between the client and TC server.

Another important aspect of the implementation of the component as it relates to the sales application running under the concentrator is that the sales application is persistent across instantiations of the RPAMTerminal object. This means is that the client application need only create and use a RPAMTerminal object only when it needs to communicate with the sales application. During the interim periods when no client is connected to a sales application the concentrator will continue to execute the application. The client can reattach to the running sales application by creating a new instance of a RPAMTerminal object and setting the UnitID property (and sometimes the TermNum property) to the value used when the sales application was started and the RPAM interface layer will do the rest.

The RPAMTerminal object will not maintain any state information across instance creation. It is the responsibility of the client application to maintain this type of information. One area where this is important is if the client wishes to be able to perform mid-transaction recovery after a power-fail of the concentrator. In order for this to occur, the client application will need to retain the terminal number associated with the physical unit.

1.2 Pseudo-Code Example

Here is some VB-like pseudo-code that shows how a client application might use the RPAMTerminal object. The first thing the client should do is initialize the terminal and connect to the controller.

```
Sub Main
```

```
Dim Term As New QVSRPAM.RPAMTerminal
    Dim QVSApp As New QVSRPAM.RPAMApplication

    'set RPAM up to allow us to set terminal number
    QVSApp.TermNumAssignMode = rpaAppTermNum

    'do not try to recover transactions after power fail
    QVSApp.TransactionRecoveryMode = rpaTransRecoveryOff

    'allow rpam to configure our device IDs
    QVSApp.AutoDeviceConfigure = TRUE

    'allow rpam to handle field level input buffering
    Term.IOProc.FieldLevelInputMode = rpaFlRproc

    'set up the terminal to get ready to come online
    Term.UnitId = MyMacID
    Term.TermNum = 1

    ' come online the controller
    Term.Connect

    'Start the application
    Term.StartApplication
End Sub
```

At this point the QVS Terminal Concentrator has started an instance of the 4690 sales application and is actively executing it. It is acceptable for the client application to destroy its instance of the RPAMTerminal object at this time. The concentrator will continue to execute the application and buffer the state of the virtual terminal.

At a later time if the client application wanted to send keyboard data to the application it could instantiate another instance of the RPAMTerminal object. It would then assign the UnitID and TermNum properties to the same values used to when the Connect method was invoked and then use the IOProc device to send keystrokes.

```
Sub SignOn
    'create a new RPAMTerminal object
    Dim Term As New QVSRPAM.RPAMTerminal

    'assign UnitID and TermNum.  This will allow the RPAM
    'interface layer to reconnect the client application
    'to the 4690 sales application.
    Term.UnitId = MyMacID
    Term.TermNum = 1

    'at this point the client can feed data to the sales 'application.
    Sign on using keystroke "1/1" followed by the 'signon key

    Term.IOProc.SendKeyboardData (FUNC_CODE_1)
    Term.IOProc.SendKeyboardData (FUNC_CODE_SLASH)
    Term.IOProc.SendKeyboardData (FUNC_CODE_1)
    Term.IOProc.SendKeyboardData (FUNC_CODE_SIGNON)

    ' the application is now signed on and ready to receive scan
    'data
End Sub
```

Now assume that the client has scanner data to send to the application. The client is going to send in the scanner data and wait for the application to do the PLU. The client is going to trigger off of the first display message after the scan data was sent to determine when the PLU has completed. This code snippet assumes that the client has implemented an event sink to receive events dispatched from the RPAMTerminal object and that the event sink is active at the time the scan data is sent.

```
Dim gTerm As New RPAMTerminal

Sub SendScanData
    gTerm.UnitID = MyMacID
    gTerm.TermNum = 1

    GTerm.IOProc.SendScannerData (LBL_UPC_E, "1234567")
End Sub

Sub gTerm_OnDisplayEvent (Row1Text As String, Row2TextAsString)
    Dim Descriptor as String
    Dim Price As Integer

    ' assume that we want to parse the display data to get the
    'item descriptor and price
    Descriptor = ParseDesc (Row1Text)
    Price = ParsePrice(Row2Text)

    ' now do something with Price and Descriptor
End Sub
```

As an alternative to implementing an event sink to handle RPAMTerminal object, the client could simply wait an appropriate amount of time and then inspect the Row1Text and Row2Text properties of the ANDisplay property of the terminal.

Determining what constitutes an 'appropriate' amount of time is error prone though. If possible you should make every attempt to keep the instance of the terminal object around long enough to respond to any events generated by the application. The ability to return to the 4690 application later with another instance of the RPAMTerminal object is mentioned only to emphasize that the 4690 application instance is persistent across the instantiation and/or destruction of individual RPAMTerminal objects.

To disconnect from the sales application, clear the UnitId and/or TermNum and then issue the Disconnect command.

```
Sub CleanUp
    ' shut down the 4690 sales application
    Term.UnitID = 0
    Term.TermNum = 0

    'shutdown the comm like with TC
    Term.Disconnect
End Sub
```

Chapter 2. RPAMApp Object

Represents the entire QVS RPAM terminal application. The application object contains system-wide settings that control all RPAMTerminal objects instantiated. You can use the RPAMApp object to change the default behavior of the RPAM interface layer.

2.1 Properties

AutoDeviceConfigure
EventMode
OptimizeDisplay
StoreNum
TermNumAssignMode
Tracing

2.2 Methods

SetTCPortNumbers
SetTCServerAddresses
SetStoreNum

2.3 Application Object Reference

2.3.1 AutoDeviceConfigure Property

Syntax
BOOL AutoDeviceConfigure;

Remarks
Controls whether the RPAM interface layer or the client application assigns 4690 POS device IDs for the RPAMTerminal objects created.

When set to TRUE, the RPAM interface layer takes care of assigning the 4690 POS device IDs to the device objects. The device IDs are configured based on the terminal configuration data stored on the 4690 store controller. If AutoDeviceConfigure is set to FALSE then the client application is responsible for setting each device object's DeviceID property to a valid 4690 POS device ID before issuing the Connect command to the RPAMTerminal object. It's vitally important to understand that you should have all the 4690 controller terminal device configurations in the automatic TermNumAssignMode range set to the same device configuration if you are using automatic TermNum assignment.

2.3.2 EventMode Property

Syntax
LONG EventMode;

EventMode	Description
rpaSingleDisp	Single dispatch mode. This is the default and recommended mode.
rpaMultiDisp	Multiple dispatch mode. Not recommended.

Remarks
Identifies what event handling mechanism is being utilized by the RPAM interface layer on the client application.

The default value for this property is a single dispatch mode which uses the _IQVSRPAMEvents dispatch interface. This is the recommended mode if you are developing using the Microsoft MFC environment. This is the only mode used for the Java development environment. Notify your QVS technical advisor for further information.

2.3.3 OptimizeDisplay Property

Syntax
BOOL OptimizeDisplay;

Remarks
Controls whether the RPAM interface layer on the client application is to receive all display information no matter if the displayed contents has changed or not.

The default value for this property is TRUE so that the client only receives display updates if the display has changed.

2.3.4 SetStoreNum Method

Syntax
void SetStoreNum(long StoreNumber);

Remarks
Allows the user to set the store number.

2.3.5 SetTCPortNumbers Method

Syntax
void SetTCPortNumbers(long ServerFindPort, long ServerListenPort);

Parameter	Description
ServerFindPort	The TC server find port (default=7503).
ServerListenPort	The TC server listen port (default=7502).

Remarks
The RPAM client and TC server communicate using the Windows socket API internet domain (AF_INET). To communicate with the TC server the RPAM client must know the TC server IP address and port number. This method allows the user to override the default TC port numbers.

2.3.6 SetTCServerAddresses Method

Syntax
void SetTCServerAddresses(BSTR PrimaryServerAddress, BSTR BackupServerAddress);

Parameter	Description
PrimaryServerAddress	The primary TC server IP address (eg. “129.5.24.1”).
BackupServerAddress	The backup TC server IP address.

Remarks
The RPAM client and TC server communicate using the Windows socket API internet domain (AF_INET). To communicate with the TC server the RPAM client must know the TC server IP address and port number. This method allows the user to override the default TC server IP addresses. There will always be a primary TC server. There may also be a backup TC server available.

2.3.7 StoreNum Property

Syntax

LONG StoreNum;

Remarks
StoreNum contains the store number that the RPAM interface layer received from Terminal Concentrator. If Connnect() has not yet been called on this RPAMTerminal object, StoreNumber has not yet been determined and is set to 0.

2.3.8 TermNumAssignMode Property

Syntax
LONG TermNumAssignMode;

Remarks
Controls whether the RPAM interface layer or the client application assigns 4690 terminal numbers to the RPAMTerminal objects created.

Normally the RPAM interface layer takes care of assigning the 4690 terminal number to the RPAMTerminal object. The assigned terminal number is made available to the client application through the TermNum property of the terminal object. If you want to take control of how 4690 terminal numbers are assigned change this property to rpaAppTermNum and then set the TermNum properties accordingly. Default value rpaAutoTermNum

2.3.9 Tracing Property

Syntax
BOOL Tracing;

Remarks
Controls whether the RPAM interface layer or the client application will enable logging or disable logging.

When set to TRUE, the RPAM interface layer will enable the logging feature and when sent to FALSE the logging feature will be disabled.

Chapter 3. RPAMTerminal Object

The RPAMTerminal object provides the application with a high level interface to the device level I/O performed by a 4690 sales application. With the RPAMTerminal object an application is provided with a wide range of properties, methods and events that allow the application to inform the 4690 sales about hardware configurations and events, feed input to the 4690 application and monitor and respond to output performed by the 4690 sales application.

3.1 Properties

Terminal Properties
RunningTotal
TermNum
TwoPhaseConnect
UnitID

Terminal Devices
ANDisplay
ANDisplay2
CashDrawer
IOProcessor
Keyboard
MonitoredFiles
MSR
Printer
RPATermControl
Scale
Serial1
Serial2
TermControl
TotRet
VDisplay
VDisplay2

3.2 Methods

CompleteConnection
Connect
GetCDIField
ResumeConnection

3.3 Events

EnableIOProcWaitEvents

3.4 RPAMTerminal Object Reference

3.4.1 ANDisplay/ANDisplay2 Property

ANDisplay and ANDisplay2 properties are POSDisplay objects that represents the first and second 2x20 POS display devices of the RPAM terminal. See also **POSDisplay Object**.

Remarks
The POSDisplay object allows the user of the RPAM terminal object to monitor and respond to requests from the 4690 sales application to display data on the 2x20 POS displays. This object fires events when the

sales application modifies the display data and provides properties that allow you to see what the application has displayed.

3.4.2 CashDrawer Property

A POSCashDrawer object that represents the cash drawer device for the RPAM terminal. See also **POSCashDrawer Object**.

Remarks
The POSCashDrawer object fires an event whenever the 4690 sales application issues a drawer open command and is also used inform the sales application about the status of the cash drawers attached to the terminal. Status information such as the number of drawers present and whether they are opened or closed is conveyed to the sales application using the POSCashDrawerObject.

3.4.3 Connect Method

Syntax
HRESULT Connect();

Remarks
Causes the RPAM interface layer to initiate a physical connection to the 4690 store controller. You must issue a Connect() on the RPAMTerminal object before attempting any actions such as file I/O that require interaction with the store controller or Terminal Concentrator/NT.

3.4.4 EnableIOProcWaitEvents Method

Syntax
HRESULT EnableIOProcWaitEvents();

Remarks
This enables the client to receive I/O Processor wait events.

3.4.5 GetCDIField Method

Syntax
void GetCDIField(BSTR, BSTR, BSTR, Variant);

Remarks
This method allows the client to retrieve CDIFile field parameter information. Note: This currently only works in the Java client not the Windows client.

3.4.6 IOProcessor Property

A IOProcessor object that represents the I/O processor device of the RPAM terminal. See also **IOProcessor Object**.

Remarks
Use the IOProc property object to send keyboard, keylock and scanner data to the application. The IOProcessor object also fires events as the application locks and unlocks the I/O processor and as keyboard indicator lights are turned on and off

3.4.7 Keyboard Property

A keyboard object that represents the keyboard device of the RPAM terminal. See also **Keyboard Object**.

Remarks

The MSR MagStripeReader object fires an event whenever the 4690 sales application issues a lock, unlock or read to the MSR device. The MagStripeReader is also used inform the sales application about the track reading capabilities of the MSR device and to query the RPAM interface layer to determine if the MSR is locked or not.

3.4.8 MonitoredFiles Property

A collection of the MonitoredFile objects requested by the client of the RPAMTerminal object. See also **MonitoredFile Object**.

Remarks
A monitored file object lets the client of the RPAMTerminal object watch file reads and writes performed by the 4690 sales application. The MonitoredFiles property is a collection of the file objects currently being monitored.

3.4.9 MSR Property

A POSMSR (Mag Stripe Reader) object that represents the MSR device of the RPAM terminal. See also **POSMSR Object**.

Remarks
The POSMSR object fires an event whenever the 4690 sales application issues a lock, unlock or read to the MSR device. The MSR is also used inform the sales application about the track reading capabilities of the MSR device and to query the RPAM interface layer to determine if the MSR is locked or not.

3.4.10 Printer Property

A POSPrinter object that represents the POS printer device of the RPAM terminal. See also **POSPrinter Object**.

Remarks
The Printer POSPrinter object allows you to inform the 4690 sales application about the status of the printer attached to the terminal and to respond to requests from the application to print data at the various print stations.

3.4.11 ResumeConnection Method

Syntax
void ResumeConnection();

Remarks
Causes the RPAM interface layer to resume connection to the TC server. This is normally done when the client is a handheld MPOS device and the unit is powered off or goes to sleep and then is powered on at a later time.

3.4.12 RunningTotal Property

Syntax
UNSIGNED LONG RunningTotal;

Remarks
This property holds the running transaction total.

3.4.13 Scale Property

A Scale object that represents the scale device of the RPAM terminal. See also **Scale Object**.

Remarks
The scale object fires an event whenever the 4690 sales application issues a scale reading.

3.4.14 Scanner Property

A scanner object that represents the scanner device of the RPAM terminal. See also **Scanner Object**.

Remarks
The scanner object fires an event whenever the 4690 sales application issues a scan.

3.4.15 Serial1/Serial2 Property

A POSSerialDevice object that represents serial communications ports 1 & 2 of the RPAM terminal. See also **POSSerialDevice Object**.

Remarks
The POSSerialDevice object allows your RPAM client application to interact with 4690 sales applications that read data from and write data to the serial communications ports.

3.4.16 StartApplication Method

Syntax
void StartApplication();

Remarks
Use the StartApplication method to kick off the execution of the 4690 sales application under Terminal Concentrator/NT. You need to make sure that you've properly informed the RPAM interface layer what devices you are supporting (or let RPAM automatically configure the devices) before calling this method..

3.4.17 TermNum Property

Syntax
LONG TermNum;

Remarks
Contains the 4690 terminal ID for the RPAM terminal object. If the TermNumAssignMode property of the application is set to rpaAutoTermNum you should treat this property as read-only. If TermNumAssignMode is set to rpaAppTermNum you are responsible for setting this property to a valid 4690 terminal ID before you issue a connect to the RPAM interface layer.

3.4.18 TermControl Property

Represents the RPAM 'terminal control' device of the RPAM terminal. See also **RPATermControl Object**.

Remarks
This RPA 'pseudo' device is a mechamism that the 4690 application developer can take advantage of to send custom messages and data to this RPAMTerminal object.

3.4.19 TotRet Property

Represents the battery backed totals retention device of the RPAM terminal. See also **HardTotals Object**.

Remarks
Use this property to respond to the 4690 sales application's requests to read and write hard totals data

3.4.20 TwoPhaseConnect Property

Syntax
LONG TwoPhaseConnect;

Remarks
This property is used when the if the client wants to retrieve CDI field information before coming officially online to the TC server.

3.4.21 UnitID Property

Syntax
LONG UnitID;

Remarks
Contains the client assigned unique ID number for the physical terminal. A good number to use for this property is the MAC ID of the network interface card installed in the terminal. To shut down the 4690 sales application set this property to 0. Read/write Long.

3.4.22 VDisplay/VDisplay2 Property

VDisplay and VDisplay2 are VideoDisplay objects representing the two video display devices of the RPAM terminal. See also **VideoDisplay Object**.

Remarks
Use VDisplay and VDisplay2 property objects to respond to the 4690 sales application's requests to display information on the two video displays.

Chapter 4. QVSRPAMEvents Object (single dispatch mode)

The QVSRPAMEvents object provides the application with a high level interface to the device level I/O performed by a 4690 sales application. With the QVSRPAMEvents object an application is provided with a wide range of events that allow the application to inform the 4690 sales events, feed input to the 4690 application and monitor and respond to output performed by the 4690 sales application.

This implementation was done to ease the programming development in the MFC C++ environment.

4.1 Events

Cash Drawer Events
DrawerOpenEvent()

Display Events
ActivateMarqueeDisplayEvent()
CursorVisibilityChangeEvent()
DisplayEvent()
DisplayEventEx()
DisplayFormatChangeEvent()
DeactivateMarqueeDisplayEvent()
LoadMarqueeDisplayEvent()
VDisplayEvent()

Keyboard Events
MessageLightEvent()
OfflineLightEvent()
ToneEvent()
WaitLightEvent()

Hardtotals Events
OfflineTotalsDataEvent()
TotalsReadEvent()
TotalsWriteEvent()

IOProc Events
AlphaEntryModeEnterEvent()
AlphaEntryModeExitEvent()
FieldLevelInputEvent()
FieldLevelInputEvent2()
IOProcErrEvent()
IOProcWaitEvent()
LockEvent()
ReadEvent()
UnlockEvent()

MICR Events
ActivateMICREvent()

MSR Events
LockMSR()
UnlockMSR()

Printer Events
PrintEjectEvent()
PrintEvent()
PrintExEvent()
ReceiptCutEvent()

Scale Events
ScaleRead()

Scanner Events
ScannerLockEvent ()
ScannerUnlockEvent ()

Serial Events
SerialCloseEvent()
SerialOpenEvent()
SerialPutLongEvent()
SerialWriteEvent()

Terminal Events
ConnectResumeAttemptEvent()
ConnectResumeCompleteEvent()
ConnectTimeoutEvent()
OfflineEvent()
OnlineEvent()

Terminal Control Events
RunningTotalDataEvent()
TermIPLControlEvent()
TermNumberChangedEvent()
TermReloadEvent()
UserDataEvent()

4.2 QVSRPAMEvents Object Reference

4.2.1 AlphaEntryModeEnterEvent

Syntax
HRESULT AlphaEntryModeEnterEvent ();

Remarks
The client is being notified that the I/O Processor will allow alpha entry for the current state.

4.2.2 AlphaEntryModeExitEvent

Syntax
HRESULT AlphaEntryModeExitEvent ();

Remarks
The client is being notified that the I/O Processor will disallow alpha entry for the current state.

4.2.3 ActivateMarqueeDisplayEvent

Syntax
HRESULT ActivateMarqueeDisplayEvent ();

Remarks

The client is being notified that the display is now in the marquee mode where the display will scroll a message on the display.

4.2.4 ActivateMICREvent

Syntax
HRESULT ActivateMICREvent();

Remarks
The client is being notified that the MICR device is now activated.

4.2.5 ConnectResumeAttemptEvent

Syntax
HRESULT ConnectResumeAttemptEvent ();

Remarks
The client was online to the TC server at one time an is in the process of trying to reconnect..

4.2.6 ConnectResumeCompleteEvent

Syntax
HRESULT ConnectResumeCompleteEvent ();

Remarks
The client was online to the TC server at one time and has now completed the reconnection.

4.2.7 ConnectTimeoutEvent

Syntax
HRESULT ConnectTimeoutEvent ();

Remarks
Occurs when a timeout occurs when the client tries to come online to the TC server.

4.2.8 CursorVisibilityChangeEvent

Syntax
HRESULT CursorVisibilityChangeEvent (LONG Device, LONG CursorVisible);

Parameter	Description
Device	Display device.
CursorVisible	A zero value means the cursor is not visible.

Remarks
This event is sent when the cursor visibility is changed by the 4690 POS application to either make the cursor visible or not. Applicable for applications that use the video display only.

4.2.9 DeactivateMarqueeDisplayEvent

Syntax
HRESULT DeactivateMarqueeDisplayEvent ();

Remarks
The client is being notified that the display is deactivating the marquee mode where the display will scroll a message on the display.

4.2.10 DisplayEvent

Syntax
HRESULT DisplayEvent (LONG Device, BSTR Row1Text, BSTR Row2Text);

Parameter	Description
Device	0=ANDISPLAY 1=ANDISPLAY2
Row1Text	The text currently displayed on row 1 of the POS display.
Row2Text	The text currently displayed on row 2 of the POS display.

Remarks
The client is notified with this event when the application writes text to the list of defined display devices described in the device parameter above.

4.2.11 DisplayEventEx

Syntax
HRESULT DisplayEventEx (LONG Device, BSTR Row1Text, BSTR Row2Text,
LONG UpdateFlags);

Parameter	Description
Device	0=ANDISPLAY 1=ANDISPLAY2
Row1Text	The text currently displayed on row 1 of the POS display.
Row2Text	The text currently displayed on row 2 of the POS display.
UpdateFlags	Specifies which display row was written to.

Remarks
The client is notified with this event when the application writes text to the list of defined display devices described in the device parameter above.

4.2.12 DisplayFormatChangeEvent

Syntax
HRESULT DisplayFormatChange (LONG Device, LONG DisplayFormat);

Parameter	Description
Device	Display device.
DisplayFormat	0 = 25x80 8 = 12x40 24 = 16x60 32 = 16x80

Remarks
This event is sent when the display format is changed by the 4690 POS application. Applicable for applications that use the video display only.

4.2.13 DrawerOpenEvent

Syntax
HRESULT DrawerOpenEvent(LONG DrawerNumber);

Parameter	Description
DrawerNumber	The cash drawer number that was opened (eg. 1).

Remarks
The POSCashDrawer object fires an event whenever the 4690 sales application issues a drawer open command and is also used inform the sales application about the status of the cash drawers attached to the terminal. Status information such as the number of drawers present and whether they are opened or closed is conveyed to the sales application using the POSCashDrawerObject.

4.2.14 FieldLevelInputEvent

Syntax

HRESULT FieldLevelInputEvent(LONG Maxlen, LONG EchoInput, LONG FirstDisplayCol,
LONG LastDisplayCol, LONG StateNum);

Parameter	Description
Maxlen	Maximum length of data to be returned to the application.
EchoInput	Specifies whether data should be echoed to the screen.
FirstDisplayCol	First display column for keyed data.
LastDisplayCol	Last display column for keyed data.
StateNum	Current input state number.

Remarks

This event occurs when the I/O Processor device unlocks to a new state and the field level input is enabled.

4.2.15 FieldLevelInputEvent2

Syntax

HRESULT FieldLevelInputEvent2(LONG Maxlen, LONG EchoInput, LONG FirstDisplayCol,
LONG LastDisplayCol, LONG StateNum, BOOL AlphaEntry,
LONG TransitionFlag);

Parameter	Description
Maxlen	Maximum length of data to be returned to the application.
EchoInput	Specifies whether data should be echoed to the screen.
FirstDisplayCol	First display column for keyed data.
LastDisplayCol	Last display column for keyed data.
StateNum	Current input state number.
AlphaEntry	Indicates whether alpha entry is allowed in this state.
TransitionFlag	Indicates the reason for the unlock into this state.

Remarks

This event occurs when the I/O Processor device unlocks to a new state and the field level input is enabled. Only valid when the event mode is a single dispatch implementation

4.2.16 IOProcErrEvent

Syntax

HRESULT IOProcErrEvent();

Remarks

Occurs when the IO processor detects an error.

4.2.17 IOProcWaitEvent

Syntax

HRESULT IOProcWaitEvent();

Remarks

This occurs when the sales application is ready to accept input from the IO processor.

4.2.18 LoadMarqueeDisplayEvent

Syntax

HRESULT LoadMarqueeDisplayEvent ();

Remarks

The client is being notified that the display is loading the proper information to enable the marquee mode where the display will scroll a message on the display.

4.2.19 LockEvent

Syntax
HRESULT LockEvent();

Remarks
This occurs when the sales application issues a lock command to the IO processor.

4.2.20 LockMSR

Syntax
HRESULT LockMSR();

Remarks
The client is being notified that the MSR device is now locked and unavailable.

4.2.21 MessageLightEvent

Syntax
HRESULT MessageLightEvent(BOOL NewValue);

Parameter	Description
NewValue	TRUE = Message light is on. FALSE = Message light is off.

Remarks
This event is sent to the client when the status of the message light changes. This corresponds to the message light displayed on a real POS terminal keyboard.

4.2.22 OfflineEvent

Syntax
HRESULT OfflineEvent();

Remarks
Occurs after the RPAM terminal loses its connection to Terminal Concentrator.

4.2.23 OfflineLightEvent

Syntax
HRESULT OfflineLightEvent(BOOL NewValue);

Parameter	Description
NewValue	TRUE = Offline light is on. FALSE = Offline light is off.

Remarks
This event is sent to the client when the status of the offline light changes. This corresponds to the offline light displayed on a real POS terminal keyboard.

4.2.24 OnlineEvent

Syntax
HRESULT OnlineEvent();

Remarks
Occurs after the RPAM terminal has successfully connected to Terminal Concentrator.

4.2.25 OfflineTotalsDataEvent

Syntax
HRESULT OfflineTotalsDataEvent(SAFEARRAY Data, LONG Datalen);

Parameter	Description
Data	The offline totals data.
Datalen	Length of the offline totals data.

Remarks
Occurs when the sales application issues an update of the offline totals.

4.2.26 PrintEjectEvent

Syntax
HRESULT PrintEjectEvent();

Remarks
Occurs when the sales application issues an eject command to the printer.

4.2.27 PrintEvent

Syntax
HRESULT PrintEvent(LONG Station, BSTR PrintLine, LONG LineFeeds);

Parameter	Description
Station	Which station to print on. Journal, receipt, or the document insertion station.
PrintLine	The print line to be printed.
LineFeeds	Number of line feeds to be performed.

Remarks
This event occurs when the sales application prints data to one of the print stations on the POS printer device for a model 1 or model 2 printer.

4.2.28 PrintExEvent

Syntax
HRESULT PrintExEvent(LONG Station, LONG Command, LONG Linefeeds, LONG Datalen,
LONG Flags, VARIANT Data);

Parameter	Description
Station	Which station to print on. Journal, receipt, or the document insertion station.
Command	The print command to be issued.
LineFeeds	Number of line feeds to be performed.
Datalen	Length of the data to print.
Flags	Reserved field.
Data	The print line to be printed.

Remarks
This event occurs when the 4690 POS application prints data to one of the print stations on the model 3 or model 4 POS printer devices.

4.2.29 ReadEvent

Syntax
HRESULT ReadEvent();

Remarks
This occurs when the sales application issues a read command to the IO processor.

4.2.30 RunningTotalDataEvent

Syntax
HRESULT RunningTotalDataEvent(UNSIGNED LONG RunningTotal);

Parameter	Description
RunningTotal	Current transaction total amount.

Remarks
This event retrieves the running transaction total.

4.2.31 ReceiptCutEvent

Syntax
HRESULT ReceiptCutEvent();

Remarks
Occurs when the sales application issues a cut command to the printer.

4.2.32 ScaleRead

Syntax
HRESULT ScaleRead();

Remarks
This occurs when the sales application issues a read command to the scale device.

4.2.33 ScannerLockEvent

Syntax
HRESULT ScannerLockEvent();

Remarks
The client is being notified that the scanner device is now locked and unavailable.

4.2.34 ScannerUnlockEvent

Syntax
HRESULT ScannerLockEvent();

Remarks
The client is being notified that the scanner device is now unlocked and available to be used

4.2.35 SerialCloseEvent

Syntax
HRESULT SerialCloseEvent();

Remarks
Please see remarks under the CloseEvent() for the POSSerialDevice object.

4.2.36 SerialOpenEvent

HRESULT SerialOpenEvent(long Port, long Speed, BSTR Parity, long DataBits, long StopBits, long BuffSize);

Parameter	Description
Port	Communication port number (eg. 1)
Speed	Transmit/receive bit rate in bits per second.
Parity	Transmit/receive parity. “E” for event, “O” for odd, “N” for none.
DataBits	Number of transmit/receive data bits.

StopBits	Number of stop bits.
BuffSize	Size of the I/O buffer in bytes.

Remarks
This event is fired when the 4690 application issues an open to the serial port. Your application should take the appropriate action to open the physical or virtual communications port when you receive this event.

Refer to the 4690 Basic Language reference for more information on the parameters to this event.

4.2.37 SerialPutLongEvent

HRESULT SerialPutLongEvent(long Port, long TimeOut, long PortStatusValue);

Parameter	Description
Port	Communication port number (eg. 1)
TimeOut	The timeout value in milliseconds.
PortStatusValue	The port status information which the 4690 sales application wrote with the PUTLONG command it issued. This value corresponds to the CC byte of the PUTLONG value. See the 4690 Basic Language reference for more information on the bit fields that make up the CC byte.

Remarks
This event is fired when the 4690 application issues a PUTLONG to the serial port. Your application should take the appropriate action to configure the physical or virtual communications port when you receive this event.

4.2.38 SerialWriteEvent

HRESULT Write(long Port, long Length, SAFEARRAY Data);

Parameter	Description
Port	Communication port number (eg. 1)
Length	The number of bytes of data you wish to send to the 4690 application.
Data	The data to be sent to the 4690 application Data is a SAFEARRAY of type VT_UI1 (unsigned character array). .

Return Value

Returns the 0 if the method succeeded. This method will always succeed unless the RPAM client is offline from terminal concentrator.

Remarks
Call this method to send serial data to the 4690 sales application. Whenever you have read data from the port you are managing you should call this method with Length set to the number of bytes contained in the Data paramater and Data should contain the actual data you read from the port

4.2.39 TermIPLControlEvent

Syntax
HRESULT TermIPLControlEvent(BOOL Enable);

Parameter	Description
Enable	Boolean value

Remarks
Occurs when the terminal is IPL'd.

4.2.40 TermNumberChangedEvent

Syntax
HRESULT TermNumberChangedEvent();

Remarks
Issued when the terminal number changes. Occurs after the S1-71-S2 processing has been initiated by the terminal operator and the TC server has formally assigned the new terminal.

4.2.41 TermReloadEvent

Syntax
HRESULT TermReloadEvent();

Remarks
Occurs when the sales application is reloaded.

4.2.42 ToneEvent

Syntax
HRESULT ToneEvent(LONG Frequency, LONG Duration);

Parameter	Description
Frequency	The frequency to issue the tone sound.
Duration	The duration of the tone sound. 0 = turn tone off -1 = turn tone on x = dureation in milliseconds

Remarks
This occurs when the sales application issues a command to the tone device.

4.2.43 TotalsReadEvent

Syntax
HRESULT TotalsReadEvent(LONG Length, LONG Offset);

Parameter	Description
Length	Number of bytes of hard totals data requested by the application.
Offset	Starting offset of where to read from in the hard totals area.

Remarks
Occurs when the 4690 sales application issues a write command to the hard totals device.

4.2.44 TotalsWriteEvent

Syntax
HRESULT TotalsWriteEvent(SAFEARRAY Data, LONG Length, LONG Offset);

Parameter	Description
Data	Data to be written to the hard totals device.
Length	Number of bytes of hard totals data to write.
Offset	Starting offset of where to write to in the hard totals area.

Remarks
Occurs when the 4690 sales application issues a write command to the hard totals device.

4.2.45 UnlockEvent

Syntax

HRESULT UnlockEvent();

Remarks
The client is being notified that the I/O Processor has just unlocked to a state.

4.2.46 UnlockMSR

Syntax
HRESULT UnlockMSR();

Remarks
The client is being notified that the MSR device is now unlocked and available to be used

4.2.47 UserDataEvent

Syntax
HRESULT UserDataEvent(SAFEARRAY Data, LONG Datalen);

Parameter	Description
Data	Data to be written.
Length	Number of bytes to write.

Remarks
Occurs when the 4690 sales application issues a user defined message to the client.

4.2.48 VDisplayEvent

Syntax
HRESULT VDisplayEvent(LONG Device, LONG Row, LONG Column, LONG Length,
BSTR Text, BSTR Attributes, LONG Flags);

Parameter	Description
Device	0=ANDISPLAY 1=ANDISPLAY2
Row	Row number to display text.
Column	Column number that display starts.
Length	Length of text to display.
Text	Text to display.
Attributes	Attributes of the text to display.
Flags	Reserved flag information.

Remarks
This event is sent when the text is sent to the display by the 4690 POS application. Applicable for applications that use the video display only.

4.2.49 WaitLightEvent

Syntax
HRESULT WaitLightEvent(BOOL NewValue);

Parameter	Description
NewValue	TRUE = Wait light is on. FALSE = Wait light is off.

Remarks
This event is sent to the client when the status of the wait light changes. This corresponds to the wait light displayed on a real POS terminal keyboard.

Chapter 5. POSCashDrawer Object (multiple dispatch mode)

The POSCashDrawer object represents a virtual cash drawer device to the 4690 sales application. This object supports up to two cash drawers. The sales application uses this device by querying it for the open status of the drawers and by issuing drawer open commands to the device. You can use this object to inform the application of the status of the cash drawers attached to the terminal and for responding to the application's requests to open the cash drawers.

5.1 Properties

ConfiguredID
DeviceID
DeviceResponseMode
Drawer1Opened
Drawer1Present
Drawer2Opened
Drawer2Present
DrawerOpenMsec

5.2 Events

DrawerOpenEvent

5.3 POSCashDrawer Object Reference

5.3.1 ConfiguredID Property

Syntax
LONG ConfiguredID;

Remarks
The ConfiguredID property contains the 4690 POS device id for this device as loaded from the terminal device configuration record for this object's terminal number. The RPAM interface layer will set this property based on the store controller configuration information and it is provided to the user for informational purposes only.

If you are using automatic device configuration the ConfiguredID and DeviceID will have the same value. If you're using manual device configuration it is your responsibility to ensure that the value you choose for DeviceID is compatible with the application's requirements. Read-only LONG.

5.3.2 DeviceID Property

Syntax
LONG DeviceID;

Remarks
The DeviceID property contains the 4690 POS device id for this device. If you are using automatic terminal device configuration, this property is read-only as the RPAM interface layer will manage its value based on the 4690 terminal device configuration.

If you are using manual device configuration and you intend to emulate this device, it's your responsibility to set this property to a valid POS cash drawer device ID device before calling Connect for the terminal.

5.3.3 Drawer1Open Property

Syntax

BOOL Drawer1Open;

Remarks
True if the first cash drawer attached to the physical terminal is open. Read/write Boolean
Default Value: TRUE.

5.3.4 Drawer1Present Property

Syntax
BOOL Drawer1Present;

Remarks
True if there is a cash drawer attached to the physical terminal. Read/write Boolean.
Default Value: TRUE.

5.3.5 Drawer2Open Property

Syntax
BOOL Drawer2Open;

Remarks
True if the second cash drawer attached to the physical terminal is open. Read/write Boolean.
Default Value: FALSE.

5.3.6 Drawer2Present Property

Syntax
BOOL Drawer2Present;

Remarks
True if there is a second cash drawer attached to the physical terminal. Read/write Boolean.
Default Value: FALSE.

5.3.7 DrawerOpenEvent

Syntax
void DrawerOpenEvent (LONG DrawerNumber);

Parameter	Description
DrawerNumber	Drawer number being opened. 1 for first drawer, 2 for second.

Remarks
Occurs when the 4690 sales application issues a drawer open command to the cash drawer device.

5.3.8 DrawerOpenMsec Property

Syntax
LONG DrawerOpenMsec;

Remarks
When running the POSCashDrawer object in rpaAutoDeviceResponse mode (where the RPAMInterface layer is handling return codes from the device) this property controls how long in milliseconds that the RPAM interface layer reports to the application that the drawer is open before reporting that it's closed.
Default value 1000.

5.3.9 DeviceResponseMode Property

Syntax

LONG DeviceResponseMode;

Remarks
Informs the RPAM interface layer who will be responsible for providing return code data to the 4690 sales application on device writes. Set this property to rpaAutoDeviceResponse to have the RPAM interface layer manage the drawer properties after an open command has been received. Set it to rpaAppDeviceResponse if your application will be responsible for managing the drawer status information. Default value: rpaAutoDeviceResponse.

Chapter 6. POSHardTotals Object (multiple dispatch mode)

The POSHardTotals object represents a virtual totals retention device to the 4690 sales application. The totals retention device is battery backed memory that the terminal uses to store configuration and system information data and that the application uses to store data that will be saved if the terminal loses power for a prolonged period of time. The application uses this device by reading from and writing to the totals retention driver. You can optionally use this object if you desire to provide hard totals data space in your remote device. This is normally not necessary except for dual-concentrator environments where one concentrator backs up the other.

6.1 Properties

ConfiguredID
DeviceID
DeviceResponseMode

6.2 Methods

SendTotalsData
SendTotalsData2
SendTotalsWriteResponse
SetInstanceData

6.3 Events

DoFireTotalsReadEvent
DoFireTotalsWriteEvent

6.4 POSHardTotals Object Reference

6.4.1 ConfiguredID Property

Syntax
LONG ConfiguredID;

Remarks
The ConfiguredID property contains the 4690 POS device id for this device as loaded from the terminal device configuration record for this object's terminal number. The RPAM interface layer will set this property based on the store controller configuration information and it is provided to the user for informational purposes only.

If you are using automatic device configuration the ConfiguredID and DeviceID will have the same value. If you're using manual device configuration it is your responsibility to ensure that the value you choose for DeviceID is compatible with the application's requirements. Read-only LONG.

6.4.2 DeviceID Property

Syntax
LONG DeviceID;

Remarks
The DeviceID property contains the 4690 POS device id for this device. If you are using automatic terminal device configuration, this property is read-only as the RPAM interface layer will manage its value based on the 4690 terminal device configuration.

If you are using manual device configuration and you intend to emulate this device, it's your responsibility to set this property to a valid POS cash drawer device ID device before calling Connect for the terminal.

6.4.3 DeviceResponseMode Property

Syntax
LONG DeviceResponseMode;

Remarks
Informs the RPAM interface layer who will be responsible for providing return code data to the 4690 sales application on device writes. Can be either rpaAutoDeviceResponse to have the RPAM interface layer return good return codes on all device writes or rpaAppDeviceResponse if your application will be responsible for providing return code data to the 4690 sales application. Default value: rpaAutoDeviceResponse.

6.4.4 DoFireTotalsReadEvent Event

Syntax
void DoFireTotalsReadEvent(LONG Offset, LONG Length);

Parameter	Description
OffSet	The offset within hard totals memory to start reading from.
Length	Number of bytes of data requested by the application.

Remarks
Occurs when the 4690 sales application issues a direct read command to the hard totals device.

6.4.5 DoFireTotalsWriteEvent Event

Syntax
void DirectWriteEvent(VOID *BUf, LONG Length , LONG Offset);

Parameter	Description
Buf	The data written to hard totals by the application.
Length	Number of bytes of data written by the application.
OffSet	The offset within hard totals memory to start writing to.

Remarks
Occurs when the 4690 sales application issues a direct write command to the hard totals device.

6.4.6 SendTotalsData Method

Syntax
void SendTotalsData(VOID *BUf, LONG Length);

Parameter	Description
Buf	The data to be sent to the application.
Length	Number of bytes of data to be sent to the application.

Remarks

Use this method when you want to send hard totals direct read data to the 4690 sales application.

6.4.7 SendTotalsData2 Method

Syntax
void SendTotalsData2(VOID *BUf, LONG Length);

Parameter	Description
Buf	The data to be sent to the application.
Length	Number of bytes of data to be sent to the application.

Remarks
Use this method when you want to send hard totals direct read data to the 4690 sales application.

6.4.8 SendTotalsWriteResponse Method

Syntax
void SendHardTotalsDirectWriteResponse(LONG Rc);

Parameter	Description
Rc	The return code from the write operation to be passed back to the application.

Remarks
Use this method when you want to pass a hard totals write return code back to the 4690 sales application.

6.4.9 SetInstanceData Method

Syntax
void SetInstanceData(LONG Instance, LONG UnitID, INT RPAMDevice);

Parameter	Description
Instance	Instance
UnitID	Unit ID
RPAMDevice	TOTRET

Chapter 7. IOProcessor Object (multiple dispatch mode)

The IOProcessor object represents a virtual 4690 I/O processor to the 4690 sales application. The I/O processor is a driver that processes input from the keyboard and scanner according to the input sequence tables stored on the 4690 controller. The application uses this device by issuing reads to the device as well as issuing lock and unlock commands to the driver. You can use this object to provide keyboard and scanner data to the application and to respond to the lock and unlock commands sent by the application.

7.1 Properties

AlphaEntryMode
ConfiguredID
DeviceID
FieldLevelInputMode
FLIEchoInput
FLIFirstDisplayCol
FLIMaxLen
IsLocked
MgrKeyStatus
ResultCode
ScannerLocked
StateNum
TransitionFlag

7.2 Methods

ExitAlphaEntryMode
SendScannerData
SendKeyboardData
SendKeyboardScanCode
SetFLIData
SetInstanceData

7.3 Events

Do_Fire_FieldLevelInputEvent
Do_Fire_IOProcErrEvent
Do_Fire_IOProcWaitEvent
Do_Fire_LockEvent
Do_FireScanLockEvent
Do_FireScanUnlockEvent
Do_Fire_UnlockEvent

7.4 IOProcessor Object Reference

7.4.1 AlphaEntryMode Property

Syntax
BOOL AlphaEntryMode;

Remarks
AlphaEntryMode is set to TRUE when 4690 sales application's I/O processor device enters a state where alpha entry mode is enabled. Otherwise FALSE. Note: Calling the ExitAlphaEntryMode() method of the IOProcessor object causes the AlphaEntryMode property to be set to FALSE which causes the RPAM

interface layer to discontinue marking keystrokes sent to the 4690 sales application's I/O processor as alpha keys.

7.4.2 ConfiguredID Property

Syntax
LONG ConfiguredID;

Remarks
The ConfiguredID property contains the 4690 POS device id for this device as loaded from the terminal device configuration record for this object's terminal number. The RPAM interface layer will set this property based on the store controller configuration information and it is provided to the user for informational purposes only.

If you are using automatic device configuration the ConfiguredID and DeviceID will have the same value. If you're using manual device configuration it is your responsibility to ensure that the value you choose for DeviceID is compatible with the application's requirements. Read-only LONG.

7.4.3 DeviceID Property

Syntax
LONG DeviceID;

Remarks
The DeviceID property contains the 4690 POS device id for this device. If you are using automatic terminal device configuration, this property is read-only as the RPAM interface layer will manage its value based on the 4690 terminal device configuration.

If you are using manual device configuration and you intend to emulate this device, it's your responsibility to set this property to a valid POS cash drawer device ID device before calling Connect for the terminal.

7.4.4 Do_Fire_FieldLevelInputEvent

Syntax
void Do_Fire_FieldLevelInputEvent(LONG MaxLen, LONG EchoInput, LONG FirstDisplayCol, LONG LastDisplayCol, LONG StateNum);

Parameter	Description
MaxLen	Maximum length of the data to be returned to the application.
EchoInput	Says whether data should be echoed to the screen. Can be one of the following values: KBD_DISP, KBD_DISP_EDIT or KBD_NO_DISP.
FirstDisplayCol	First display column for keyed data.
LastDisplayCol	Last display column for keyed data.
StateNum	Current input state number.

Remarks
Occurs when the 4690 I/O processor device unlocks to a new state and the FieldLevelInputMode property is set to rpaFlApProc or rpaFlRpStates.

7.4.5 Do_Fire_IOProcErrEvent

Syntax
void Do_Fire_IOProcErrEvent (LONG ErrCode);

Parameter	Description
ErrCode	I/O Processor error code..

Remarks
Occurs when the 4690 sales application issues a command that produces an I/O Processor error.

7.4.6 Do_Fire_IOProcWaitEvent

Syntax
void Do_Fire_IOProcWaitEvent ();

Remarks
Occurs when the 4690 sales application issues a command to notify the application that the I/O Processor has issued a wait event.

7.4.7 Do_Fire_LockEvent

Syntax
void Do_Fire_LockEvent ( );

Remarks
Occurs when the 4690 sales application issues a lock command to the I/O processor.

7.4.8 Do_FireScanLockEvent

Syntax
void Do_FireScanLockEvent ();

Remarks
Occurs when the 4690 sales application issues a command to lock the scanner.

7.4.9 Do_FireScanUnlockEvent

Syntax
void Do_FireScanUnlockEvent ();

Remarks
Occurs when the 4690 sales application issues a command to unlock the scanner.

7.4.10 Do_Fire_UnlockEvent

Syntax
void Do_Fire_UnlockEvent (BOOL KeyboardLocked, BOOL ScannerLocked);

Parameter	Description
Keyboard	LockedTrue if the keyboard is locked in this state, otherwise false.
ScannerLocked	True if the scanner is locked in this state, otherwise false.

Remarks
Occurs when the 4690 sales application issues an unlock command to the I/O processor.

7.4.11 ExitAlphaEntryMode Method

Syntax
void ExitAlphaEntryMode();

Remarks

Call this method to inform the RPAM interface layer that the client application no longer wishes to send keyboard input to the 4690 sales application's I/O processor with keys marked as alpha keys.

7.4.12 FieldLevelInputMode Property

Syntax
LONG FieldLevelInputMode;

Remarks
Field level input is a buffering scheme where the RPAM client is expected by Terminal Concentrator/NT to manage field input and display until a function code is pressed. When an RPAM client handles field level input much less network traffic is generated and operator input is potentially more responsive. Field level input can be handled either by the RPAM interface layer or by the client application. In an actual 4690OS terminal, field level input is handled by the I/O processor driver/process. Valid values for this property are:

rpaFlApProc	The client application will accept and process all field level data.
rpaFlRpProc	The RPAM interface layer will gather keyboard data and display it as needed.
rpaFlRpStates	Field level data will be passed up to the client application by the RPAM interface layer so the client can see application state changes but the RPAM interface layer will still gather and redisplay as needed.

Default value: rpaFlRpProc.

7.4.13 FLIEchoInput Property

Syntax
LONG FLIEchoInput;

Remarks
Non-zero if the I/O processor for the 4690 sales application is echoing data in field level input mode. Zero if no echo.

7.4.14 FLIFirstDisplayCol Property

Syntax
LONG FLIFirstDisplayCol;

Remarks
The first display column of the field level input data as defined by the I/O processor.

7.4.15 FLIMaxLen Property

Syntax
LONG FLIMaxLen;

Remarks
The maximum length of data accepted for field level input data as defined by the I/O processor.

7.4.16 IsLocked Property

Syntax
BOOL IsLocked;

Remarks
True when the I/O processor for the 4690 sales application is locked. Read-only Boolean. Default value: TRUE.

7.4.17 MgrKeyStatus Property

Syntax
LONG MgrKeyStatus;

Remarks
Non-zero if a manager key is present (turned on). Zero if no manager key is present.

7.4.18 ResultCode Property

Syntax
LONG ResultCode;

Remarks
This property contains the status of the most recent method or the most recently changed property. Read-only Long.

7.4.19 ScannerLocked Property

Syntax
BOOL ScannerLocked;

Remarks
True when the scanner should be locked, false otherwise.
Read-only BOOL. Default value: TRUE

7.4.20 SendKeyboardData Method

Syntax
void SendKeyboardData(LONG FunctionCode);

Parameter	Description	
FunctionCode	Number identifying the keyboard function code you wish to send to the application. Function codes above 60 are application defined.	
	1	S1 special code
	2	S2 special code
	3	Double 0 special code
	4	Triple 0 special code
	48-57 (x30 - x39)	Numeric keypad codes 0 - 9
	61-255 (x3D - xFF)	4690 keyboard function codes

Remarks
Call this method to send keyboard function code data to the 4690 sales application.

7.4.21 SendKeyboardScanCode Method

Syntax
void SendKeyboardScanCode(LONG ScanCode);

Remarks
Call this method to send keyboard scan codes to the 4690 sales application.

ScanCode must be one of the codes defined in the IBM Point of Sale Subsystem: Installation, Keyboards and Code Pages Book.

7.4.22 SendScannerData Method

Syntax

void SendScannerData(LONG LabelFormat, BSTR LabelData);

Parameter	Description
LabelFormat	Number identifying the format of the label read from the scanner. Valid values are: LBL__CODE39 LBL__CODE93 LBL__CODE128 LBL__CODABAR LBL_EAN_8 LBL__EAN8_2 LBL__EAN8_5 LBL_EAN_13 LBL__EAN13_2 LBL__EAN13_5 LBL_UPC_A LBL__UPCA_2 LBL__UPCA_5 LBL_UPC_D1 LBL_UPC_D2 LBL_UPC_D3 LBL_UPC_D4 LBL_UPC_D5 LBL_UPC_E LBL__UPCE_2 LBL__UPCE_5 LBL__ITF LBL_UNKN_DEFAULT
LableData	The data read from the scanner.

Remarks
Call this method to send scanner data to the 4690 sales application.

7.4.23 SetFLIData Method

Syntax
void SetFLIData(LONG MaxLen, LONG EchoInput, LONG FirstDisplayCol, LONG LastDisplayCol, LONG StateNum, BOOL AlphaEntryMode, LONG TransitionFlag);

Parameter	Description
MaxLen	Maximum length of the data to be returned to the application.
EchoInput	Says whether data should be echoed to the screen. Can be one of the following values: KBD_DISP, KBD_DISP_EDIT or KBD_NO_DISP.
FirstDisplayCol	First display column for keyed data.
LastDisplayCol	Last display column for keyed data.
StateNum	Current input state number.
AlphaEntryMode	TRUE if entering alpha mode or FALSE if not.
TransitionFlag	Tells the client the state transition remained the same or was caused by the application or that the transition was caused by the state table.

Remarks
Call this method to set up the field level input data for the 4690 sales application.

7.4.24 SetInstanceData Method

Syntax
void SetInstanceData(LONG Instance, LONG UnitID, INT RPAMDevice);

Parameter	Description
Instance	Instance

UnitID	Unit ID
RPAMDevice	IOPROC

7.4.25 StateNum Property

Syntax
LONG StateNum;

Remarks
The current state number of the 4690 sales application.

7.4.26 TransitionFlag Property

Syntax
LONG TransitionFlag;

Remarks
This property tells the RPAM client the reason for the state transition that just happened. Either the state remained the same or the application caused the change or the state table caused the transition.

Chapter 8. POSKeyboard Object (multiple dispatch mode)

The POSKeyboard object represents the keyboard device.

8.1 Properties

ConfiguredID
DeviceID
KeylockPosition
OfflineLight
MessageLight
WaitLight

8.2 Methods

SendKeyboardData
SendKeyboardScanCode
SetInstanceData

8.3 Events

Do_FireLightEvent
Do_Fire_ToneEvent

8.3.1 ConfiguredID Property

Syntax
LONG ConfiguredID;

Remarks
The ConfiguredID property contains the 4690 POS device id for this device as loaded from the terminal device configuration record for this object's terminal number.

8.3.2 DeviceID Property

Syntax
LONG DeviceID;

Remarks
The DeviceID property contains the 4690 POS device id for this device.

8.3.3 Do_ FireLightEvent Event

Syntax
HRESULT Do_FireLightEvent(LONG LightNum, BOOL NewValue);

Remarks
Send the light status to the 4690 sales application..

8.3.4 Do_Fire_ToneEvent Event

Syntax
HRESULT Do_Fire_ToneEvent(LONG Frequency, LONG Duration);

Parameter	Description
Frequency	The frequency to issue the tone sound.
Duration	The duration of the tone sound. 0 = turn tone off

-1 = turn tone on
x = dureation in milliseconds

Remarks
Send the light status to the 4690 sales application..

8.3.5 KeyLockPosition Property

Syntax
LONG KeylockPosition;

Remarks
The keylock position of the keylock on the keyboard device.

8.3.6 OfflineLight Property

Syntax
LONG OfflineLight;

Remarks
Status of the offline light on the keyboard device.

8.3.7 MessageLight Property

Syntax
LONG MessageLight;

Remarks
Status of the message light on the keyboard device.

8.3.8 WaitLight Property

Syntax
LONG WaitLight;

Remarks
Status of the wait light on the keyboard device.

8.3.9 SendKeyboardData Method

Syntax
void SendKeyboardData(LONG FunctionCode);

Parameter	Description	
FunctionCode	Number identifying the keyboard function code you wish to send to the application. Function codes above 60 are application defined.	
	1	S1 special code
	2	S2 special code
	3	Double 0 special code
	4	Triple 0 special code
	48-57 (x30 - x39)	Numeric keypad codes 0 - 9
	61-255 (x3D - xFF)	4690 keyboard function codes

Remarks
Call this method to send keyboard function code data to the 4690 sales application.

8.3.10 SendKeyboardScanCode Method

Syntax
void SendKeyboardScanCode(LONG ScanCode);

Remarks
Call this method to send keyboard scan codes to the 4690 sales application.

ScanCode must be one of the codes defined in the IBM Point of Sale Subsystem: Installation, Keyboards and Code Pages Book.

8.3.11 SetInstanceData Method

Syntax
void SetInstanceData(LONG Instance, LONG UnitID, INT RPAMDevice);

Parameter	Description
Instance	Instance
UnitID	Unit ID
RPAMDevice	Keyboard device.

Chapter 9. POSMSR Object (multiple dispatch mode)

The POSMSR (Mag Stripe Reader) object represents a virtual magnetic stripe reader device to the 4690 sales application. The magnetic stripe reader is an input device that reads encoded data on a credit or debit card. The application will read data from this device and issue lock and unlock commands to this device. You can use this object to inform the application about the capabilities of the MSR device attached to the physical terminal and to provide MSR data to the application

9.1 Properties

ConfiguredID
DeviceID
IsLocked

9.2 Methods

SendMSRDataAscii
SetInstanceData

9.3 Events

DoFireLockMSR
DoFireUnlockMSR

9.4 POSMSR Object Reference

9.4.1 ConfiguredID Property

Syntax
LONG ConfiguredID;

Remarks
The ConfiguredID property contains the 4690 POS device id for this device as loaded from the terminal device configuration record for this object's terminal number. The RPAM interface layer will set this property based on the store controller configuration information and it is provided to the user for informational purposes only.

If you are using automatic device configuration the ConfiguredID and DeviceID will have the same value. If you're using manual device configuration it is your responsibility to ensure that the value you choose for DeviceID is compatible with the application's requirements. Read-only LONG.

9.4.2 DeviceID Property

Syntax
LONG DeviceID;

Remarks
The DeviceID property contains the 4690 POS device id for this device. If you are using automatic terminal device configuration, this property is read-only as the RPAM interface layer will manage its value based on the 4690 terminal device configuration.

If you are using manual device configuration and you intend to emulate this device, it's your responsibility to set this property to a valid POS cash drawer device ID device before calling Connect for the terminal.

9.4.3 DoFireLockMSR

Syntax
void DoFireLockMSR( );

Remarks
Occurs when the 4690 sales application issues a lock command to the MSR device.

9.4.4 DoFireUnlockMSR

Syntax
void DoFireUnlockMSR ( );

Remarks
Occurs when the 4690 sales application issues an unlock command to MSR device.

9.4.5 IsLocked Property

Syntax
BOOL IsLocked;

Remarks
True if the 4690 sales application has locked the MSR device, otherwise false. Read-only Boolean. Default value: TRUE.

9.4.6 SendMSRDataAscii Method

Syntax
void SendMsrData(BSTR Track1Data,
BSTR Track2Data,
BSTR Track3Data);

Parameter	Description
Track1Data	Data read by the MSR from track 1 of the card.
Track2Data	Data read by the MSR from track 2 of the card.
Track3Data	Data read by the MSR from track 3 of the card.

Remarks
Call this method to send MSR data to the 4690 sales application. The BSTR for a track should have a zero length if there is no data available for the track from the device (device is not capable of reading a particular track, the card does not have the track or there was a read error on the track).

9.4.7 SetInstanceData Method

Syntax
void SetInstanceData(LONG Instance, LONG UnitID, INT RPAMDevice);

Parameter	Description
Instance	Instance
UnitID	Unit ID
RPAMDevice	MSR

Chapter 10. POSDisplay Object (multiple dispatch mode)

The POSDisplay object represents a virtual 2x20 POS display to the terminal sales application. The display is comprised of two rows numbered 1-2 from top to bottom and 20 columns per row numbered 1-20 from left to right. The application uses this device by issuing cursor locate commands to the device to control where the next output will be displayed, by issuing writes to the device to display data on the display and by issuing reads to the device to determine what is on the display. You can use this object to be informed when the application writes data to one of the POS ANDisplay devices.

10.1 Properties

ConfiguredID
DeviceID
Row1Text
Row2Text

10.2 Events

DisplayEvent

10.3 POSDisplay Object Reference

10.3.1 ConfiguredID Property

Syntax
LONG ConfiguredID;

Remarks
The ConfiguredID property contains the 4690 POS device id for this device as loaded from the terminal device configuration record for this object's terminal number. The RPAM interface layer will set this property based on the store controller configuration information and it is provided to the user for informational purposes only.

If you are using automatic device configuration the ConfiguredID and DeviceID will have the same value. If you're using manual device configuration it is your responsibility to ensure that the value you choose for DeviceID is compatible with the application's requirements. Read-only LONG.

10.3.2 DeviceID Property

Syntax
LONG DeviceID;

Remarks
The DeviceID property contains the 4690 POS device id for this device. If you are using automatic terminal device configuration, this property is read-only as the RPAM interface layer will manage its value based on the 4690 terminal device configuration.

If you are using manual device configuration and you intend to emulate this device, it's your responsibility to set this property to a valid POS cash drawer device ID device before calling Connect for the terminal.

10.3.3 DisplayEvent

Syntax
void DisplayEvent(BSTR Row1Text, BSTR Row2Text);

Parameter	Description
Row1Text	The text currently displayed on row 1 of the POS display.

Row2Text	The text currently displayed on row 2 of the POS display.

Remarks
Occurs when the 4690 sales application writes to the POS display. The length of these parameters is always 20 characters long.

10.3.4 Row1Text Property

Syntax
BSTR Row1Text;

Remarks
Contains the text currently displayed on the first row of the POS display. Read-only BSTR.

10.3.5 Row2Text Property

Syntax
BSTR Row2Text;

Remarks
Contains the text currently displayed on the second row of the POS display. Read-only BSTR.

Chapter 11. POSFile Object (multiple dispatch mode)

The POSFile object represents a 4690 OS non-keyed file or pipe. You can use the POSFile object to manipulate 4690 OS non-keyed files stored locally on the POS terminal or remotely on the store controller. You can also use the POSFile object to create and use 4690 pipes.

11.1 Properties

ErrorCode
Name

11.2 Methods

Close
Create
Delete
Open
Read
Rename
Write

11.3 POSFile Object Reference

11.3.1 Close Method

Syntax
void Close();

Remarks
Closes the OS file. Error code is set to 0 on success, otherwise –1.

11.3.2 Create Method

Syntax
void Create(LONG Flags, LONG FileSize, LONG RecSize, BOOL Priority);

Parameter	Description
Flags	Combination of 1 or more of the following: rpaFlgDelete — Use if the application can delete the file before closing it. rpaFlgWrite — Open file for write rpaFlgRead — Open file for read. rpaFlgShare — Open file shared. rpaFlgShareRO — Allow shared reads only, disallow writes.
FileSize	Initial size of the file in bytes. Set to 0 for files whose size is dynamic.
RecSize	This is used by the read, write and record lock routines to ensure that the requested action falls on a record boundary. Use 0 or 1 to prevent boundary checking.
Priority	Set to TRUE to give file accesses for this file priority over other pending disk requests.

Remarks

Use the Create method to create a non-keyed POS file on the store controller and allocate disk space for it. If the file exists it will be erased before the new file is created.

11.3.3 Delete Method

Syntax
void Delete();

Remarks
Deletes the OS file or pipe. Error code is set to 0 on success, otherwise –1.

11.3.4 ErrorCode Property

Syntax
LONG ErrorCode;

Remarks
Holds the 4690 OS error code returned from the last operation performed on this file object.

11.3.5 Name Property

Syntax
BSTR Name;

Remarks
Holds the name or logical name of the file or pipe. The name must include the path specification when necessary. Read/write BSTR.

11.3.6 Open Method

Syntax
void Open(LONG Flags, BOOL Priority);

Parameter	Description	
Flags	Combination of 1 or more of the following:	
	rpaFlgDelete	Use if the application can delete the file before closing it.
	rpaFlgWrite	Open file for write
	rpaFlgRead	Open file for read.
	rpaFlgShare	Open file shared.
	rpaFlgShareRO	Allow shared reads only, disallow writes.

PrioritySet to TRUE to give file accesses for this file priority over other pending disk requests.

Remarks
Use the Open method to open an existing 4690 pipe or non-keyed POS file on the store controller. The file can be opened for reading, writing or both.

11.3.7 Read Method

Syntax
void Read(LONG Origin, VOID *Buf, LONG BufSize, LONG Offset);

Parameter	Description
Origin	Determines how Offset parameter is interpreted. Can be either rpaSeekBOF or rpaSeekEOF.
Buf	Address of the buffer where data from the read operation will be stored.
BufSize	Number of bytes to read.

Offset	Position to begin reading relative to the position indicated by Origin..

Remarks
Use the Read method to read data from an existing non-keyed POS file on the store controller or from a terminal pipe.

11.3.8 Rename Method

Syntax
void Rename(BSTR NewName);

Parameter	Description
NewName	The new file name.

Remarks
Use the Rename method to rename an existing non-keyed POS file on the store controller. The new name must include the path specification when necessary.

11.3.9 Write Method

Syntax
void Write (LONG Origin, VOID *Buf, LONG BufSize, LONG Offset, BOOL Flush);

Parameter	Description
Origin	Determines how Offset parameter is interpreted. Can be either rpaSeekBOF or rpaSeekEOF.
Buf	Address of the buffer where data to be written is stored.
BufSize	Number of bytes to write.
Offset	Position to begin writing relative to the position indicated by Origin.

Remarks
Use the Write method to write data to an existing non-keyed POS file on the store controller or to a terminal pipe.

Chapter 12. POSKeyedFile Object (multiple dispatch mode)

The POSKeyedFile object represents a 4690 OS keyed file. A keyed file is a file formatted into logical records and accessed using a key related to the data in the file. You can use the POSKeyedFile object to manipulate 4690 OS keyed files stored locally on the POS terminal or remotely on the store controller.

12.1 Properties

ErrorCode
Name

12.2 Methods

Close
DeleteRecord
Open
Read
Rename
Write

12.3 POSKeyedFile Object Reference

12.3.1 Close Method

Syntax
void Close();

Remarks
Closes the OS file. Error code is set to 0 on success, otherwise –1.

12.3.2 DeleteRecord Method

Syntax
void DeleteRecord(VOID *Buf, LONG KeyLen);

Parameter	Description
Buf	Address of the buffer containing the key value of the record that is to be deleted.
KeyLen	Length of the key string in Buf.

Remarks
Use the Read method to read a record from an existing keyed POS file on the store controller.

12.3.3 ErrorCode Property

Syntax
LONG ErrorCode;

Remarks
Holds the 4690 OS error code returned from the last operation performed on this file object.

12.3.4 Name Property

Syntax
BSTR Name;

Remarks
Holds the name of the file. The name must include the path specification when necessary. Read/write BSTR.

12.3.5 Open Method

Syntax
void Open(LONG Flags, BOOL Priority);

Parameter	Description	
Flags	Combination of 1 or more of the following:	
	rpaFlgDelete	Use if the application can delete the file before closing it.
	rpaFlgWrite	Open file for write
	rpaFlgRead	Open file for read.
	rpaFlgShare	Open file shared.
	rpaFlgShareRO	Allow shared reads only, disallow writes.
Priority	Set to TRUE to give file accesses for this file priority over other pending disk requests.	

Remarks
Use the Open method to open an existing keyed POS file on the store controller. The file can be opened for reading, writing or both.

12.3.6 Read Method

Syntax
void Read(BOOL Lock, VOID *Buf, LONG BufSize, LONG KeyLen);

Parameter	Description
Lock	Lock the record before reading it when set to TRUE.
Buf	Address of the buffer where data from the read operation will be stored.
BufSize	Number of bytes to read, must match record length of the underlying file.
KeyLen	Length of the key string in Buf.

Remarks
Use the Read method to read a record from an existing keyed POS file on the store controller.

12.3.7 Rename Method

Syntax
void Rename(BSTR NewName);

Parameter	Description
New	NameThe new file name.

Remarks
Use the Rename method to rename an existing keyed POS file on the store controller. The new name must include the path specification when necessary.

12.3.8 Write Method

Syntax
void Write(BOOL Lock, VOID *Buf, LONG BufSize);

Parameter	Description

Lock	Lock the record before writing it when set to TRUE.
Buf	Address of the buffer that contains the data to be written to the file.
BufSize	Number of bytes to write, must match record length of the underlying file.

Remarks

Use the Write method to write records to an existing keyed POS file on the store controller.

Chapter 13. POSPrinter Object (multiple dispatch mode)

The POSPrinter object represents a virtual POS printer device to the 4690 sales application. Version 1.0 of the RPAMTerminal object and the current TC/NT version only support for 'remoting' the IBM Model 2 printer.

The RPAM interface layer will always respond back to the application with a good return code for each print (DeviceResponseMode is always set to rpaAutoDeviceResponse). Currently there is no way for the client application to notify the sales application regarding printer status or printer errors.

13.1 Properties

BufferedPrintCount
ConfiguredID
CurrentCRPrints
CurrentCRPrintCount
CurrentDIPrints
CurrentDIPrintCount
CurrentSJPrints
CurrentSJPrintCount
DeviceID
EnablePrintBuffer

13.2 Events

PrintEvent

13.3 POSPrinter Object Reference

13.3.1 BufferedPrintCount

Syntax
LONG BufferdPrintCount;

Remarks
When EnablePrintBuffer is set to TRUE this property controls how many prints are buffered. Prints are purged from the buffer on a FIFO basis after the buffer becomes full. Default value: 25.

13.3.2 ConfiguredID Property

Syntax
LONG ConfiguredID;

Remarks
The ConfiguredID property contains the 4690 POS device id for this device as loaded from the terminal device configuration record for this object's terminal number. The RPAM interface layer will set this property based on the store controller configuration information and it is provided to the user for informational purposes only.
If you are using automatic device configuration the ConfiguredID and DeviceID will have the same value. If you're using manual device configuration it is your responsibility to ensure that the value you choose for DeviceID is compatible with the application's requirements. Read-only LONG.

13.3.3 CurrentCRPrints Property

Syntax
SAFEARRAY CurrentCRPrints;

Remarks
When EnablePrintBuffer is set to TRUE this property holds the accumulated prints issued by the 4690 sales application to the CR print station. If EnablePrintBuffer is set to FALSE this array is empty. Read-only SAFEARRAY.

13.3.4 CurrentCRPrintCount Property

Syntax
LONG CurrentCRPrintCount;

Remarks
Contains the number of prints contained in the CurrentCRPrints property array. Read-only LONG.

13.3.5 CurrentDIPrints Property

Syntax
SAFEARRAY CurrentDIPrints;

Remarks
When EnablePrintBuffer is set to TRUE this property holds the accumulated prints issued by the 4690 sales application to the DI print station. If EnablePrintBuffer is set to FALSE this array is empty. Read-only SAFEARRAY.

13.3.6 CurrentDIPrintCount Property

Syntax
LONG CurrentDIPrintCount;

Remarks
Contains the number of prints contained in the CurrentDIPrints property array. Read-only LONG.

13.3.7 CurrentSJPrints Property

Syntax
SAFEARRAY CurrentSJPrints;

Remarks
When EnablePrintBuffer is set to TRUE this property holds the accumulated prints issued by the 4690 sales application to the SJ print station. If EnablePrintBuffer is set to FALSE this array is empty. Read-only SAFEARRAY.

13.3.8 CurrentSJPrintCount Property

Syntax
LONG CurrentSJPrintCount;

Remarks
Contains the number of prints contained in the CurrentSJPrints property array. Read-only LONG.

13.3.9 DeviceID Property

Syntax
LONG DeviceID;

Remarks
The DeviceID property contains the 4690 POS device id for this device. If you are using automatic terminal device configuration, this property is read-only as the RPAM interface layer will manage its value based on the 4690 terminal device configuration.

If you are using manual device configuration and you intend to emulate this device, it's your responsibility to set this property to a valid POS printer device ID device before calling Connect for the terminal.

Currently suppored device IDs for POS printers are:
rpaMod12prt (48) for Mod 1/2 POS printer
rpaMod34prt(52) for Mod 3/4 POS printer and
rpa4610prt(53) for 4610 POS printer

13.3.10 EnablePrintBuffer

Syntax
BOOL EnablePrintBuffer;

Remarks
The POSPrinter device has the capability to retain prints in internal buffers for examination at a later time by the client application. Setting this property to TRUE enables this behavior.

13.3.11 PrintEvent

Syntax
void PrintEvent(LONG Station, BSTR PrintLine, LONG LineFeeds);

Parameter	Description
Station	rpaCRStation for the prints to the cash receipt station, rpaSJStation for prints to the store journal station, rpaDIStation for prints to the document insert station.
PrintLine	The data to be printed.
LineFeeds	The number of line feeds to issue after printing the data.

Remarks
Occurs when the 4690 sales application prints data to one of the print stations on the POS printer device.

Chapter 14. RPATermControl (multiple dispatch mode)

The RPATermControl object represents a special RPA defined ‘pseudo’ device that allows the 4690 sales application developer to communicate with the RPAMTerminal object (and it’s client application) through special writes to the 4690 totals retention driver

For example, the sales application might be enhanced to send a subtotal amount to a remote GUI application following each rung item. To facilitate this capability the sales application would be enhanced to issue a direct write to Offset 1 of the hard totals area with the first 4 bytes of data containing the values 0x7F 0xFF 0x53 0x43. The 7FFF5343 is a special data sentinel that tells the TC totals retention driver that the data is not totals data but is instead a "control" message to be sent to the remote RPAM terminal. A maximum of x3C bytes can be passed including the 4 bytes of sentinel data.

The first byte after the sentinal indicates the type of message bine sent to the RPAM terminal. This value has meaning to the RPAM interface layer. The following values have been impletmented in the RPAM interface layer and are included here for completeness.

Value	Description
D	Reserved for system. Causes RPAM to dump internal trace data to the controller.
H	Reserved for system (system heartbeat)
I	Rreserved for system (Terminal IPL control)
M	Marquee (Times Square) display control
O	Offline application data
P	Print control
R	Reserved for system (terminal reload command)
T	Reserved for system (terminal date time)
U	User data

The only value that is is currently useful in the generat case is U. You can use this value to pass data directly to a RPAM client.

14.1 Events

ActivateMarqueeDisplayEvent
DeActivateMarqueeDisplayEvent
LoadMarqueeDisplayEvent
OfflineTotalsDataEvent
PrintEjectEvent
ReceiptCutEvent
TermIPLEvent
TermReloadEvent
UserDataEvent

14.1.1 ActivateMarqueeDisplayEvent

Syntax

void ActivateMarqueeDisplayEvent(LONG StartCol, LONG EndCol, LONG Delay);

Parameter	Description
StartCol	Starting column for message display.
EndCol	Ending column for message display.
Delay	Milliseconds between writes.

Remarks

Occurs when the 4690 sales application issues the command to begin scrolling a previously loaded marquee display message.

14.1.2 DeactivateMarqueeDisplayEvent

Syntax
void DecctivateMarqueeDisplayEvent();

Remarks
Occurs when the 4690 sales application issues the command to stop scrolling a previously activated marquee display message.

14.1.3 LoadMarqueeDisplayEvent

Syntax
void LoadMarqeeDisplayEvent(LONG Segment, BSTR Data, LONG Length);

Parameter	Description
Segment	The application can load the marquee display message in up to three segments, each of which can be up to 40 characters long. This parameter is the segment being loaded.
Data	The text for the segment.
Length	Length of the text for the segment in characters.

Remarks
Occurs when the 4690 sales application issues the command to load a marquee display message.

14.1.4 OfflineTotalsDataEvent

Syntax
void OfflineTotalsDataEvent(VOID *Data, LONG DataLen);

Parameter	Description
Data	The offline totals data.
DataLen	Length of the data as written by the application.

Remarks
Occurs when the 4690 sales application issues the command to send offline totals data to the RPAM client device.

14.1.5 PrintEjectEvent

Syntax
void PintEjectEvent();

Remarks
Occurs when the 4690 sales application issues the command to eject the paper in the printer to the RPAM client device.

14.1.6 ReceiptCutEvent

Syntax
void ReceiptCutEvent();

Remarks
Occurs when the 4690 sales application issues the command to cut the receipt in the printer to the RPAM client device.

14.1.7 TermIPLControlEvent

Syntax
void TermIPLControlEvent(BOOL Enable);

Parameter	Description
Enable	The terminal should respect the request to reload.
Disable	The terminal should not respect the request to reload.

Remarks
Occurs when the 4690 sales application issues the command to control how the terminal should respond to the TermReloadEvent.

14.1.8 TermReloadEvent

Syntax
void TermReloadEvent();

Remarks
Occurs when the 4690 sales application issues the command to reload the terminal to the RPAM client device.

14.1.9 UserDataEvent

Syntax
void UserDataEvent(VOID *Data, LONG DataLen);

Parameter	Description
Data	Data written by the application.
DataLen	Length of the data written.

Remarks
Occurs when the 4690 sales application issues a user defined message to the RPAM client device.

Chapter 15. POSScale Object (multiple dispatch mode)

The POSScale object represents the scale device.

15.1 Properties

ConfiguredID
DeviceID
Enabled

15.2 Methods

Do_Fire_ScaleRead
SendScaleData
SetInstanceData

15.2.1 ConfiguredID Property

Syntax
LONG ConfiguredID;

Remarks
The ConfiguredID property contains the 4690 POS device id for this device as loaded from the terminal device configuration record for this object's terminal number.

15.2.2 DeviceID Property

Syntax
LONG DeviceID;

Remarks
The DeviceID property contains the 4690 POS device id for this device.

15.2.3 Enabled Property

Syntax
LONG Enabled;

Remarks
Contains whether or not the scale device is enabled.

15.2.4 Do_Fire_ScaleRead Method

Syntax
HRESULTS Do_Fire_ScaleRead ( );

Remarks
Occurs when the 4690 sales application issues a scale read.

15.2.5 SendScaleData Method

Syntax
HRESULTS SendScaleData(LONG Weight);

Parameter	Description
Weight	Weight value.

Remarks
Use this method when you want to send a scale weight to the 4690 sales application.

15.2.6 SetInstanceData Method

Syntax

HRESULT SetInstanceData(LONG Instance, LONG UnitID, INT RPAMDevice);

Parameter	Description
Instance	Instance
UnitID	Unit ID
RPAMDevice	Scale

Chapter 16. POSScanner Object (multiple dispatch mode)

The POSScanner object represents the scanner device.

16.1 Properties

ConfiguredID
DeviceID
ScannerLocked

16.2 Methods

SendScannerData
SetInstanceData

16.2.1 ConfiguredID Property

Syntax
LONG ConfiguredID;

Remarks
The ConfiguredID property contains the 4690 POS device id for this device as loaded from the terminal device configuration record for this object's terminal number.

16.2.2 DeviceID Property

Syntax
LONG DeviceID;

Remarks
The DeviceID property contains the 4690 POS device id for this device.

16.2.3 ScannerLocked Property

Syntax
LONG ScannerLocked;

Remarks
Contains whether or not the scanner device is locked.

16.2.4 SendScannerData Method

Syntax
HRESULTS SendScannerData(LONG LabelFormat, BSTR LabelData);

Parameter	Description
LabelFormat	Label format.
LabelData	Data from the label scanned.

Remarks
Use this method when you want to send a scanner information to the 4690 sales application.

16.2.5 SetInstanceData Method

Syntax
HRESULT SetInstanceData(LONG Instance, LONG UnitID, INT RPAMDevice);

Parameter	Description

Instance	Instance
UnitID	Unit ID
RPAMDevice	Scale

Chapter 17. POSVDisplay Object (multiple dispatch mode)

The POSVDisplay object represents a virtual video display to the 4690 sales application. There are four types of displays that are supported: 25x80, 16x60, 12x40 and 6x20. The application uses this device to locate the cursor on the screen, output data to the screen and read what data is visible on the screen. The application can also set character attributes and character color for the data to be written to the screen next. You can use this object to respond to the application's requests to set video display attributes and colors and to respond to requests to write data to the screen.

17.1 Properties

ConfiguredID
DeviceID
VideoMode

17.2 Events

DisplayEvent

17.3 POSVDisplay Object Reference

17.3.1 ConfiguredID Property

Syntax
LONG ConfiguredID;

Remarks
The ConfiguredID property contains the 4690 POS device id for this device as loaded from the terminal device configuration record for this object's terminal number. The RPAM interface layer will set this property based on the store controller configuration information and it is provided to the user for informational purposes only.

If you are using automatic device configuration the ConfiguredID and DeviceID will have the same value. If you're using manual device configuration it is your responsibility to ensure that the value you choose for DeviceID is compatible with the application's requirements. Read-only LONG.

17.3.2 DeviceID Property

Syntax
LONG DeviceID;

Remarks
The DeviceID property contains the 4690 POS device id for this device. If you are using automatic terminal device configuration, this property is read-only as the RPAM interface layer will manage its value based on the 4690 terminal device configuration.

If you are using manual device configuration and you intend to emulate this device, it's your responsibility to set this property to a valid POS cash drawer device ID device before calling Connect for the terminal.

17.3.3 DisplayEvent

Syntax
void DisplayEvent(BSTR Text, LONG Len, LONG Row, LONG Col);

Parameter	Description

Text	The text to be written to the video display.
Len	The length of the data to be written.
Row	The starting row for the write operation.
Col	The starting column for the write operation.

Remarks
Occurs when the 4690 sales application writes to the video display device.

17.3.4 VideoMode Property

Syntax
LONG VideoMode;

Remarks
Informs the 4690 sales application what the current video mode is. Can be one of the following values: rpaVDisp2580, rpaVDisp1660, rpaVDisp1240 or rpaVDisp0620. Read/write Long. Default value: rpaVDisp2580.

Chapter 18. POSPrsReadPipe Object (multiple dispatch mode)

Support for the creation of PRS pipes which can be read by RPAM client applications is provided through the POSPrsReadPipe object. When you want to read data from a PRS pipe you use the POSPrsReadPipe object to create the PRS pipe and then issues reads to the pipe.

Note: Sample C++ programs demonstrating the use of PRS pipe objects in the RPAM client library can be found in the 'pipesamp' subdirectory of the 'Samples' directory where you installed Terminal Concentrator (if you chose to install the sample programs when you installed TC).

18.1 Properties

BuffSize
ErrorCode4690
PipeID
UnitID

18.2 Methods

Close
Create
Read
Read2

18.3 POSPrsReadPipe Object Reference

18.3.1 BuffSize Property

Syntax
long BuffSize;

Remarks
The number of bytes the operating system should allocate for the pipe's buffer. Maximum buffer size for pipes created by terminal applications is 240 bytes. See the 4690 Programming Guide for more information on this property.

18.3.2 Close Method

Syntax
HRESULT Close();

Return Value

Returns 0 if successful otherwise wrn4690Error (0x00046001) if a 4690 error occurred closing the pipe. If a 4690 error occurred the ErrorCode4690 property will hold the 4690 return code from the operation.

Remarks
Closes the PRS pipe. The pipe must have successfully be created for the Close() to succeed.

18.3.3 Create Method

Syntax
HRESULT Create();

Return Value

Returns 0 if successful otherwise wrn4690Error (0x00046001) if a 4690 error occurred creating the pipe. If a 4690 error occurred the ErrorCode4690 property will hold the 4690 return code from the operation.

Remarks
Use the Create method to create a PRS pipe that other applications can use to write data to your RPAM client application.
Before calling create you must set the UnitID and PipeID properties of the POSPrsReadPipe object. UnitID must be set the UnitID of an already connected RPAM Terminal object and PipeID must be set to character between 'A' and 'Z'.

18.3.4 ErrorCode4690 Property

Syntax
long ErrorCode4690;

Remarks
This property holds the 4690 return code value from the last operation on this object. If no 4690 error occurred performing the operation this property is set to zero. If any method returns the value wrn4690Error, you can query this property to determine what 4690 error was returned. Refer to the 4690 Programming Guide and the Basic Language reference to interpret these return code values.

18.3.5 PipeID Property

Syntax
BSTR PipeID;

Remarks
Set this property to a single character between 'A' and 'Z'. This is the value that other applications which wish to write data to this pipe will specify when they issue their write PRS pipe calls. For example, if the UnitID property of a PRSReadPipe object was set to the UnitID of terminal 013 and the PipeID property was set to 'J', then other applications could write data to this pipe object by specifying "013J" as the destination in their calls to the PRS write functions.
Refer to the 4690 Programming Guide and the Basic Language reference for more information on the PipeID property of PRS pipes.

18.3.6 Read/Read2 Method

Syntax
HRESULT Read(SAFEARRAY *Buf, long BufSize, long TimeOut, long *BytesRead);
HRESULT Read2(VARIANT *Buf, long BufSize, long TimeOut, long *BytesRead);

Parameter	Description
Buf	For the Read method Buf is a SAFEARRAY of type VT_UI1 (unsigned character array). For the Read2 method Buf is a VARIANT whose parray member holds a pointer to a SAFEARRAY of type VT_UI1 (unsigned character array).
BufSize	Number of bytes to read from the pipe.
TimeOut	The length of time the method should wait for data to become available in the pipe in milliseconds.
BytesRead	Pointer to storage where the number of bytes actually read will be stored.

Return Value

Returns the number of bytes read from the pipe if data was available. If the read timed out the method returns –1 and ErrorCode4690 is set to errReadTimedOut (0x80046017). Otherwise wrn4690Error (0x00046001) if a 4690 error occurred reading from the pipe. If a 4690 error occurred the ErrorCode4690 property will hold the 4690 return code from the operation.

Remarks

Use the Read method to read data from a PRS pipe your application created. Read and Read2 are functionally equivalent. Read2 is provided to support development environments such as Visual Basic and Visual C++ applications utilizing MFC. These development environments have difficulty dealing with SAFEARRAYs of unsigned characters.

18.3.7 UnitID Property

Syntax

long UnitID;

Remarks

UnitID must be set to the same value as the UnitID property of an already connected RPAMTerminal object. 4690 requires that all PRS pipe activity occur within the context of a terminal which is online to the controller. Therefore you must associate the PRSReadPipe object with a terminal that is online to the controller using the UnitID property.

Chapter 19. POSPrsWritePipe Object (multiple dispatch mode)

Support for RPAM client application that want to write data to PRS pipes created by other terminal and controller applications is provided through the POSPrsWritePipe object.

Note: Sample C++ programs demonstrating the use of PRS pipe objects in the RPAM client library can be found in the 'pipesamp' subdirectory of the 'Samples' directory where you installed Terminal Concentrator (if you chose to install the sample programs when you installed TC).

19.1 Properties

DestNode
ErrorCode4690
PipeID
UnitID

19.2 Methods

Write
Write2

19.3 POSPrsWritePipe Object Reference

19.3.1 DestNode Property

Syntax
BSTR DestNode;

Remarks
This property holds the address of terminal or store controller to which to send. When sending data to a pipe created by a controller application you can set the DestNode to the controller ID of the controller ("0CC" – "0ZZ") or you can specify the master store controller using "0AA" or the controller to which the terminal is attached by specifying "0BB". Refer to the 4690 Programming Guide and the Basic Language reference for more information on the DestNode property of PRS pipes.

19.3.2 ErrorCode4690 Property

Syntax
long ErrorCode4690;

Remarks
This property holds the 4690 return code value from the last operation on this object. If no 4690 error occurred performing the operation this property is set to zero. If any method returns the value wrn4690Error, you can query this property to determine what 4690 error was returned. Refer to the 4690 Programming Guide and the Basic Language reference to interpret these return code values.

19.3.3 PipeID Property

Syntax
BSTR PipeID;

Remarks
Set this property to a single character between 'A' and 'Z'. This property should match the PipeID value of the pipe you are writing to. This property combined with the DestNode property determines which pipe

will receive the data you are trying to write. Refer to the 4690 Programming Guide and the Basic Language reference for more information on the PipeID and DestNode properties of PRS pipes.

19.3.4 UnitID Property

Syntax
long UnitID;

Remarks
UnitID must be set to the same value as the UnitID property of an already connected RPAMTerminal object. 4690 requires that all PRS pipe activity occur within the context of a terminal which is online to the controller. Therefore you must associate the PRSReadPipe object with a terminal that is online to the controller using the UnitID property.

19.3.5 Write/Write2 Method

Syntax
HRESULT Write(SAFEARRAY *Buf, long BufSize);
HRESULT Read2(VARIANT *Buf, long BufSize);

Parameter	Description
Buf	For the Write method Buf is a SAFEARRAY of type VT_UI1 (unsigned character array). For the Read2 method Buf is a VARIANT whose parray member holds a pointer to a SAFEARRAY of type VT_UI1 (unsigned character array).
BufSize	Number of bytes to write to the pipe.

Return Value

Returns the 0 if the method succeeded. Otherwise wrn4690Error (0x00046001) if a 4690 error occurred writing to the pipe. If a 4690 error occurred the ErrorCode4690 property will hold the 4690 return code from the operation.

Remarks
Use the Write method to write data to a PRS pipe created by another application. Write and Write2 are functionally equivalent. Write2 is provided to support development environments such as Visual Basic and Visual C++ applications utilizing MFC. These development environments have difficulty dealing with SAFEARRAYs of unsigned characters.
Before writing to a pipe you must set the UnitID, PipeID and DestNode properties.

Chapter 20. POSSerialDevice Object (multiple dispatch mode)

Support for 4690 sales applications which communicate with devices via the serial communications ports is provided through the POSSerialDevice object.

Note: Sample C++ programs demonstrating the use of the POSSerialDevice object in the RPAM client library can be found in the 'sersamp' subdirectory of the 'Samples' directory where you installed Terminal Concentrator (if you chose to install the sample programs when you installed TC).

20.1 Properties

ConfiguredID
DeviceEnabled
DeviceID

20.2 Methods

SendOpenRc
SendPutLongRc
SendWriteRc
UpdatePortStatus
Write

20.3 Events

CloseEvent
OpenEvent
PutLongEvent
WriteEvent

20.4 POSSerialDevice Object Reference

20.4.1 Close Event

Syntax
HRESULT CloseEvent();

Remarks
This event is fired when the 4690 application issues a close to the serial port. Your application should take the appropriate action to close the physical or virtual communications port when you receive this event. There is no response or return code required when a Close event is received.

20.4.2 ConfiguredID Property

Syntax
LONG ConfiguredID;

Remarks
The ConfiguredID property contains the 4690 POS device id for this device as loaded from the terminal device configuration record for this object's terminal number. The RPAM interface layer will set this property based on the store controller configuration information and it is provided to the user for informational purposes only.

If you are using automatic device configuration the ConfiguredID and DeviceID will have the same value. If you're using manual device configuration it is your responsibility to ensure that the value you choose for

DeviceID is compatible with the application's requirements. Read-only LONG.

20.4.3 DeviceEnabled Property

Syntax
LONG DeviceEnabled;

Remarks
By default, the RPAMTerminal object reports that there are no serial communications ports available to the 4690 sales application. This is done because many RPAM client application s are not interested in supporting serially attached devices (Pinpad, MICR, etc). By reporting to the application that there are no serial ports, it relieves the RPAM client application of having to emulate the device that the sales application thinks is connected to the port.

If you do want to support a serially attached device (either physical or virtual) you need to set the DeviceEnabled property of the Serial1 and/or Serial2 POSSerialDevice objects of the RPAM terminal. You must set this property to 1 after you have set the UnitID property of the RPAM terminal object but before you issue the Connect() to connect the terminal to TC.

20.4.4 DeviceID Property

Syntax
LONG DeviceID;

Remarks
The DeviceID property contains the 4690 POS device id for this device. If you are using automatic terminal device configuration, this property is read-only as the RPAM interface layer will manage its value based on the 4690 terminal device configuration.

If you are using manual device configuration and you intend to emulate this device, it's your responsibility to set this property to a valid POS cash drawer device ID device before calling Connect for the terminal.

20.4.5 OpenEvent Event

Syntax
HRESULT OpenEvent(long Speed, BSTR Parity, long DataBits, long StopBits, long BuffSize);

Parameter	Description
Speed	Transmit/receive bit rate in bits per second.
Parity	Transmit/receive parity. "E" for event, "O" for odd, "N" for none.
DataBits	Number of transmit/receive data bits.
StopBits	Number of stop bits.
BuffSize	Size of the I/O buffer in bytes.

Remarks
This event is fired when the 4690 application issues an open to the serial port. Your application should take the appropriate action to open the physical or virtual communications port when you receive this event. Once your application has opened the port you are responsible for calling the SendOpenRc method of the POSSerialDevice to inform the 4690 application of the success or failure of the OpenEvent.

Refer to the 4690 Basic Language reference for more information on the parameters to this event

20.4.6 PutLongEvent Event

Syntax
HRESULT PutLongEvent(long TimeOut, long PortStatusValue);

Parameter	Description
TimeOut	The timeout value in milliseconds.
PortStatusValue	The port status information which the 4690 sales application wrote with the PUTLONG command it issued. This value corresponds to the CC byte of the PUTLONG value. See the 4690 Basic Language reference for more information on the bit fields that make up the CC byte.

Remarks
This event is fired when the 4690 application issues a PUTLONG to the serial port. Your application should take the appropriate action to configure the physical or virtual communications port when you receive this event. Once your application has configured the port you are responsible for calling the SendPutLongRc method of the POSSerialDevice to inform the 4690 application of the success or failure of the PutLongEvent.

20.4.7 SendOpenRc Method

Syntax
HRESULT SendOpenRc(long Rc);

Parameter	Description
Rc	The return code that will be reported to the 4690 application in response to the serial communications port OPEN that was issued.

Return Value

Returns the 0 if the method succeeded. This method will always succeed unless the RPAM client is offline from terminal concentrator.

Remarks
You must call this method in response to the receipt of an OpenEvent. When the 4690 sales application issues an Open to a communications port under TC, it is blocked until the RPAM client application responds to the Open request with a return code indicating the success or failure of the actual port open. You should send a 0 return code if your application opened the port successfully otherwise you should call this method with a value of 0x08980009 (this value corresponds to the 4690 return code for 'device not attached').

20.4.8 SendPutLongRc Method

Syntax
HRESULT SendPutLongRc(long Rc);

Parameter	Description
Rc	The return code that will be reported to the 4690 application in response to the serial communications port PUTLONG that was issued.

Return Value

Returns the 0 if the method succeeded. This method will always succeed unless the RPAM client is offline from terminal concentrator.

Remarks

You must call this method in response to the receipt of a PutLongEvent. When the 4690 sales application issues a PUTLONG to a communications port under TC, it is blocked until the RPAM client application responds to the request with a return code indicating the success or failure of the serial port set operation. You should send a 0 return code if your application responded successfully to the PUTLONG request otherwise you should call this method with a value of 0x08980009 (this value corresponds to the 4690 return code for 'device not attached').

20.4.9 SendWriteRc Method

Syntax
HRESULT SendWriteRc(long Rc);

Parameter	Description
Rc	The return code that will be reported to the 4690 application in response to the serial communications port WRITE that was issued.

Return Value

Returns the 0 if the method succeeded. This method will always succeed unless the RPAM client is offline from terminal concentrator.

Remarks
You should call this method in response to the receipt of a WriteEvent. The value of Rc should be set to the number of bytes you were able to write to the port. This is the value that the 4690 application will receive as its return code from the WRITE operation.

20.4.10 UpdatePortStatus Method

Syntax
HRESULT UpdatePortStatus(long Status);

Parameter	Description
Status	The value that will be used to construct the 4 byte port status information value reported to the 4690 sales application when it issues a GETLONG command to the serial port.

Return Value

Returns the 0 if the method succeeded. This method will always succeed unless the RPAM client is offline from terminal concentrator.

Remarks
The UpdatePortStatus method should be used by your application to inform the 4690 sales application that the status of the port(s) managed by your application has changed.

The value that you send to Terminal Concentrator using this method will be used by Terminal Concentrator to supply the application with a 4 byte value representing the status of the port the next time the sales application issues a GETLONG command for the port.

The naming convention used by RPAM follows the 4690 Basic Language Reference sections that describe the values returned by the GETLONG command for devices attached to 4684/4693/4694 serial port rather than to a 4383 feature C card. That is, the connected device is considered to be DCE equipment.

It is the responsibility of the RPAM client application to monitor the status of the port and call UpdatePortStatus whenever one of the following line status or modem status bits changes value.

Data Lost (8250 UART line status register bit 1)
DSR Active (8250 UART modem status register bit 5)
CTS Active (8250 UART modem status register bit 4)
Parity Error (8250 UART line status register bit 2)
Framing Error (8250 UART line status register bit 3)
Break Detected (8250 UART line status register bit 4)

The value of StatusValue can be a combination of the following values.

0x200 (DataLost)
0x02 (DSRActive)
0x04 (CTSActive)
0x08 (ParityError)
0x10 (FramingError)
0x20 (BreakDetected)

20.4.11 Write Method

Syntax
HRESULT Write(long Length, SAFEARRAY Data);

Parameter	Description
Length	The number of bytes of data you wish to send to the 4690 application.
Data	The data to be sent to the 4690 application Data is a SAFEARRAY of type VT_UI1 (unsigned character array). .

Return Value

Returns the 0 if the method succeeded. This method will always succeed unless the RPAM client is offline from terminal concentrator.

Remarks
Call this method to send serial data to the 4690 sales application. Whenever you have read data from the port you are managing you should call the Write method of the POSSerialDevice with Length set to the number of bytes contained in the Data paramater and Data should contain the actual data you read from the port