OLE for Retail POS

Control Programmer’s Guide

Release 1.6 July 15, 2001

International Standard

Windows 95/98/ME, Windows NT, Windows 2000, or other OLE/ActiveX compliant 32-bit operating system

OLE for Retail POS Committee

Core Companies
Epson
Fujitsu-ICL
Microsoft
NCR

plus
OPOS-Japan
OPOS-Europe

OLE for Retail POS

Control Programmer's Guide

Also see the following Web sites for OPOS information:

Microsoft Retail Industry Page:
http://www.microsoft.com/industry/retail/opos/home.asp

Reference implementation – Common Control Objects:
http://monroecs.com/opos.htm

NRF-ARTS Standards Body
http://www.nrf-arts.com

Table of Contents

INTRODUCTION

OLE for Retail POS Control Interface Overview

What Is “OLE for Retail POS?”

OLE for Retail POS provides an open device driver architecture that allows Point-of-Sale (“POS”)[1] hardware to be easily integrated into POS systems based on Microsoft Windows-95/98, Microsoft Windows-NT, and Microsoft Windows 2000.[2]

The goals of OLE for Retail POS (or “OPOS”) include:

- Defining an architecture for Win32-based POS device access.
- Defining a set of POS device interfaces sufficient to support a range of POS solutions.

Deliverables in this release of OPOS are:

- Application Programmer’s Guide: For application developers and hardware providers.
- Control Programmer’s Guide – this document: For hardware providers.
- Header files with OPOS constants.
- No complete software components: Hardware providers or third-party providers develop and distribute these components.

---

[1] POS may also refer to Point-of-Service – a somewhat broader category that Point-of-Sale.

[2] Other future operating systems that support OLE Controls may also support OLE for Retail POS, depending upon software support by the hardware manufacturers or third -party developers.

Who Should Read This Document

The Control Programmer's Guide is targeted for the system developer who will write an OPOS Control.

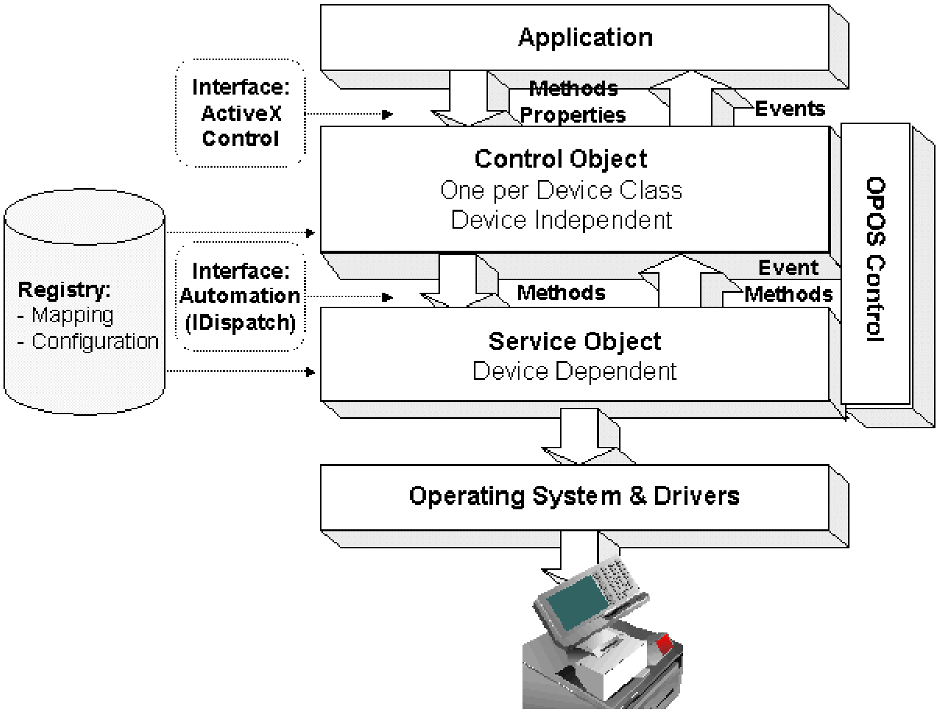
This guide assumes that the reader understands the following:

- The POS peripheral device to be supported.
- ActiveX Control and Automation terminology and architecture.
- ActiveX Control Container development environments, such as Microsoft Visual Basic or Microsoft Visual C++. These will be required for testing the OPOS Control.
- A thorough knowledge of the OPOS models and APIs presented in the Application Programmer's Guide. Little of the material from that guide is replicated in this guide.

General OLE for Retail POS Control Model

OLE for Retail POS Controls adhere to the ActiveX Control specifications. They expose properties, events, and methods to a containing application. They specifically do not include a user interface, but rather rely exclusively upon the containing application for requests through methods and sometimes properties. Responses are given to the application through method return values and parameters, events, and properties.

The OLE for Retail POS software is implemented using the layers shown in the following diagram:



OPOS Definitions

Device Class

A device class is a category of POS devices that share a consistent set of properties, methods, and events. Examples are Cash Drawer and POS Printer.

Some devices support more than one device class. For example, some POS Printers include a Cash Drawer kickout. Also, some Bar Code Scanners include an integrated Scale.

Control Object *or* CO

A Control Object exposes a set of properties, methods, and events to an application for its device class. The OPOS Application Programmer's Guide describes these APIs.

A CO is a standard ActiveX (that is, OLE 32-bit) Control that is invisible at runtime. The CO interfaces have been designed such that all implementations of a class' Control Object will be compatible. This allows the CO to be developed independently of the SO's for the same class – including development by different companies.

Service Object *or* SO

A Service Object is called by a Control Object to implement the OPOS-prescribed functionality for a specific device.

An SO is implemented as an Automation server. It exposes a set of methods that are called by a CO. It can also call special methods exposed by the CO to cause events to be fired to the application.

A Service Object may include multiple sets of methods in order to support devices with multiple device classes.

A Service Object is typically implemented as a local in-proc server (in a DLL). In theory, it may also be implemented as a local out-proc server (in a separate executable process). However, we have found that, in practice, out-proc servers do not work well for OPOS Service Objects, and do not recommend their use.

OPOS Control *or* Control

An OPOS Control consists of a Control Object for a device class – which provides the application interface, plus a Service Object – which implements the APIs. The Service Object must support a device of the Control Object's class.

Note - Service Object Implementation: Out-of-Process vs. In-Process Servers

In general, the primary difficulty in using out-proc automation servers arises when either of the two possible scenarios may occur:

(A) The server is processing a COM function for the client application (such as when the client has called a Control's method) when another server calls a COM function in the client (such as when a Control's event is fired).

(B) The server has called a COM function in a client application (such as when a Control's event is fired) when another client application calls a COM function in the server (such as when this client calls a Control's method).

The likelihood of these scenarios, especially (A), is greater for OPOS Service Objects since:

- Some OPOS methods require an indeterminately long time to be processed, such as the Cash Drawer **WaitForDrawerClose**.
- Some OPOS events may require an indeterminately long time to be processed, such as an **ErrorEvent** whose application handler waits for a user response to a dialog box.

The case where an OPOS event occurs from one service object while another service object is processing a method call or a property access then becomes probable.

These scenarios could be handled if both the client application and the out-proc server installed message filters (using the function **CoRegisterMessageFilter**), and the code for these filters dealt with these scenarios in an OPOS-prescribed manner. However, the default message filters for client environments such as Visual Basic and Visual C++ do not adequately handle the scenarios. Behavior varies from displaying a dialog and waiting for a user response (which is unacceptable for many POS operations) to generating an exception that terminates the client application (which is certainly unacceptable for POS applications). In addition, some environments do not provide a mechanism that easily allows an application to set up its custom message filter.

These issues simply do not exist when in-proc servers are used. Therefore, we recommend that they be used to implement service objects. This does, however, somewhat complicate sharing a Control between applications. Interprocess communication mechanisms, such as shared memory and named mutexes and events, may be used for this sharing.

If out-proc servers are used, then both the service object developer and the application developer will need to carefully implement message filters. The service object vendor should properly document this requirement to its application writers.

Interface Overview

A major OLE for Retail POS objective is to provide general peripheral device APIs that can be applied to many vendors' peripherals. This leads to a requirement that any implementation of a Control Object be able to communicate with any vendor's Service Object. A straightforward example is with printers: Suppose a fast-food restaurant requires a local printer by one vendor and a remote kitchen printer by another vendor. Two instances of the printer CO will be required where each instance communicates with a different SO. The single CO must work with both vendors' SOs.

In order to define Control Objects that work across many vendors' Service Objects, the Control Object interfaces should be as generic and simple as possible. Therefore, the CO will maintain very little information and will, in general, perform the following three duties:

- Service Object coupling: Supervises a dispatch interface with a Service Object for the device.
- Methods and properties: Performs a pass-through of the application's method and property requests to the Service Object.
- Events: When a Service Object calls one of the special event request methods in the Control Object, the CO fires an appropriate event to the application.

The various paths of communication between the application, Control Object, and Service Object are shown in the following sections.

Methods

An application initiates method calls to the OPOS Control.

Open Method

The **Open** method is processed as follows:

Application

1. App calls CO's **Open** method.

Control Object

2. CO calls SO's **OpenService** method.

Service Object

Close Method

The **Close** method is processed as follows:

Application

1. App calls CO's **Close** method.

Control Object

2. CO calls SO's **CloseService** method, if present; otherwise calls **Close** method.

Service Object

Other Methods

All other methods are processed as follows, where *Method* represents the name of the method:

Application

1. App calls CO's *Method* method.

Control Object

2. CO calls SO's *Method* method.

Service Object

Properties

An application initiates property accesses to the OPOS Control.

String Properties

Gets and sets of string properties are processed as follows, where *StringProp* represents the name of the property:

Application

1. App accesses CO's *StringProp* property.

Control Object

2. If get, CO calls SO's **GetPropertyString** method, with an index that represents *StringProp*.
   If set, CO calls SO's **SetPropertyString** method, with an index that represents *StringProp*.
   The index values are specified in header files. See Appendix C.

Service Object

LONG and BOOL Properties

Gets and sets of long and boolean properties are processed as follows, where *NumericProp* represents the name of the property:

Application

1. App accesses CO's *NumericProp* property.

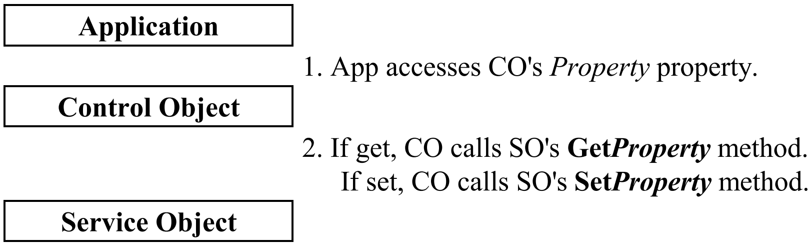
Control Object

2. If get, CO calls SO's **GetPropertyNumber** method, with an index that represents *NumericProp*.
   If set, CO calls SO's **SetPropertyNumber** method, with an index that represents *NumericProp*.
   The index values are specified in header files. See Appendix C.

Service Object

Other Property Types

Gets and sets of properties of any other type are processed as follows, where *Property* represents the name of the property:



1. App accesses CO's *Property* property.
2. If get, CO calls SO's **Get*Property*** method.
   If set, CO calls SO's **Set*Property*** method.

Events

See the Application Programmer's Guide, "Introduction" chapter, "Events" section for an overview of event handling.

The Service Object enqueues events, which are delivered to an application handler for the event.

The Service Object delivers events as follows:

Application

2. CO event request method delivers a Data, DirectIO, Error, OutputComplete, or StatusUpdate event to the App.

Control Object

1. SO calls a CO event request method. The methods **SOData**, **SODirectIO**, **SOError**, **SOOutputComplete**, and **SOStatusUpdate** are exposed specifically to cause events to be delivered to the App.

Service Object

Architecture: Firing an Event

A Service Object may need to fire an event for the following reasons:

- Method call or property set. A side effect of an application request to the control may cause an event to be fired.

  Example: Assume that some data has been read and enqueued by the SO. When the application changes the **DataEventEnabled** property to TRUE, then the SO needs to deliver a **DataEvent**.

- Asynchronous activity. The Service Object will usually create one or more worker threads to monitor the device's input or output. The SO cannot rely upon the application to call OPOS methods or access OPOS properties on a regular basis to gain some processing time, so independently scheduled worker threads are a good solution. These threads may determine that an event needs to be fired.

  Example: Assume that the **DataEventEnabled** property is TRUE, and that a worker thread is managing device input through a serial port. When the thread receives a data message, then the SO enqueues and needs to deliver a **DataEvent**.

When the SO needs to deliver an event, it calls a special event request method within the CO. The CO then delivers the event to the application.

Architectural Issue: Freezing Events by the Container

ActiveX control containers may freeze and unfreeze events by calling the **IOleControl::FreezeEvents** function. This is presented to a control written with MFC via the **COleControl::OnFreezeEvents** member function, or to an control written with ATL via the **IOleControlImpl::FreezeEvents** member function. (One use of this feature is by the Visual Basic Common Dialog functions, which freeze events while the dialog is up.) When events have been frozen, a control should not deliver events. The Visual C++ documentation notes that the control may either discard events that occur during the freeze period, or it may buffer them for later delivery.

For OPOS Controls, enqueued events must be retained but not delivered while the container has frozen them. Then, when events are unfrozen by the container, the events may be delivered.

Each Service Object must support the method **COFreezeEvents**. The Control Object will call this method to freeze and unfreeze events.

Architectural Feature: Freezing Events by the Application

The application may wish to disable the arrival of events for a period of time. They may do this by setting the common boolean property **FreezeEvents** to TRUE.

The event freezing mechanism implemented for container-requested freezing is utilized to remember requests while the application has frozen them. Then, when the application sets the property to FALSE to unfreeze events, the events are delivered.

Summary of Event Firing

When a Service Object needs to deliver an event, it calls the appropriate event request method within the Control Object.

However, if events have been frozen due to a Control Object call to **COFreezeEvents** or due to the application setting the **FreezeEvents** property to TRUE, then the SO keep the event on its event queue. If the event is to be delivered from a worker thread, then this typically will be implemented by blocking the thread until events are unfrozen.

CHAPTER 1

Control Object Responsibilities

The following sections describe the responsibilities of the Control Object. The Common Control Object is a reference implementation, whose source is available on the web.

Methods

The following sections describe the responsibilities of the Control Object methods.

If a device class does not support a common method (as specified by the device class Summary section in the “Application Programmer’s Guide”), then the Control Object should not define that method.

Since a Control Object must perform properly with any version of Service Object, as long as the major version numbers match, some bookkeeping must be performed in the Control Object. Specifically, the Control Object must not call methods that are not defined by a Service Object, or access properties that it does not define. In addition, it must perform additional management with the return values and **ResultCode**. (See the Application Programmer’s Guide, “Common Properties, Methods, and Events” chapter, “ControlObjectVersion” section for additional information.) The processing steps below assume that the Control Object defines a ResultCode flag to help manage version mismatch conditions.

Open Method

1. If the Control Object is already open, then set **OpenResult** to OPOS_OR_ALREADYOPEN return OPOS_E_ILLEGAL.
2. If an empty device name has been passed, then set **OpenResult** to OPOS_OR_REGBADNAME and return OPOS_E_NOEXIST.
3. Query the registry to find the Service Object that corresponds to this device class and device name. If the device class or device name is not found in the registry, then set **OpenResult** to OPOS_OR_REGBADNAME and return OPOS_E_NOEXIST.

4. Load the Service Object for the device name. This requires (a) reading the device's Programmatic ID from the registry, (b) converting it to a Class ID, (c) creating an instance of the Service Object, and (d) getting its IDispatch interface. If any of these are unsuccessful, then return OPOS_E_NOSERVICE. Set **OpenResult** to OPOS_OR_REGPROGID if (a) or (b) fails, or OPOS_OR_CREATE if (c) or (d) fails.

   MFC (a) Use **RegQueryValueEx**. (b) Use **CLSIDFromProgID**.
   (c)-(d) Calling the **CreateDispatch** member function of an instance of the Service Object class, passing the Class ID from (b).

   The Service Object class is generated by using the Visual C++ Class Wizard:

   - Within the "OLE Automation" tab, push the "Add Class from an OLE TypeLib..." button. Then choose the .TLB file generated by a Service Object project.
   - The Class Wizard will generate a **COleDispatchDriver** derivative, with member functions matching the OLE Automation methods exposed by the Service Object.

   The Class Wizard will also generate an implementation of the member functions, which call **InvokeHelper** with fixed dispatch IDs. Since dispatch IDs depend upon the definition order of the automation methods, this implementation must be updated by the next step to allow for Service Objects that define the methods in a different order.

   ATL (a) Use **RegQueryValueEx**. (b) Use **CLSIDFromProgID**.
   (c) Use **CoCreateInstance**. (d) Use **QueryInterface** on the interface pointer returned by (c).

5. .Look up the dispatch IDs for all of the Service Object methods defined by the device class.

   If any of the dispatch IDs defined in the initial version of the device class are not found in the Service Object, then close the dispatch interface, set **OpenResult** to OPOS_OR_BADIF, and return OPOS_E_NOSERVICE. (This ensures that the Service Object supports at least the minimum methods of a valid Service Object for the device class, before calling any of its methods.)

   MFC Look up the dispatch IDs by calling the Service Object instance's **m_lpDispatch → GetIDsOfNames** function. Update the generated Service Object methods to pass these dispatch IDs to the **InvokeHelper** member function.

   ATL Look up the dispatch IDs by calling the Service Object instance's **GetIDsOfNames** function. Save them for later use – they must be passed to the Service Object dispatch's **Invoke** function.

6. Call the **OpenService** method of the Service Object, passing a device class string, a device name string, and the **IDispatch** pointer to the Control Object. If **OpenService** returns any result except OPOS_SUCCESS, then close the dispatch interface and return the **OpenService** result to the application. If the Service Object supports the method **GetOpenResult**, then call it and set **OpenResult** to its returned value; otherwise set **OpenResult** to OPOS_OR_FAILEDOPEN.

   MFC The Control Object's dispatch pointer is accessed through its **GetIDispatch(FALSE)** member function.

   ATL The Control Object's dispatch pointer is accessed by calling its **QueryInterface** function, requesting an **IDispatch** interface.

7. Call the **GetPropertyNumber(PIDX_ServiceObjectVersion)** method of the Service Object to retrieve its version number. If the major version number is not one (1), then set **OpenResult** to OPOS_OR_BADVERSION and return OPOS_E_NOSERVICE.

8. If any of the dispatch IDs for the methods that should be defined by the Service Object's version are not found, then:
   - call the Service Object's **CloseService** method if present, otherwise call its **Close** method,
   - close the dispatch interface,
   - set **OpenResult** to OPOS_OR_BADIF,
   - and return OPOS_E_NOSERVICE.

   (This ensures that the Service Object supports all of the methods of a valid Service Object for the device class and version it claims to support. If the Service Object's version is newer than the Control Object, then the Control Object ensures that all of the methods for the Control Object's version are supported.)

9. If all of the steps above are successful, then set an internal variable that shows that the Control Object is open, set **OpenResult** to OPOS_SUCCESS, and return OPOS_SUCCESS. Otherwise, the Control Object remains closed.

Close Method

1. If the Control Object is closed, then return OPOS_E_CLOSED.
2. If the Service Object supports the **CloseService** method, then call it. Otherwise, call its **Close** method.
3. Set an internal variable that shows that the Control Object is closed.
4. Release the Service Object.
    - **MFC** Call the **ReleaseDispatch** member function of the Service Object class.
    - **ATL** Call the Service Object dispatch pointer's **Release** member function.
5. Return the result of the Service Object's **Close** method.

Other method calls

1. If the Control Object is closed, then return OPOS_E_CLOSED.
2. If the method was not defined in the Service Object's version of the device class, then:
    - Set the special ResultCode flag to show "version violation state".
    - Return OPOS_E_NOSERVICE.
3. If the method is defined in the Service Object, then:
    - Pass the request down to the Service Object by calling the identically named Service Object method, using an identical list of parameters.
    - Set the special ResultCode flag to show "normal state".
    - Return the result of the Service Object method.

Properties

The Control Object processes property accesses as follows:

1. The Control Object only maintains the properties **ControlObjectDescription, ControlObjectVersion**, and **OpenResult**. The Control Object will handle accesses to these properties directly, and return their value.
2. If the Control Object is closed, then:
    - If setting a property, then return. (There is no means of informing the application that the set failed.)
    - If getting a property, then:
        - If the property is **State**, return OPOS_S_CLOSED.
        - If the property is **ResultCode**, return OPOS_E_CLOSED.
        - Otherwise, return a default property value:
          FALSE for boolean.
          Zero for numeric.
          “[Error]” for string.
3. If getting the property **ResultCode** and the special ResultCode flag is “version violation state”, then return OPOS_E_NOSERVICE.
4. If the property is not supported by the version of the Service Object, then:
    - If setting a property, then set the special ResultCode flag to show “version violation state” and return.
    - If getting a property, then return the default property value.

If not one of the cases above...

1. Set the internal ResultCode flag to show “normal state”.
2. Pass down the request to the Service Object as follows.
    - If the property type is a 4-byte numeric value, including boolean and long, then call the Service Object's **GetPropertyNumber** or **SetPropertyNumber**. A parameter specifies the index of the property. These indices are established in the OPOS header files. (See “OPOS Internal Header Files”, page 55.)
    - If the property type is string, then call the Service Object’s **GetPropertyString** or **SetPropertyString**. A parameter specifies the index of the property. These indices are established in the OPOS header files. (See “OPOS Internal Header Files”, page 55)

- If the property is any other type, then call the Service Object's get or set method for that property.

Events

The Service Object initiates events. The SO calls an event request method exposed by the Control Object.

The mapping of event request methods called by the Service Object into OPOS events is:

Event Request Methods	OPOS Event
SOData	DataEvent
SODirectIO	DirectIOEvent
SOError	ErrorEvent
SOOutputComplete	OutputCompleteEvent
SOStatusUpdate	StatusUpdateEvent

Upon receiving one of these event request methods, the Control Object delivers the appropriate event to the application. The Service Object thread will not regain control until the application event handler has completed.

Warning: These methods are only for use by the Service Object. Though accessible to the application, the application should not call them.

These five event request methods are defined on the following pages.

SOData

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

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

Remarks Requests the Control Object to deliver the event:

void DataEvent (LONG *Status*);

Called by the Service Object to deliver input data from the device to the application. The SO must not call **SOData** unless the **DataEventEnabled** property is TRUE. Just before calling **SOData**, the SO must change this property to FALSE, so that no further data events will be generated until the application sets this property back to TRUE. The actual input data is placed in one or more device class-specific properties.

SODirectIO

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

Parameter	Description
EventNumber	Event number. Specific values assigned by the Service Object.
pData	Pointer to additional numeric data. Specific values vary by *EventNumber* and the Service Object.
pString	Pointer to additional string data. Specific values vary by *EventNumber* and the Service Object.

Remarks Requests the Control Object to deliver the event:

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

Called by the Service Object to communicate information directly to the application.

This event provides a means for a Service Object to deliver events to the application that are not otherwise supported by the Control Object.

The Service Object must ensure that *pString* points to a valid system string, and then must free this string after **SODirectIO** returns.

SOError

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

Parameter	Description
ResultCode	Result code causing the error event. See **ResultCode** in the "Application Programmer's Guide" for values.
ResultCodeExtended	Extended result code causing the error event. See **ResultCodeExtended** in the "Application Programmer's Guide" for values.
ErrorLocus	Location of the error. See values below.
pErrorResponse	Pointer to the error event response. See values below.

The *ErrorLocus* parameter may be one of the following:

Value	Meaning
OPOS_EL_OUTPUT	Error occurred while processing asynchronous output.
OPOS_EL_INPUT	Error occurred while gathering or processing event-driven input. No input data is available.
OPOS_EL_INPUT_DATA	Error occurred while gathering or processing event-driven input, and some previously buffered data is available.

The contents at the location pointed to by the *pErrorResponse* parameter are preset to a default value, based on the *ErrorLocus*. The application may change the value to one of the following:

Value	Meaning
OPOS_ER_RETRY	Typically valid only when locus is OPOS_EL_OUTPUT. Retry the asynchronous output. The error state is exited. May be valid when locus is OPOS_EL_INPUT. Default when locus is OPOS_EL_OUTPUT.
OPOS_ER_CLEAR	Clear the asynchronous output or buffered input data. The error state is exited. Default when locus is OPOS_EL_INPUT.
OPOS_ER_CONTINUEINPUT	Use only when locus is OPOS_EL_INPUT_DATA.

Acknowledges the error and directs the Control to continue processing. The Control remains in the error state and will fire additional **DataEvent**s as directed by the **DataEventEnabled** property. When all input has been fired and the **DataEventEnabled** property is again set to TRUE, then another **ErrorEvent** is fired with locus OPOS_EL_INPUT.
Default when locus is OPOS_EL_INPUT_DATA.

Remarks Requests the Control Object to deliver the event:

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

Once **SOError** has been called, the Service Object must not request another error event until the error has been cleared. However, if an error with locus OPOS_EL_INPUT_DATA is fired and the event handler responds with OPOS_ER_CONTINUEINPUT, then the SO may fire another error event with OPOS_EL_INPUT after the enqueued input has been delivered.

SOOutputComplete

Syntax **void SOOutputComplete (LONG** *OutputID*);

The *OutputID* parameter indicates the number of the asynchronous output request that has completed.

Remarks Requests the Control Object to deliver the event:

void OutputCompleteEvent (LONG *OutputID*);

Called by the Service Object when a previously started asynchronous output request completes successfully.

SOStatusUpdate

Syntax **void SOStatusUpdate (LONG** *Data***);**

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

Remarks Requests the Control Object to deliver the event:

void StatusUpdateEvent (LONG *Data***);**

Called by the Service Object when the SO needs to alert the application of a device status change.

Examples include a change in the cash drawer position (open vs. closed) and a change in a POS printer sensor (form present vs. absent).

The following method is not related to event firing, but is a special purpose support method.

SOProcessID

Syntax **LONG SOProcessID();**

Remarks Return the process ID of the application in which the Control Object exists.

The following method is provided to support local out-proc Service Objects. As noted in the introduction chapter, out-proc servers are not recommended for OPOS Service Objects. However, if a vendor successfully designs and implements such a Service Object, this method may be useful.

For example, if a Service Object which supports Printer with MICR has allowed an application to **Claim** the printer, then it will want to restrict **Claim** of the MICR to the same application, since it is not reasonable for two applications to share such a device with such closely interacting classes.

C H A P T E R 2

Service Object Responsibilities and Implementation

Methods

The following common Service Object methods are defined for implementing corresponding Control Object methods. If a device class does not support a common method (as specified by the device class Summary section in the "Application Programmer's Guide"), then the Service Object should not define that method.

For each device class, additional methods are defined for each device specific method.

The general rules used to define the Service Object methods are:

- The Service Object method name is the same as the Control Object's method name.
- The parameters match those of the Control Object, both in order and type.

The only exceptions to these rules are the **OpenService**, **CloseService** (optional – may use **Close** instead), **GetOpenResult** (optional), and **COFreezeEvents** methods.

Note that these methods are always called through the Service Object's IDispatch interface.

For each of the methods below, syntax is shown for MFC as entered into the control's "Add Method" dialog, and for ATL as entered into the COM object's "Add Method to Interface" dialog.

CheckHealth

MFC long CheckHealth(long *Level***);**
ATL HRESULT CheckHealth(long *Level***, [out, retval] long*** *pRC***);**

Remarks Called to test the state of a device.

ClaimDevice / Claim

MFC long ClaimDevice(long *Timeout***);**
long Claim(long *Timeout***);**

ATL HRESULT ClaimDevice(long *Timeout***, [out, retval] long*** *pRC***);**
HRESULT Claim(long *Timeout***, [out, retval] long*** *pRC***);**

Remarks Called to request exclusive access to the device.

Release 1.0 – 1.4

Control Objects for these releases will only look for the **Claim** method.

Release 1.5 and later

A Control Object for this release will first look for the **ClaimDevice** method. If **ClaimDevice** is not present, then the Control Object looks for **Claim**.

ClearInput

MFC long ClearInput();
ATL HRESULT ClearInput([out, retval] long* *pRC***);**

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

ClearOutput

MFC long ClearOutput();
ATL HRESULT ClearOutput([out, retval] long* *pRC***);**

Remarks Called to clear all device output that has been buffered. Also, when possible, halts outputs that are in progress.

Close

MFC long CloseService();
long Close();
ATL HRESULT CloseService([out, retval] long* *pRC***);**
HRESULT Close([out, retval] long* *pRC***);**

Remarks Called to release the device and its resources.

Release 1.0 – 1.4

Control Objects for these releases will only look for the **Close** method.

Release 1.5 and later

A Control Object for this release will first look for the **CloseService** method. If **CloseService** is not present, then the Control Object looks for **Close**.

COFreezeEvents Internal Control/Service Object Method

MFC long COFreezeEvents(BOOL Freeze);
ATL HRESULT COFreezeEvents(VARIANT_BOOL Freeze,
[out, retval] long* pRC);

The *Freeze* parameter is TRUE / VARIANT_TRUE when event firing must be frozen, and FALSE / VARIANT_FALSE when event firing is reenabled.

Remarks This method is for internal use by the Control Object.

The CO calls it in response to a container event freeze request to inform the SO of a change in the state of event firing. See page 15 for more information.

DirectIO

MFC long DirectIO(long *Command***, long*** *pData***, BSTR*** *pString***);**
ATL HRESULT DirectIO(long *Command***, [in, out] long*** *pData***,**
[in, out] BSTR* *pString***, [out, retval] long*** *pRC***);**

Remarks Call to communicate directly with the Service Object.

GetOpenResult

Internal Control/Service Object Method Added in Release 1.5

MFC long GetOpenResult();
ATL HRESULT GetOpenResult([out, retval] long* ***pRC*****);**

Remarks This method is for internal use by the Control Object.
It is optional.

If a Service Object's **OpenService** method returns a status other than OPOS_SUCCESS, and it has implemented this method, then the Control Object calls this method to set its **OpenResult** property.

The Service Object may select one of the values given in the OPOS.H header file, or return a Service Object-specific value.

Return For MFC implementations, return one of the following values. For ATL implementations, store one of the following values at *pRC*, and return S_OK.

Value	Meaning
OPOS_ORS_NOPORT	The Service Object tried to access an I/O port (for example, an RS232 port) during Open processing, but the port that is configured for the DeviceName is invalid or inaccessible. As a general rule, an SO should refrain from accessing the physical device until the DeviceEnabled property is set to TRUE. But in some cases, it may require some access at Open; for instance, to dynamically determining the device type in order to set the DeviceName and DeviceDescription properties.
OPOS_ORS_NOTSUPPORTED	The Service Object does not support the specified device. The SO has determined that it does not have the ability to control the device it is opening. This determination may be due to an inspection of the registry entries for the device, or dynamic querying of the device during open processing.
OPOS_ORS_CONFIG	Configuration information error. Usually this is due to incomplete configuration of the registry, such that the SO does not have sufficient or valid data to open the device.
OPOS_ORS_SPECIFIC	Errors greater than this value are service object-specific.

If the previous return values do not apply, then the SO may define additional OpenResult values. These values are Service Object-specific, but may be of value in these cases:

1) The Application logs or reports this error during debug and testing.

2) The Application adds SO-specific logic, to attempt to report more error conditions or to recover from them.

OpenService Internal Control/Service Object Method

MFC long OpenService(LPCTSTR *DeviceClass*, **LPCTSTR** *DeviceName*,
LPDISPATCH *pDispatch*);
ATL HRESULT OpenService(BSTR *DeviceClass*, **BSTR** *DeviceName*,
IDispatch* *pDispatch*, **[out, retval] long*** *pRC*);

Parameter	Description
DeviceClass	Contains the requested device class, which are given in the header file OPOS.HI, page 56. Examples are "CashDrawer" and "POSPrinter."
DeviceName	Contains the Device Name to be managed by this Service Object. The relationship between the device name and physical devices is determined by entries within the operating system registry; a setup or configuration utility maintains these entries. (See the "Application Programmer's Guide" appendix "OPOS Registry Usage.")
pDispatch	Points to the Control Object's dispatch interface, which contains the event request methods.

Remarks Call to open a device for subsequent I/O. The Control Object calls this method as part of its **Open** method processing.

The Service Object will use the *DeviceClass* and *DeviceName* parameters in inquiring the registry for additional device specific information.

The following steps assume that the Control Object is written using Visual C++ with MFC. Modify appropriately if another development environment is selected.

For MFC implementations, the *pDispatch* parameter may be used as follows:

1. Attach to the Control Object's **IDispatch** interface by passing the *pDispatch* **IDispatch** pointer to the **AttachDispatch** member function of an instance of a class that defines the Control Object's event request methods.

   This class is generated by using the Visual C++ Class Wizard:

   - Within the "OLE Automation" tab, push the "Add Class from an OLE TypeLib..." button. Then choose the .TLB file generated by a Control Object project.
   - The Class Wizard will generate a **COleDispatchDriver** derivative, with member functions matching the OLE Automation methods exposed by the Control Object.

  - The Class Wizard will also generate an implementation of the member functions, which call **InvokeHelper** with fixed dispatch IDs. Since dispatch IDs depend upon the definition order of the automation methods, this implementation must be updated by the next step to allow for Control Objects that define the methods in a different order.
  - The class definition and implementation should be updated to remove all of the non-event request methods.
2. Look up the event request methods (such as **SOData**) by calling the Control Object instance's **m_lpDispatch → GetIDsOfNames** function. Update the generated Control Object methods to pass these dispatch IDs to the **InvokeHelper** member function.

For ATL implementations, the *pDispatch* parameter may be used directly to call IDispatch's **GetIDsOfNames** and **Invoke** functions. Alternatively, a **CComDispatchDriver** class instance may be created; its **Invoke1** and **InvokeN** functions may be used to call the event functions.

Note

The Service Object attaches back to the Control Object's dispatch pointer in order to access the event request methods within the CO. This implies the following two points:

- When the Control Object exposes the event request methods for access by the Service Object, these methods also become accessible by the application. The application, of course, should not call these methods.
- The Service Object can access other methods and properties within the Control Object. This is not usually beneficial; however, the SO may wish to access the **ControlObjectDescription** or **ControlObjectVersion** to validate compatibility between itself and the CO.

Return For MFC implementations, return one of the following values. For ATL implementations, store one of the following values at *pRC*, and return S_OK.

Value	Meaning
OPOS_SUCCESS	The Service Object is open. This **does not** tell the Control Object or Application that the device is online and functional. Rather, it states that the Service Object software is initialized, and ready to attempt device interaction when the **DeviceEnabled** property is set to TRUE.
Other Values	See **ResultCode** values in the APG.

Any return value except OPOS_SUCCESS is an open failure, and will result in the Control Object shutting down the Service Object (by releasing its COM pointer) and passing this status to the Application.

Since the APG defines meanings for OPOS_E_ILLEGAL and OPOS_E_NOEXIST, a Service Object should return one of these only if the failure is similar to one of these meanings. Otherwise, the Application may be misled.

Release 1.5 and later

On a failure, the Control Object will call the Service Object's **GetOpenResult** method, if present, to retrieve an additional status value.

ReleaseDevice / Release

MFC long ReleaseDevice();
long Release();

ATL HRESULT ReleaseDevice([out, retval] long* *pRC***);**

Remarks Called to release exclusive access to the device.

Release 1.0 – 1.4

Control Objects for these releases will only look for the **Release** method.

Release 1.5 and later

A Control Object for this release will first look for the **ReleaseDevice** method. If **ReleaseDevice** is not present, then the Control Object looks for **Release**.

Note that ATL implementations cannot support the **Release** method (at least not without updating/overriding ATL classes).

Properties

The following methods are defined for getting and setting properties of the following types: 4-byte numeric and string.

For each method, the first parameter is:
LONG *PropIndex*
The values of *PropIndex* are specified in Opos.hi for the common properties. The values of class-specific properties are specified in the class-specific header files.

For robustness, the Service Object should validate the *PropIndex*. If an invalid value is found, then it could display a message box specifying the error, generate a debug exception, or produce another alert to the developer. This type of error should be found during development, testing, or staging prior to rollout to a customer, so the method of informing the user may be rather terse.

Note that these methods are always called through the Service Object's IDispatch interface.

GetPropertyNumber

MFC long GetPropertyNumber(long *PropIndex***);**
ATL HRESULT GetPropertyNumber(long *PropIndex***,**
[out, retval] long* *pNumber***);**

Return The current value of the LONG or BOOL / VARIANT_BOOL property.

For BOOL properties - VARIANT_BOOL COM IDL type - the Common Control Objects return a zero value as VARIANT_FALSE and a non-zero value as VARIANT_TRUE.

GetPropertyString

MFC BSTR GetPropertyString(long *PropIndex***);**
ATL HRESULT GetPropertyString(long *PropIndex***,**
[out, retval] BSTR* *pString***);**

Return The current value of the string property.

SetPropertyNumber

MFC void SetPropertyNumber(long *PropIndex*, **long** *Number***);**
ATL HRESULT SetPropertyNumber(long *PropIndex*, **long** *Number***);**

Remarks Sets the LONG or BOOL property to *Number*.

For BOOL properties - VARIANT_BOOL COM IDL type - the Common Control Objects pass a zero value as zero (0) and a non-zero value as one (1).

SetPropertyString

MFC void SetPropertyString(long *PropIndex*, **LPCTSTR** *String***);**
ATL HRESULT SetPropertyString(long *PropIndex*, **BSTR** *String***);**

Remarks Sets the string property to *String*.

Note – Rationale for property get and set methods

Instead of using the four methods above, the Service Object interface could have defined distinct get methods for every property, plus set methods for writable properties.

Due to the large number of properties present in several Control Objects, however, the four methods above were chosen to reduce the amount of overhead and Service Object code.

Other Types: Not BSTR, LONG, or BOOL

If the Control defines properties of types that are not BStrings, LONGs, or BOOLeans, then the Service Object must define additional get and set methods for these properties.

If using Visual C++ with MFC, this is most easily accomplished through the Class Wizard by adding an Automation property.

Getting Other Property Types

MFC *Type* Get*PropertyName*();
ATL HRESULT Get*PropertyName*([out, retval] *Type pProp);**

Where ***Type*** is replaced by the property's type,
and ***PropertyName*** is replaced by the property's name.

Return The current value of the property.

Example: If a property
CURRENCY SomeMoney;
is defined by the control, then the Service Object must define the method
MFC CURRENCY GetSomeMoney();
ATL HRESULT GetSomeMoney([out, retval] CURRENCY* pCY);

Setting Other Property Types

MFC void Set*PropertyName*(*Type* *value***);**
ATL HRESULT Set*PropertyName*(*Type* *value***);**

Where ***Type*** is replaced by the property's type,
and ***PropertyName*** is replaced by the property's name.

Remarks Sets the property to *value*.

This method is only defined if the property ***PropertyName*** is a writable property.

Example: If a read/write property
CURRENCY SomeMoney;
is defined by the control, then the Service Object must define the method
MFC void SetSomeMoney(CURRENCY *NewMoneyValue***);**
ATL HRESULT SetSomeMoney(CURRENCY *NewMoneyValue***);**

Events

A Service Object causes events to be fired by calling event methods in the Control Object. These methods are named:

SOData
SODirectIO
SOError
SOOutputComplete
SOStatusUpdate

They are described in the Control Object chapter, beginning on page 17.

See the **OpenService** description on page 34 for information about how to get the dispatch interface and dispatch IDs necessary for calling these functions.

APPENDIX A

Change History

Release 1.01

Release 1.01 mostly adds clarifications and corrections, but the Line Display and Signature Capture chapters received substantive changes to correct deficiencies in their definition.

Section	Change
Second Page	Add name of Microsoft Web site for OPOS information.
Opos.hi header file	Remove HKEY_LOCAL_MACHINE from the root keys.
OposPtr.hi header file	Change **...Nearend** to **...NearEnd**. Change **...Barcode** to **...BarCode**.
OposScal.hi header file	Correct **WeightUnits** value from 1 to 2.
OposSig.hi header file	Change **TotalVectors** to **TotalPoints**. Change **VectorArray** to **PointArray**.

Release 1.1

Release 1.1 adds APIs based on requirements from OPOS-J, the Japanese OPOS consortium.

Section	Change
Second Page	Remove CompuServe reference.
Opos.hi header file	Add POS Keyboard values.
OposKbd.hi header file	New header file for POS Keyboard.
OposPtr.hi header file	Add the following properties: **CapCharacterSet** **CapTransaction** **ErrorLevel** **ErrorString** **FontTypefaceList** **RecBarCodeRotationList** **RotateSpecial** **SlpBarCodeRotationList**

Release 1.2

Release 1.2 adds additional device classes, plus additional APIs based on requirements from various OPOS-US, OPOS-Japan, and OPOS-Europe members.

Release 1.2 is a superset of Release 1.1.

Section	Change
First Two Pages	Update company names. Update copyright notices. Update web reference.
Introduction	Add discussion of out-proc and in-proc service objects.
Control Object Chapter	Update to include handling of version mismatch between the Control Object and Service Object. Add the method **SOProcessID**.
Opos.hi header file	Add Cash Changer and Tone Indicator. Add the following properties: **AutoDisable** **BinaryConversion** **DataCount**
OposChan.hi header file	New header file for Cash Changer.
OposMsr.hi header file	Add the property **ErrorReportingType**. Add the property **ParseDecodedData**, with value set the same as **ParseDecodeData**.
OposKbd.hi header file	Add the following properties: **CapKeyUp** **EventTypes** **POSKeyEventType** properties
OposScal.hi header file	Add the following properties: **CapDisplay** **WeightUnit**.

OposScan.hi header file

Add the following properties:
ScanDataLabel
ScanDataType

OposSig.hi header file

Add the following properties:
CapRealTimeData
RealTimeDataEnabled.

OposTone.hi header file

New header file for Tone Indicator.

Release 1.3

Release 1.3 adds additional device classes, a few additional APIs, and some corrections.

Release 1.3 is a superset of Release 1.2.

Section	Change
First Two Pages	Update copyright notices. Update web reference.
General	Modify the use of the term event "firing." Use "enqueue" and "deliver" appropriately to describe event firing.
Control Object Chapter	**SOError**: Allow OPOS_ER_RETRY to be returned on input events if the Control supports it.
Service Object Chapter	Add descriptions of property methods that don't fall into "4-byte number" or "string" types.
Opos.hi header file	Add Bump Bar, Fiscal Printer, PIN Pad, and Remote Order Display. Add the following properties: **CapPowerReporting** **PowerNotify** **PowerState**
OposBb.hi header file	New header file for Bump Bar
OposChan.hi header file	Correct the string indices to use PIDX_STRING instead of PIDX_NUMBER.
OposFptr.hi header file	New header file for Fiscal Printer
OposPPad.hi header file	New header file for PIN Pad
OposROD.hi header file	New header file for Remote Order Display
OposScal.hi header file	Add the following properties: **CapDisplayText** **CapPriceCalculating** **CapTareWeight** **CapZeroScale** **AsyncMode** **MaxDisplayTextChars** **TareWeight**

Several header files	Add validation functions for the first release containing the device.

Release 1.4

Release 1.4 adds 1 additional device class.

Release 1.4 is a superset of Release 1.3.

Section	Change
Opos.hi header file	Add CAT.
OposCat.hi header file	New header file for CAT.

Release 1.5

Release 1.5 adds 2 additional device classes.

Release 1.5 is a superset of Release 1.4.

Section	Change
First Two Pages	Update copyright notices. Update web references.
General	Update **Claim** and **Release** references to include **ClaimDevice** and **ReleaseDevice** information. Update references to OLE to ActiveX where appropriate. Generalize some references to MFC implementations, and add some ATL implementation information.
Control Object Responsibilities	Remove implementation details, and refer to the Common Control Objects.
Service Object **GetOpenResult** method	Add new method.
Opos.hi header file	Added Point Card Reader Writer and POS Power device categories.
OposCash.hi header file	Add **CapMultiDrawerDetect** property.
OposCat.hi header file	Add **PaymentMedia** property
OposCash.hi header file	Add **DepositAmount**, **DepositStatus**, **DeviceStatus**, **CapDeposit**, **CapDepositDataEvent**, **CapPauseDeposit**, **CapRepayDeposit**, **DepositCashList**, **DepositCodeList** and **DepositCounts** properties.
OposMSR.hi header file	Add **CapTransmitSentinels**, **Track4Data** and **TransmitSentinels** properties.
OposPcrw.hi header file	New header file for Point Card Reader Writer.

OposPpad.hi header file	Update to match the released 1.3 header file, then Remove the Amount property index – it isn't a string. Add **Track4Data** property.
OposPtr.hi header file	Add **CapJrnCartridgeSensor**, **CapJrnColor**, **CapRecCartrdigeSensor**, **CapRecColor**, **CapRecMarkFeed**, **CapSlpBothSidesPrint**, **CapSlpCartridgeSensor**, **CapSlpColor**, **CartridgeNotify**, **JrnCartridgeState**, **JrnCurrentCartridge**, **RecCartridgeState**, **RecCurrentCartridge**, **SlpPrintSide**, **SlpCartridgeState**, and **SlpCurrentCartridge** properties.
OposPwr.hi header file	New header file for POS Power.

Release 1.6

Release 1.6 is a superset of Release 1.5.

Section	Change
OposDisp.hi header file	Added **CapBlinkRate**, **CapCursorType**, **CapCustomGlyph**, **CapReadBack**, **CapReverse**, **BlinkRate**, **CursorType**, **CustomGlyphList**, **GlyphHeight** and **GlyphWidth** properties.
OposFptr.hi header file	Added **CapAdditionalHeader**, **CapAdditionalTrailer**, **CapChangeDue**, **CapEmptyReceiptIsVoidable**, **CapFiscalReceiptStation**, **CapFiscalReceiptType**, **CapMultiContractor**, **CapOnlyVoidLastItem**, **CapPackageAdjustment**, **CapPostPreLine**, **CapSetCurrency**, **CapTotalizerType**, **ActualCurrency**, **AdditionHeader**, **AdditionalTrailer**, **ChangeDue**, **ContractorId**, **DateType**, **FiscalReceiptStation**, **FiscalReceiptType**, **MessageType**, **PostLine**, **PreLine** and **TotalizerType** properties.

APPENDIX B

Common Control Objects

As a combination of the personal effort of Curtiss Monroe plus as part of the commitment of his employer, Research Computer Services, Inc. (based in Dayton, Ohio) to the retail community, a complete set of OPOS control objects have been developed for public use. These have been dubbed the "Common Control Objects."

These control objects are delivered as a reference implementation, believed to be correct and suitable for direct use by applications, but not warranted to be correct or to work with any vendor's Service Objects.

Features

- All OPOS controls are supported.
- ATL-based, using dual interfaces so that the app can access them via IDispatch or COM interfaces (of the form IOPOSCashDrawer, etc).
- Built using Microsoft Visual C++. (Currently at Version 6.0, Service Pack 4.)
- Backward compatible with all releases of service objects. This means that they check for older SOs, and return the proper errors to the application if it accesses unsupported properties or methods.
- They have been tested with several major hardware vendors' Service Objects.
- Event firing logic supports well-behaved service objects that fire events from the thread that created the control, plus other service objects that fire them from other threads.
- Self-contained, requiring only standard OS DLLs. Specifically, they do not require MFC or ATL DLLs.
- Both MBCS and Unicode versions have been built and given limited testing. At this time, only the MBCS versions are being posted.
- Source code for all control objects is available.

- For future additions, it is easy to add new control objects or update old ones. A custom generator was developed that reads a data file for each control to be built. To add properties or methods, the procedure is (a) update the data files, (b) regenerate, and (c) build the resulting projects.

Availability and Future

Curtiss intends to maintain the control objects, and post corrections plus new releases at the site http://www.monroecs.com as needed, for as long as he is affiliated with OPOS. Should he not be able to perform this function, then the OPOS Core Committee is authorized to do so.

In order to supply control objects for new devices, the writers of new device chapters may be requested to prepare the approximately 2-page data file used to define some of the key attributes of the device to the generator.

APPENDIX C

OPOS Internal Header Files

The header files are listed in alphabetical order. The mapping of device class name to header file name is as follows:

– General –	Opos.hi
Bump Bar	OposBb.hi
Cash Changer	OposChan.hi
Cash Drawer	OposCash.hi
CAT	OposCat.hi
Coin Dispenser	OposCoin.hi
Fiscal Printer	OposFptr.hi
Hard Totals	OposTot.hi
Keylock	OposLock.hi
Line Display	OposDisp.hi
MICR	OposMicr.hi
MSR	OposMsr.hi
PIN Pad	OposPpad.hi
Point Card Reader Writer	OposPcrw.hi
POS Keyboard	OposKbd.hi
POS Power	OposPwr.hi
POS Printer	OposPtr.hi
Remote Order Display	OposRod.hi
Scale	OposScal.hi
Scanner	OposScan.hi
Signature Capture	OposSig.hi
Tone Indicator	OposTone.hi

Opos.hi : Main OPOS Internal Header File

```
//////////////////////////////////////////////////////////////////////
//
// Opos.hi
//
//   General header file for OPOS Control Objects and Service Objects.
//
// Modification history
// -------------------------------------------------------------------
// 95-12-08 OPOS Release 1.0                                    CRM
// 96-03-18 OPOS Release 1.01                                   CRM
//   Remove HKEY_LOCAL_MACHINE from the root keys, so that they
//     may be directly used by RegOpenKey.
// 96-04-22 OPOS Release 1.1                                    CRM
//   Add POS Keyboard values.
// 97-06-04 OPOS Release 1.2                                    CRM
//   Add Cash Changer and Tone Indicator.
//   Add the following properties:
//     AutoDisable, BinaryConversion, DataCount
// 98-03-06 OPOS Release 1.3                                    CRM
//   Add Bump Bar, Fiscal Printer, PIN Pad, Remote Order Display.
//   Add the following properties:
//     CapPowerReporting, PowerNotify, PowerState
// 98-09-29 OPOS Release 1.4                                 OPOS-J
//   Add CAT.
// 00-09-24 OPOS Release 1.5                                    BKS
//   Added Point Card Reader Writer and POS Power.
//
//////////////////////////////////////////////////////////////////////

#if !defined(OPOS_HI)
#define      OPOS_HI


#include "Opos.h"


//////////////////////////////////////////////////////////////////////
// Registry Keys
//////////////////////////////////////////////////////////////////////

// OPOS_ROOTKEY is the key under which all OPOS Service Object keys
//   and values are placed.  This key will contain device class keys.
//   (The base key is HKEY_LOCAL_MACHINE.)
```

```
#define OPOS_ROOTKEY \
    "SOFTWARE\\OLEforRetail\\ServiceOPOS"

// Device Class Keys
//   Rel 1.0
#define OPOS_CLASSKEY_CASH    "CashDrawer"
#define OPOS_CLASSKEY_COIN    "CoinDispenser"
#define OPOS_CLASSKEY_TOT     "HardTotals"
#define OPOS_CLASSKEY_LOCK    "Keylock"
#define OPOS_CLASSKEY_DISP    "LineDisplay"
#define OPOS_CLASSKEY_MICR    "MICR"
#define OPOS_CLASSKEY_MSR     "MSR"
#define OPOS_CLASSKEY_PTR     "POSPrinter"
#define OPOS_CLASSKEY_SCAL    "Scale"
#define OPOS_CLASSKEY_SCAN    "Scanner"
#define OPOS_CLASSKEY_SIG     "SignatureCapture"
//   Rel 1.1
#define OPOS_CLASSKEY_KBD     "POSKeyboard"
//   Rel 1.2
#define OPOS_CLASSKEY_CHAN    "CashChanger"
#define OPOS_CLASSKEY_TONE    "ToneIndicator"
//   Rel 1.3
#define OPOS_CLASSKEY_BB      "BumpBar"
#define OPOS_CLASSKEY_FPTR    "FiscalPrinter"
#define OPOS_CLASSKEY_PPAD    "PINPad"
#define OPOS_CLASSKEY_ROD     "RemoteOrderDisplay"
//   Rel 1.4
#define OPOS_CLASSKEY_CAT     "CAT"
//   Rel 1.5
#define OPOS_CLASSKEY_PCRW    "PointCardRW"
#define OPOS_CLASSKEY_PWR     "POSPower"

// OPOS_ROOTKEY_PROVIDER is the key under which a Service Object
//   provider may place provider-specific information.
//   (The base key is HKEY_LOCAL_MACHINE.)
#define OPOS_ROOTKEY_PROVIDER \
    "SOFTWARE\\OLEforRetail\\ServiceInfo"


//////////////////////////////////////////////////////////////////////
// Common Property Base Index Values.
//////////////////////////////////////////////////////////////////////

// * Base Indices *

const LONG PIDX_NUMBER                 =        0;
const LONG PIDX_STRING                 =  1000000; // 1,000,000

// * Range Test Functions *

inline BOOL IsNumericPidx(LONG Pidx)
{
  return ( Pidx < PIDX_STRING ) ? TRUE : FALSE;
}
inline BOOL IsStringPidx(LONG Pidx)
{
  return ( Pidx >= PIDX_STRING ) ? TRUE : FALSE;
}

// **Warning**
//   OPOS property indices may not be changed by future releases.
//   New indices must be added sequentially at the end of the
//     numeric, capability, and string sections.
//     Also, the validation functions must be updated.


// Note: The ControlObjectDescription and ControlObjectVersion
//   properties are processed entirely by the CO. Therefore, no
//   property index values are required below.


//////////////////////////////////////////////////////////////////////
// Common Numeric Property Index Values.
//////////////////////////////////////////////////////////////////////

// * Properties *

const LONG PIDX_Claimed                =   1 + PIDX_NUMBER;
const LONG PIDX_DataEventEnabled       =   2 + PIDX_NUMBER;
const LONG PIDX_DeviceEnabled          =   3 + PIDX_NUMBER;
const LONG PIDX_FreezeEvents           =   4 + PIDX_NUMBER;
const LONG PIDX_OutputID               =   5 + PIDX_NUMBER;
const LONG PIDX_ResultCode             =   6 + PIDX_NUMBER;
const LONG PIDX_ResultCodeExtended     =   7 + PIDX_NUMBER;
const LONG PIDX_ServiceObjectVersion   =   8 + PIDX_NUMBER;
const LONG PIDX_State                  =   9 + PIDX_NUMBER;

//      Added for Release 1.2:
const LONG PIDX_AutoDisable            =  10 + PIDX_NUMBER;
const LONG PIDX_BinaryConversion       =  11 + PIDX_NUMBER;
const LONG PIDX_DataCount              =  12 + PIDX_NUMBER;
```

```
//      Added for Release 1.3:
const LONG PIDX_PowerNotify             =  13 + PIDX_NUMBER;
const LONG PIDX_PowerState              =  14 + PIDX_NUMBER;

// * Capabilities *

//      Added for Release 1.3:
const LONG PIDX_CapPowerReporting       = 501 + PIDX_NUMBER;

// * Validation Functions *

inline BOOL IsValidNumericPidx(LONG Pidx)
{
  return ( PIDX_Claimed <= Pidx && Pidx <= PIDX_State )
    ? TRUE : FALSE ;
}

inline BOOL IsValidNumericPidx12(LONG Pidx)
{
  return ( PIDX_Claimed <= Pidx && Pidx <= PIDX_DataCount )
    ? TRUE : FALSE ;
}

inline BOOL IsValidNumericPidx13(LONG Pidx)
{
  return ( PIDX_Claimed <= Pidx && Pidx <= PIDX_PowerState )
    ? TRUE : FALSE ;
}

inline BOOL IsValidCapPidx(LONG Pidx)
{
  return FALSE ;
}

inline BOOL IsValidCapPidx13(LONG Pidx)
{
  return ( PIDX_CapPowerReporting == Pidx )
    ? TRUE : FALSE ;
}


//////////////////////////////////////////////////////////////////////
// Common String Property Index Values.
//////////////////////////////////////////////////////////////////////

// * Properties *
```

```
const LONG PIDX_CheckHealthText        =   1 + PIDX_STRING;
const LONG PIDX_DeviceDescription      =   2 + PIDX_STRING;
const LONG PIDX_DeviceName             =   3 + PIDX_STRING;
const LONG PIDX_ServiceObjectDescription=   4 + PIDX_STRING;

// * Validation Function *

inline BOOL IsValidStringPidx(LONG Pidx)
{
  return ( PIDX_CheckHealthText <= Pidx &&
           Pidx <= PIDX_ServiceObjectDescription )
    ? TRUE : FALSE ;
}


//////////////////////////////////////////////////////////////////////
// Class Property Base Index Values.
//////////////////////////////////////////////////////////////////////

//   Rel 1.0
const LONG PIDX_CASH    =  1000;  // Cash Drawer.
const LONG PIDX_COIN    =  2000;  // Coin Dispenser.
const LONG PIDX_TOT     =  3000;  // Hard Totals.
const LONG PIDX_LOCK    =  4000;  // Keylock.
const LONG PIDX_DISP    =  5000;  // Line Display.
const LONG PIDX_MICR    =  6000;  // Magnetic Ink Character Recognition.
const LONG PIDX_MSR     =  7000;  // Magnetic Stripe Reader.
const LONG PIDX_PTR     =  8000;  // POS Printer.
const LONG PIDX_SCAL    =  9000;  // Scale.
const LONG PIDX_SCAN    = 10000;  // Scanner - Bar Code Reader.
const LONG PIDX_SIG     = 11000;  // Signature Capture.
//   Rel 1.1
const LONG PIDX_KBD     = 12000;  // POS Keyboard.
//   Rel 1.2
const LONG PIDX_CHAN    = 13000;  // Cash Changer.
const LONG PIDX_TONE    = 14000;  // Tone Indicator.
//   Rel 1.3
const LONG PIDX_BB      = 15000;  // Bump Bar.
const LONG PIDX_FPTR    = 16000;  // Fiscal Printer.
const LONG PIDX_PPAD    = 17000;  // PIN Pad.
const LONG PIDX_ROD     = 18000;  // Remote Order Display.
//   Rel 1.4
const LONG PIDX_CAT     = 19000;  // CAT.
//   Rel 1.5
const LONG PIDX_PCRW    = 20000;  // Point Card Reader Writer.
const LONG PIDX_PWR     = 21000;  // POS Power.


#endif                  // !defined(OPOS_HI)
```

OposBb.hi: Bump Bar Internal Header File

```
//////////////////////////////////////////////////////////////////////
//
// OposBb.hi
//
//   Bump Bar header file for OPOS Controls and Service Objects.
//
// Modification history
// -------------------------------------------------------------------
// 98-03-06 OPOS Release 1.3                                     CRM
//
//////////////////////////////////////////////////////////////////////

#if !defined(OPOSBB_HI)
#define      OPOSBB_HI


#include "Opos.hi"
#include "OposBb.h"


//////////////////////////////////////////////////////////////////////
// Numeric Property Index Values.
//////////////////////////////////////////////////////////////////////

// * Properties *

const LONG PIDXBb_AsyncMode            =   1 + PIDX_BB+PIDX_NUMBER;
const LONG PIDXBb_AutoToneDuration     =   2 + PIDX_BB+PIDX_NUMBER;
const LONG PIDXBb_AutoToneFrequency    =   3 + PIDX_BB+PIDX_NUMBER;
const LONG PIDXBb_BumpBarDataCount     =   4 + PIDX_BB+PIDX_NUMBER;
const LONG PIDXBb_CurrentUnitID        =   5 + PIDX_BB+PIDX_NUMBER;
const LONG PIDXBb_ErrorUnits           =   6 + PIDX_BB+PIDX_NUMBER;
const LONG PIDXBb_EventUnitID          =   7 + PIDX_BB+PIDX_NUMBER;
const LONG PIDXBb_EventUnits           =   8 + PIDX_BB+PIDX_NUMBER;
const LONG PIDXBb_Keys                 =   9 + PIDX_BB+PIDX_NUMBER;
const LONG PIDXBb_Timeout              =  10 + PIDX_BB+PIDX_NUMBER;
const LONG PIDXBb_UnitsOnline          =  11 + PIDX_BB+PIDX_NUMBER;

// * Capabilities *

const LONG PIDXBb_CapTone              = 501 + PIDX_BB+PIDX_NUMBER;

// * Validation Functions *
```

```
inline BOOL IsValidBbNumericPidx(LONG Pidx)
{
  return ( PIDXBb_AsyncMode <= Pidx && Pidx <= PIDXBb_UnitsOnline )
    ? TRUE : FALSE ;
}

inline BOOL IsValidBbNumericPidx13(LONG Pidx)
{
  return IsValidBbNumericPidx(Pidx);
}

inline BOOL IsValidBbCapPidx(LONG Pidx)
{
  return ( Pidx == PIDXBb_CapTone )
    ? TRUE : FALSE ;
}

inline BOOL IsValidBbCapPidx13(LONG Pidx)
{
  return IsValidBbCapPidx(Pidx);
}


//////////////////////////////////////////////////////////////////////
// String Property Index Values.
//////////////////////////////////////////////////////////////////////

// * Properties *

const LONG PIDXBb_ErrorString           =   1 + PIDX_BB+PIDX_STRING;
const LONG PIDXBb_EventString           =   2 + PIDX_BB+PIDX_STRING;

// * Validation Function *

inline BOOL IsValidBbStringPidx(LONG Pidx)
{
  return ( PIDXBb_ErrorString <= Pidx && Pidx <= PIDXBb_EventString )
    ? TRUE : FALSE ;
}

inline BOOL IsValidBbStringPidx13(LONG Pidx)
{
  return IsValidBbStringPidx(Pidx);
}
```

```
#endif          // !defined(OPOSBB_HI)
```

OposCash.hi : Cash Drawer Internal Header File

```
//////////////////////////////////////////////////////////////////////
//
// OposCash.hi
//
//   Cash Drawer header file for OPOS Controls and Service Objects.
//
// Modification history
// -------------------------------------------------------------------
// 95-12-08 OPOS Release 1.0                                      CRM
// 00-09-24 OPOS Release 1.5                                      BKS
//
//////////////////////////////////////////////////////////////////////

#if !defined(OPOSCASH_HI)
#define      OPOSCASH_HI


#include "Opos.hi"
#include "OposCash.h"



//////////////////////////////////////////////////////////////////////
// Numeric Property Index Values.
//////////////////////////////////////////////////////////////////////

// * Property *

const LONG PIDXCash_DrawerOpened        =   1 + PIDX_CASH+PIDX_NUMBER;

// * Capabilities *

const LONG PIDXCash_CapStatus           = 501 + PIDX_CASH+PIDX_NUMBER;

//  Added in Release 1.5
const LONG PIDXCash_CapStatusMultiDrawerDetect
                                        = 502 + PIDX_CASH+PIDX_NUMBER;

// * Validation Functions *

inline BOOL IsValidCashNumericPidx(LONG Pidx)
{
  return ( PIDXCash_DrawerOpened == Pidx )
    ? TRUE : FALSE ;
}

inline BOOL IsValidCashCapPidx(LONG Pidx)
{
  return ( PIDXCash_CapStatus == Pidx )
    ? TRUE : FALSE ;
}

inline BOOL IsValidCashCapPidx15(LONG Pidx)
{
  return ( PIDXCash_CapStatus >= Pidx &&
           Pidx <= PIDXCash_CapStatusMultiDrawerDetect )
    ? TRUE : FALSE ;
}


//////////////////////////////////////////////////////////////////////
// String Property Index Values.
//////////////////////////////////////////////////////////////////////

// * Validation Function *

inline BOOL IsValidCashStringPidx(LONG Pidx)
{
  return FALSE;
}


#endif          // !defined(OPOSCASH_HI)
```

OposCat.hi : CAT Internal Header File

```
/////////////////////////////////////////////////////////////////////
//
// OposCat.hi
//
//   CAT header file for OPOS Controls and Service Objects.
//
// Modification history
// ------------------------------------------------------------------
// 98-06-01 OPOS Release 1.4                                  OPOS-J
// 00-09-24 OPOS Release 1.5                                     BKS
//
/////////////////////////////////////////////////////////////////////

#if !defined(OPOSCAT_HI)
#define      OPOSCAT_HI


#include "Opos.hi"
#include "OposCat.h"


//////////////////////////////////////////////////////////////////////
// Numeric Property Index Values.
//////////////////////////////////////////////////////////////////////

// * Properties *

const LONG PIDXCat_AsyncMode            =   1 + PIDX_CAT+PIDX_NUMBER;
const LONG PIDXCat_TrainingMode         =   2 + PIDX_CAT+PIDX_NUMBER;

//  Changed in Release 1.5: In 1.4 TransactionType was incorrectly
//    identified as a String property
const LONG PIDXCat_TransactionType      =   3 + PIDX_CAT+PIDX_NUMBER;

//  Added in Release 1.5
const LONG PIDXCat_PaymentMedia         =   4 + PIDX_CAT+PIDX_NUMBER;

// * Capabilities *

const LONG PIDXCat_CapAdditionalSecurityInformation
                                                    = 501 + PIDX_CAT+PIDX_NUMBER;
const LONG PIDXCat_CapAuthorizeCompletion
                                                    = 502 + PIDX_CAT+PIDX_NUMBER;
```

```
const LONG PIDXCat_CapAuthorizePreSales = 503 + PIDX_CAT+PIDX_NUMBER;
const LONG PIDXCat_CapAuthorizeRefund   = 504 + PIDX_CAT+PIDX_NUMBER;
const LONG PIDXCat_CapAuthorizeVoid     = 505 + PIDX_CAT+PIDX_NUMBER;
const LONG PIDXCat_CapAuthorizeVoidPreSales
                                                     = 506 + PIDX_CAT+PIDX_NUMBER;
const LONG PIDXCat_CapCenterResultCode  = 507 + PIDX_CAT+PIDX_NUMBER;
const LONG PIDXCat_CapCheckCard         = 508 + PIDX_CAT+PIDX_NUMBER;
const LONG PIDXCat_CapDailyLog          = 509 + PIDX_CAT+PIDX_NUMBER;
const LONG PIDXCat_CapInstallments      = 510 + PIDX_CAT+PIDX_NUMBER;
const LONG PIDXCat_CapPaymentDetail     = 511 + PIDX_CAT+PIDX_NUMBER;
const LONG PIDXCat_CapTaxOthers         = 512 + PIDX_CAT+PIDX_NUMBER;
const LONG PIDXCat_CapTransactionNumber = 513 + PIDX_CAT+PIDX_NUMBER;
const LONG PIDXCat_CapTrainingMode      = 514 + PIDX_CAT+PIDX_NUMBER;


// * Validation Functions *

inline BOOL IsValidCatNumericPidx(LONG Pidx)
{
  return ( PIDXCat_AsyncMode <= Pidx &&
           Pidx <= PIDXCat_TransactionType )
    ? TRUE : FALSE ;
}

inline BOOL IsValidCatNumericPidx14(LONG Pidx)
{
  return IsValidCatNumericPidx(Pidx);
}

inline BOOL IsValidCatNumericPidx15(LONG Pidx)
{
  return ( PIDXCat_AsyncMode <= Pidx &&
           Pidx <= PIDXCat_PaymentMedia )
    ? TRUE : FALSE ;
}

inline BOOL IsValidCatCapPidx(LONG Pidx)
{
  return ( PIDXCat_CapAdditionalSecurityInformation <= Pidx &&
           Pidx <= PIDXCat_CapTrainingMode )
    ? TRUE : FALSE ;
}

inline BOOL IsValidCatCapPidx14(LONG Pidx)
{
  return IsValidCatCapPidx(Pidx);
}


//////////////////////////////////////////////////////////////////////
// String Property Index Values.
//////////////////////////////////////////////////////////////////////

// * Properties *

const LONG PIDXCat_AccountNumber        =   1 + PIDX_CAT+PIDX_STRING;
const LONG PIDXCat_AdditionalSecurityInformation
                                        =   2 + PIDX_CAT+PIDX_STRING;
const LONG PIDXCat_ApprovalCode         =   3 + PIDX_CAT+PIDX_STRING;
const LONG PIDXCat_CardCompanyID        =   4 + PIDX_CAT+PIDX_STRING;
const LONG PIDXCat_CenterResultCode     =   5 + PIDX_CAT+PIDX_STRING;
const LONG PIDXCat_DailyLog             =   6 + PIDX_CAT+PIDX_STRING;
const LONG PIDXCat_PaymentCondition     =   7 + PIDX_CAT+PIDX_STRING;
const LONG PIDXCat_PaymentDetail        =   8 + PIDX_CAT+PIDX_STRING;
const LONG PIDXCat_SequenceNumber       =   9 + PIDX_CAT+PIDX_STRING;
const LONG PIDXCat_SlipNumber           =  10 + PIDX_CAT+PIDX_STRING;
const LONG PIDXCat_TransactionNumber    =  11 + PIDX_CAT+PIDX_STRING;

//  Changed in Release 1.5: In 1.4 TransactionType was incorrectly
//    identified as a String property
//const LONG PIDXCat_TransactionType    =  12 + PIDX_CAT+PIDX_STRING;


// * Validation Functions *

inline BOOL IsValidCatStringPidx(LONG Pidx)
{
  return ( PIDXCat_AccountNumber <= Pidx &&
           Pidx <= PIDXCat_TransactionNumber )
    ? TRUE : FALSE ;
}

inline BOOL IsValidCatStringPidx14(LONG Pidx)
{
  return IsValidCatStringPidx(Pidx);
}


#endif          // !defined(OPOSCAT_HI)
```

OposChan.hi : Cash Changer Internal Header File

```
//////////////////////////////////////////////////////////////////////
//
// OposChan.hi
//
//   Cash Changer header file for OPOS Controls and Service Objects.
//
// Modification history
// ------------------------------------------------------------------
// 97-06-04 OPOS Release 1.2                                     CRM
// 00-09-24 OPOS Release 1.5                                     BKS
//
//////////////////////////////////////////////////////////////////////

#if !defined(OPOSCHAN_HI)
#define      OPOSCHAN_HI


#include "Opos.hi"
#include "OposChan.h"



//////////////////////////////////////////////////////////////////////
// Numeric Property Index Values.
//////////////////////////////////////////////////////////////////////

// * Properties *

const LONG PIDXChan_AsyncMode           =   1 + PIDX_CHAN+PIDX_NUMBER;
const LONG PIDXChan_AsyncResultCode     =   2 + PIDX_CHAN+PIDX_NUMBER;
const LONG PIDXChan_AsyncResultCodeExtended
                                        =   3 + PIDX_CHAN+PIDX_NUMBER;
const LONG PIDXChan_CurrentExit         =   4 + PIDX_CHAN+PIDX_NUMBER;
const LONG PIDXChan_DeviceExits         =   5 + PIDX_CHAN+PIDX_NUMBER;
const LONG PIDXChan_DeviceStatus        =   6 + PIDX_CHAN+PIDX_NUMBER;
const LONG PIDXChan_FullStatus          =   7 + PIDX_CHAN+PIDX_NUMBER;

// Added in Release 1.5
const LONG PIDXChan_DepositAmount       =   8 + PIDX_CHAN+PIDX_NUMBER;
const LONG PIDXChan_DepositStatus       =   9 + PIDX_CHAN+PIDX_NUMBER;

// * Capabilities *

const LONG PIDXChan_CapDiscrepancy      = 501 + PIDX_CHAN+PIDX_NUMBER;
const LONG PIDXChan_CapEmptySensor      = 502 + PIDX_CHAN+PIDX_NUMBER;
const LONG PIDXChan_CapFullSensor       = 503 + PIDX_CHAN+PIDX_NUMBER;
const LONG PIDXChan_CapNearEmptySensor  = 504 + PIDX_CHAN+PIDX_NUMBER;
const LONG PIDXChan_CapNearFullSensor   = 505 + PIDX_CHAN+PIDX_NUMBER;

// Added in Release 1.5
const LONG PIDXChan_CapDeposit          = 506 + PIDX_CHAN+PIDX_NUMBER;
const LONG PIDXChan_CapDepositDataEvent = 507 + PIDX_CHAN+PIDX_NUMBER;
const LONG PIDXChan_CapPauseDeposit     = 508 + PIDX_CHAN+PIDX_NUMBER;
const LONG PIDXChan_CapRepayDeposit     = 509 + PIDX_CHAN+PIDX_NUMBER;

// * Validation Functions *

inline BOOL IsValidChanNumericPidx(LONG Pidx)
{
  return ( PIDXChan_AsyncMode <= Pidx && Pidx <= PIDXChan_FullStatus )
    ? TRUE : FALSE ;
}

inline BOOL IsValidChanNumericPidx12(LONG Pidx)
{
  return IsValidChanNumericPidx(Pidx);
}

inline BOOL IsValidChanNumericPidx15(LONG Pidx)
{
  return ( PIDXChan_AsyncMode <= Pidx && Pidx <= PIDXChan_DepositStatus )
    ? TRUE : FALSE ;
}

inline BOOL IsValidChanCapPidx(LONG Pidx)
{
  return ( PIDXChan_CapDiscrepancy <= Pidx &&
           Pidx <= PIDXChan_CapNearFullSensor )
    ? TRUE : FALSE ;
}

inline BOOL IsValidChanCapPidx12(LONG Pidx)
{
  return IsValidChanCapPidx(Pidx);
}

inline BOOL IsValidChanCapPidx15(LONG Pidx)
{
  return ( PIDXChan_CapDiscrepancy <= Pidx &&
           Pidx <= PIDXChan_CapRepayDeposit )
    ? TRUE : FALSE ;
}


//////////////////////////////////////////////////////////////////////
// String Property Index Values.
//////////////////////////////////////////////////////////////////////

const LONG PIDXChan_CurrencyCashList    =   1 + PIDX_CHAN+PIDX_STRING;
const LONG PIDXChan_CurrencyCode        =   2 + PIDX_CHAN+PIDX_STRING;
const LONG PIDXChan_CurrencyCodeList    =   3 + PIDX_CHAN+PIDX_STRING;
const LONG PIDXChan_ExitCashList        =   4 + PIDX_CHAN+PIDX_STRING;

// Added in Release 1.5
const LONG PIDXChan_DepositCashList     =   5 + PIDX_CHAN+PIDX_STRING;
const LONG PIDXChan_DepositCodeList     =   6 + PIDX_CHAN+PIDX_STRING;
const LONG PIDXChan_DepositCounts       =   7 + PIDX_CHAN+PIDX_STRING;

// * Validation Function *

inline BOOL IsValidChanStringPidx(LONG Pidx)
{
  return ( PIDXChan_CurrencyCashList <= Pidx &&
           Pidx <= PIDXChan_ExitCashList )
    ? TRUE : FALSE ;
}

inline BOOL IsValidChanStringPidx12(LONG Pidx)
{
  return IsValidChanStringPidx(Pidx);
}

inline BOOL IsValidChanStringPidx15(LONG Pidx)
{
  return ( PIDXChan_CurrencyCashList <= Pidx &&
           Pidx <= PIDXChan_DepositCounts )
    ? TRUE : FALSE ;
}


#endif          // !defined(OPOSCHAN_HI)
```

OposCoin.hi : Coin Dispenser Internal Header File

```
//////////////////////////////////////////////////////////////////////
//
// OposCoin.hi
//
//   Coin Dispenser header file for OPOS Controls and Service Objects.
//
// Modification history
// ------------------------------------------------------------------
// 95-12-08 OPOS Release 1.0                                     CRM
//
//////////////////////////////////////////////////////////////////////

#if !defined(OPOSCOIN_HI)
#define      OPOSCOIN_HI


#include "Opos.hi"
#include "OposCoin.h"


//////////////////////////////////////////////////////////////////////
// Numeric Property Index Values.
//////////////////////////////////////////////////////////////////////

// * Property *

const LONG PIDXCoin_DispenserStatus     =   1 + PIDX_COIN+PIDX_NUMBER;

// * Capabilities *

const LONG PIDXCoin_CapEmptySensor      = 501 + PIDX_COIN+PIDX_NUMBER;
const LONG PIDXCoin_CapJamSensor        = 502 + PIDX_COIN+PIDX_NUMBER;
const LONG PIDXCoin_CapNearEmptySensor  = 503 + PIDX_COIN+PIDX_NUMBER;

// * Validation Functions *

inline BOOL IsValidCoinNumericPidx(LONG Pidx)
{
  return ( PIDXCoin_DispenserStatus == Pidx )
    ? TRUE : FALSE ;
}

inline BOOL IsValidCoinCapPidx(LONG Pidx)
```

```
{
  return ( PIDXCoin_CapEmptySensor <= Pidx &&
           Pidx <= PIDXCoin_CapNearEmptySensor )
    ? TRUE : FALSE ;
}


//////////////////////////////////////////////////////////////////////
// String Property Index Values.
//////////////////////////////////////////////////////////////////////

// * Validation Function *

inline BOOL IsValidCoinStringPidx(LONG Pidx)
{
  return FALSE;
}


#endif          // !defined(OPOSCOIN_HI)
```

OposDisp.hi : Line Display Internal Header File

```
//////////////////////////////////////////////////////////////////////
//
// OposDisp.hi
//
//   Line Display header file for OPOS Controls and Service Objects.
//
// Modification history
// ------------------------------------------------------------------
// 95-12-08 OPOS Release 1.0                                     CRM
// 96-03-18 OPOS Release 1.01                                    CRM
//   Add MarqueeFormat property.
// 01-07-15 OPOS Release 1.6                                     BKS
//   Added BlinkRate, CursorType, CustomSSGlyphList, GlyphHeight
//     and GlyphWidth properties.
//   Added CapBlinkRate, CapCursorType, CapCustomGlyph, CapReadBack
//     and CapReverse capabilities.
//
//////////////////////////////////////////////////////////////////////

#if !defined(OPOSDISP_HI)
#define      OPOSDISP_HI


#include "Opos.hi"
#include "OposDisp.h"


#if defined(CreateWindow) // If Win32 defines "CreateWindow":
#undef CreateWindow       //   Undefine it to avoid conflict
#endif                    //   with the line display method.


//////////////////////////////////////////////////////////////////////
// Numeric Property Index Values.
//////////////////////////////////////////////////////////////////////

// * Properties *

const LONG PIDXDisp_CharacterSet        =   1 + PIDX_DISP+PIDX_NUMBER;
const LONG PIDXDisp_Columns             =   2 + PIDX_DISP+PIDX_NUMBER;
const LONG PIDXDisp_CurrentWindow       =   3 + PIDX_DISP+PIDX_NUMBER;
const LONG PIDXDisp_CursorColumn        =   4 + PIDX_DISP+PIDX_NUMBER;
const LONG PIDXDisp_CursorRow           =   5 + PIDX_DISP+PIDX_NUMBER;
const LONG PIDXDisp_CursorUpdate       =   6 + PIDX_DISP+PIDX_NUMBER;
const LONG PIDXDisp_DeviceBrightness   =   7 + PIDX_DISP+PIDX_NUMBER;
const LONG PIDXDisp_DeviceColumns      =   8 + PIDX_DISP+PIDX_NUMBER;
const LONG PIDXDisp_DeviceDescriptors  =   9 + PIDX_DISP+PIDX_NUMBER;
const LONG PIDXDisp_DeviceRows         =  10 + PIDX_DISP+PIDX_NUMBER;
const LONG PIDXDisp_DeviceWindows      =  11 + PIDX_DISP+PIDX_NUMBER;
const LONG PIDXDisp_InterCharacterWait =  12 + PIDX_DISP+PIDX_NUMBER;
const LONG PIDXDisp_MarqueeRepeatWait  =  13 + PIDX_DISP+PIDX_NUMBER;
const LONG PIDXDisp_MarqueeType        =  14 + PIDX_DISP+PIDX_NUMBER;
const LONG PIDXDisp_MarqueeUnitWait    =  15 + PIDX_DISP+PIDX_NUMBER;
const LONG PIDXDisp_Rows               =  16 + PIDX_DISP+PIDX_NUMBER;
const LONG PIDXDisp_MarqueeFormat      =  17 + PIDX_DISP+PIDX_NUMBER;

// Added in Release 1.6
const LONG PIDXDisp_BlinkRate          =  18 + PIDX_DISP+PIDX_NUMBER;
const LONG PIDXDisp_CursorType         =  19 + PIDX_DISP+PIDX_NUMBER;
const LONG PIDXDisp_GlyphHeight        =  20 + PIDX_DISP+PIDX_NUMBER;
const LONG PIDXDisp_GlyphWidth         =  21 + PIDX_DISP+PIDX_NUMBER;

// * Capabilities *

const LONG PIDXDisp_CapBlink           = 501 + PIDX_DISP+PIDX_NUMBER;
const LONG PIDXDisp_CapBrightness      = 502 + PIDX_DISP+PIDX_NUMBER;
const LONG PIDXDisp_CapCharacterSet    = 503 + PIDX_DISP+PIDX_NUMBER;
const LONG PIDXDisp_CapDescriptors     = 504 + PIDX_DISP+PIDX_NUMBER;
const LONG PIDXDisp_CapHMarquee        = 505 + PIDX_DISP+PIDX_NUMBER;
const LONG PIDXDisp_CapICharWait       = 506 + PIDX_DISP+PIDX_NUMBER;
const LONG PIDXDisp_CapVMarquee        = 507 + PIDX_DISP+PIDX_NUMBER;

// Added in Release 1.6
const LONG PIDXDisp_CapBlinkRate       = 508 + PIDX_DISP+PIDX_NUMBER;
const LONG PIDXDisp_CapCursorType      = 509 + PIDX_DISP+PIDX_NUMBER;
const LONG PIDXDisp_CapCustomGlyph     = 510 + PIDX_DISP+PIDX_NUMBER;
const LONG PIDXDisp_CapReadBack        = 511 + PIDX_DISP+PIDX_NUMBER;
const LONG PIDXDisp_CapReverse         = 512 + PIDX_DISP+PIDX_NUMBER;

// * Validation Functions *

inline BOOL IsValidDispNumericPidx(LONG Pidx)
{
  return ( PIDXDisp_CharacterSet <= Pidx &&
           Pidx <= PIDXDisp_MarqueeFormat )
    ? TRUE : FALSE ;
}

inline BOOL IsValidDispNumericPidx12(LONG Pidx)
```

```
{
  return ( PIDXDisp_CharacterSet <= Pidx &&
           Pidx <= PIDXDisp_MarqueeFormat )
    ? TRUE : FALSE ;
}

inline BOOL IsValidDispNumericPidx16(LONG Pidx)
{
  return ( PIDXDisp_CharacterSet <= Pidx &&
           Pidx <= PIDXDisp_GlyphWidth )
    ? TRUE : FALSE ;
}

inline BOOL IsValidDispCapPidx(LONG Pidx)
{
  return ( PIDXDisp_CapBlink <= Pidx && Pidx <= PIDXDisp_CapVMarquee )
    ? TRUE : FALSE ;
}

inline BOOL IsValidDispCapPidx12(LONG Pidx)
{
  return ( PIDXDisp_CapBlink <= Pidx && Pidx <= PIDXDisp_CapVMarquee )
    ? TRUE : FALSE ;
}

inline BOOL IsValidDispCapPidx16(LONG Pidx)
{
  return ( PIDXDisp_CapBlink <= Pidx && Pidx <= PIDXDisp_CapReverse )
    ? TRUE : FALSE ;
}


//////////////////////////////////////////////////////////////////////
// String Property Index Values.
//////////////////////////////////////////////////////////////////////

// * Properties *

const LONG PIDXDisp_CharacterSetList    =   1 + PIDX_DISP+PIDX_STRING;

// Added in Release 1.6
const LONG PIDXDisp_CustomGlyphList     =   2 + PIDX_DISP+PIDX_STRING;

// * Validation Function *

inline BOOL IsValidDispStringPidx(LONG Pidx)
```

```
{
  return ( PIDXDisp_CharacterSetList == Pidx )
    ? TRUE : FALSE ;
}

inline BOOL IsValidDispStringPidx12(LONG Pidx)
{
  return ( PIDXDisp_CharacterSetList == Pidx )
    ? TRUE : FALSE ;
}

inline BOOL IsValidDispStringPidx16(LONG Pidx)
{
  return ( PIDXDisp_CharacterSetList <= Pidx &&
           Pidx <= PIDXDisp_CustomGlyphList)
    ? TRUE : FALSE ;
}


#endif          // !defined(OPOSDISP_HI)
```

OposFptr.hi: Fiscal Printer Internal Header File

```
//////////////////////////////////////////////////////////////////////
//
// OposFptr.hi
//
//   Fiscal Printer header file for OPOS Controls and Service Objects.
//
// Modification history
// -------------------------------------------------------------------
// 98-03-06 OPOS Release 1.3                                     PDU
// 01-07-15 OPOS Release 1.6                                     THH
//
//////////////////////////////////////////////////////////////////////

#if !defined(OPOSFPTR_HI)
#define      OPOSFPTR_HI


#include "Opos.hi"
#include "OposFptr.h"


//////////////////////////////////////////////////////////////////////
// Numeric Property Index Values.
//////////////////////////////////////////////////////////////////////

// * Properties *

const LONG PIDXFptr_AmountDecimalPlaces =   1 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_AsyncMode           =   2 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_CheckTotal          =   3 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_CountryCode         =   4 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_CoverOpen           =   5 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_DayOpened           =   6 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_DescriptionLength   =   7 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_DuplicateReceipt    =   8 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_ErrorLevel          =   9 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_ErrorOutID          =  10 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_ErrorState          =  11 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_ErrorStation        =  12 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_FlagWhenIdle        =  13 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_JrnEmpty            =  14 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_JrnNearEnd          =  15 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_MessageLength       =  16 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_NumHeaderLines      =  17 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_NumTrailerLines     =  18 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_NumVatRates         =  19 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_PrinterState        =  20 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_QuantityDecimalPlaces
                                        =  21 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_QuantityLength      =  22 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_RecEmpty            =  23 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_RecNearEnd          =  24 + PIDX_FPTR+PIDX_NUMBER;

const LONG PIDXFptr_RemainingFiscalMemory
                                        =  25 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_SlpEmpty            =  26 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_SlpNearEnd          =  27 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_SlipSelection       =  28 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_TrainingModeActive  =  29 + PIDX_FPTR+PIDX_NUMBER;

//      Added for Release 1.6:
const LONG PIDXFptr_ActualCurrency      =  30 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_ContractorId        =  31 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_DateType            =  32 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_FiscalReceiptStation
                                        =  33 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_FiscalReceiptType   =  34 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_MessageType         =  35 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_TotalizerType       =  36 + PIDX_FPTR+PIDX_NUMBER;


// * Capabilities *

const LONG PIDXFptr_CapAdditionalLines  = 501 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_CapAmountAdjustment = 502 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_CapAmountNotPaid    = 503 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_CapCheckTotal       = 504 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_CapCoverSensor      = 505 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_CapDoubleWidth      = 506 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_CapDuplicateReceipt = 507 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_CapFixedOutput      = 508 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_CapHasVatTable      = 509 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_CapIndependentHeader
                                        = 510 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_CapItemList         = 511 + PIDX_FPTR+PIDX_NUMBER;

const LONG PIDXFptr_CapJrnEmptySensor   = 512 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_CapJrnNearEndSensor = 513 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_CapJrnPresent       = 514 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_CapNonFiscalMode    = 515 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_CapOrderAdjustmentFirst
                                        = 516 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_CapPercentAdjustment
                                        = 517 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_CapPositiveAdjustment
                                        = 518 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_CapPowerLossReport  = 519 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_CapPredefinedPaymentLines
                                        = 520 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_CapReceiptNotPaid   = 521 + PIDX_FPTR+PIDX_NUMBER;

const LONG PIDXFptr_CapRecEmptySensor   = 522 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_CapRecNearEndSensor = 523 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_CapRecPresent       = 524 + PIDX_FPTR+PIDX_NUMBER;

const LONG PIDXFptr_CapRemainingFiscalMemory
                                        = 525 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_CapReservedWord     = 526 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_CapSetHeader        = 527 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_CapSetPOSID         = 528 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_CapSetStoreFiscalID = 529 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_CapSetTrailer       = 530 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_CapSetVatTable      = 531 + PIDX_FPTR+PIDX_NUMBER;

const LONG PIDXFptr_CapSlpEmptySensor   = 532 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_CapSlpFiscalDocument
                                        = 533 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_CapSlpFullSlip      = 534 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_CapSlpNearEndSensor = 535 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_CapSlpPresent       = 536 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_CapSlpValidation    = 537 + PIDX_FPTR+PIDX_NUMBER;

const LONG PIDXFptr_CapSubAmountAdjustment
                                        = 538 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_CapSubPercentAdjustment
                                        = 539 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_CapSubtotal         = 540 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_CapTrainingMode     = 541 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_CapValidateJournal  = 542 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_CapXReport          = 543 + PIDX_FPTR+PIDX_NUMBER;

//      Added for Release 1.6:
const LONG PIDXFptr_CapAdditionalHeader = 544 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_CapAdditionalTrailer
```

```
                                        = 545 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_CapChangeDue        = 546 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_CapEmptyReceiptIsVoidable
                                        = 547 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_CapFiscalReceiptStation
                                        = 548 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_CapFiscalReceiptType
                                        = 549 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_CapMultiContractor  = 550 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_CapOnlyVoidLastItem = 551 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_CapPackageAdjustment
                                        = 552 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_CapPostPreLine      = 553 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_CapSetCurrency      = 554 + PIDX_FPTR+PIDX_NUMBER;
const LONG PIDXFptr_CapTotalizerType    = 555 + PIDX_FPTR+PIDX_NUMBER;

 // * Validation Functions *

inline BOOL IsValidFptrNumericPidx(LONG Pidx)
{
  return ( PIDXFptr_AmountDecimalPlaces <= Pidx &&
           Pidx <= PIDXFptr_TrainingModeActive )
    ? TRUE : FALSE ;
}

inline BOOL IsValidFptrNumericPidx13(LONG Pidx)
{
  return IsValidFptrNumericPidx(Pidx);
}

inline BOOL IsValidFptrNumericPidx16(LONG Pidx)
{
  return ( PIDXFptr_AmountDecimalPlaces <= Pidx &&
           Pidx <= PIDXFptr_TotalizerType )
    ? TRUE : FALSE ;
}

inline BOOL IsValidFptrCapPidx(LONG Pidx)
{
  return ( PIDXFptr_CapAdditionalLines <= Pidx &&
           Pidx <= PIDXFptr_CapXReport )
    ? TRUE : FALSE ;
}

inline BOOL IsValidFptrCapPidx13(LONG Pidx)
{
  return IsValidFptrCapPidx(Pidx);
}

inline BOOL IsValidFptrCapPidx16(LONG Pidx)
{
  return ( PIDXFptr_CapAdditionalLines <= Pidx &&
           Pidx <= PIDXFptr_CapTotalizerType )
    ? TRUE : FALSE ;
}


//////////////////////////////////////////////////////////////////////
// String Property Index Values.
//////////////////////////////////////////////////////////////////////

const LONG PIDXFptr_ErrorString        =   1 + PIDX_FPTR+PIDX_STRING;
const LONG PIDXFptr_PredefinedPaymentLines
                                       =   2 + PIDX_FPTR+PIDX_STRING;
const LONG PIDXFptr_ReservedWord       =   3 + PIDX_FPTR+PIDX_STRING;

//      Added for Release 1.6:
const LONG PIDXFptr_AdditionalHeader   =   4 + PIDX_FPTR+PIDX_STRING;
const LONG PIDXFptr_AdditionalTrailer  =   5 + PIDX_FPTR+PIDX_STRING;
const LONG PIDXFptr_ChangeDue          =   6 + PIDX_FPTR+PIDX_STRING;
const LONG PIDXFptr_PostLine           =   7 + PIDX_FPTR+PIDX_STRING;
const LONG PIDXFptr_PreLine            =   8 + PIDX_FPTR+PIDX_STRING;

// * Validation Function *

inline BOOL IsValidFptrStringPidx(LONG Pidx)
{
  return ( PIDXFptr_ErrorString <= Pidx &&
           Pidx <= PIDXFptr_ReservedWord )
    ? TRUE : FALSE ;
}

inline BOOL IsValidFptrStringPidx13(LONG Pidx)
{
  return IsValidFptrStringPidx(Pidx);
}

inline BOOL IsValidFptrStringPidx16(LONG Pidx)
{
  return ( PIDXFptr_ErrorString <= Pidx &&
           Pidx <= PIDXFptr_PreLine )
    ? TRUE : FALSE ;
```

```
}

#endif          // !defined(OPOSFPTR_HI)
```

OposKbd.hi : POS Keyboard Internal Header File

```
//////////////////////////////////////////////////////////////////////
//
// OposKbd.hi
//
//   POS Keyboard header file for OPOS Controls and Service Objects.
//
// Modification history
// -------------------------------------------------------------------
// 96-04-22 OPOS Release 1.1                                     CRM
// 97-06-04 OPOS Release 1.2                                     CRM
//   Add the following properties:
//     CapKeyUp, EventTypes, POSKeyEventType
//
//////////////////////////////////////////////////////////////////////

#if !defined(OPOSKBD_HI)
#define      OPOSKBD_HI


#include "Opos.hi"
#include "OposKbd.h"


//////////////////////////////////////////////////////////////////////
// Numeric Property Index Values.
//////////////////////////////////////////////////////////////////////

// * Properties *

const LONG PIDXKbd_POSKeyData          =   1 + PIDX_KBD+PIDX_NUMBER;

//      Added for Release 1.2:
const LONG PIDXKbd_EventTypes          =   2 + PIDX_KBD+PIDX_NUMBER;
const LONG PIDXKbd_POSKeyEventType     =   3 + PIDX_KBD+PIDX_NUMBER;

// * Capabilities *

//      Added for Release 1.2:
const LONG PIDXKbd_CapKeyUp            = 501 + PIDX_KBD+PIDX_NUMBER;

// * Validation Functions *

inline BOOL IsValidKbdNumericPidx(LONG Pidx)
```

```
{
  return ( PIDXKbd_POSKeyData == Pidx )
    ? TRUE : FALSE ;
}

inline BOOL IsValidKbdNumericPidx11(LONG Pidx)
{
  return IsValidKbdNumericPidx(Pidx);
}

inline BOOL IsValidKbdNumericPidx12(LONG Pidx)
{
  return ( PIDXKbd_POSKeyData <= Pidx &&
           Pidx <= PIDXKbd_POSKeyEventType )
    ? TRUE : FALSE ;
}

inline BOOL IsValidKbdCapPidx(LONG Pidx)
{
  return FALSE;
}

inline BOOL IsValidKbdCapPidx11(LONG Pidx)
{
  return IsValidKbdCapPidx(Pidx);
}

inline BOOL IsValidKbdCapPidx12(LONG Pidx)
{
  return ( PIDXKbd_CapKeyUp == Pidx )
    ? TRUE : FALSE ;
}


///////////////////////////////////////////////////////////////////////
// String Property Index Values.
///////////////////////////////////////////////////////////////////////

// * Validation Function *

inline BOOL IsValidKbdStringPidx(LONG Pidx)
{
  return FALSE;
}

inline BOOL IsValidKbdStringPidx11(LONG Pidx)
```

```
{
  return IsValidKbdStringPidx(Pidx);
}


#endif          // !defined(OPOSKBD_HI)
```

OposLock.hi : Keylock Internal Header File

```
//////////////////////////////////////////////////////////////////////
//
// OposLock.hi
//
//   Keylock header file for OPOS Controls and Service Objects.
//
// Modification history
// -------------------------------------------------------------------
// 95-12-08 OPOS Release 1.0                                     CRM
//
//////////////////////////////////////////////////////////////////////

#if !defined(OPOSLOCK_HI)
#define      OPOSLOCK_HI


#include "Opos.hi"
#include "OposLock.h"


//////////////////////////////////////////////////////////////////////
// Numeric Property Index Values.
//////////////////////////////////////////////////////////////////////

// * Properties *

const LONG PIDXLock_KeyPosition        =   1 + PIDX_LOCK+PIDX_NUMBER;
const LONG PIDXLock_PositionCount      =   2 + PIDX_LOCK+PIDX_NUMBER;

// * Validation Functions *

inline BOOL IsValidLockNumericPidx(LONG Pidx)
{
  return ( PIDXLock_KeyPosition <= Pidx &&
           Pidx <= PIDXLock_PositionCount )
    ? TRUE : FALSE ;
}

inline BOOL IsValidLockCapPidx(LONG Pidx)
{
  return FALSE;
}
```

```
/////////////////////////////////////////////////////////////////////
// String Property Index Values.
/////////////////////////////////////////////////////////////////////

// * Validation Function *

inline BOOL IsValidLockStringPidx(LONG Pidx)
{
  return FALSE;
}


#endif          // !defined(OPOSLOCK_HI)
```

OposMicr.hi : MICR Internal Header File

```
//////////////////////////////////////////////////////////////////////
//
// OposMicr.hi
//
//   MICR header file for OPOS Controls and Service Objects.
//
// Modification history
// -------------------------------------------------------------------
// 95-12-08 OPOS Release 1.0                                      CRM
// 97-06-04 OPOS Release 1.2                                      CRM
//   Correct TransitNumber from number to string.
//
//////////////////////////////////////////////////////////////////////

#if !defined(OPOSMICR_HI)
#define      OPOSMICR_HI


#include "Opos.hi"
#include "OposMicr.h"


//////////////////////////////////////////////////////////////////////
// Numeric Property Index Values.
//////////////////////////////////////////////////////////////////////

// * Properties *

const LONG PIDXMicr_CheckType          =   1 + PIDX_MICR+PIDX_NUMBER;
const LONG PIDXMicr_CountryCode        =   2 + PIDX_MICR+PIDX_NUMBER;

// * Capabilities *

const LONG PIDXMicr_CapValidationDevice = 501 + PIDX_MICR+PIDX_NUMBER;

// * Validation Functions *

inline BOOL IsValidMicrNumericPidx(LONG Pidx)
{
  return ( PIDXMicr_CheckType <= Pidx &&
           Pidx <= PIDXMicr_CountryCode )
    ? TRUE : FALSE ;
}
```

```
inline BOOL IsValidMicrCapPidx(LONG Pidx)
{
  return ( PIDXMicr_CapValidationDevice == Pidx )
    ? TRUE : FALSE ;
}


//////////////////////////////////////////////////////////////////////
// String Property Index Values.
//////////////////////////////////////////////////////////////////////

// * Properties *

const LONG PIDXMicr_AccountNumber       =   1 + PIDX_MICR+PIDX_STRING;
const LONG PIDXMicr_Amount              =   2 + PIDX_MICR+PIDX_STRING;
const LONG PIDXMicr_BankNumber          =   3 + PIDX_MICR+PIDX_STRING;
const LONG PIDXMicr_EPC                 =   4 + PIDX_MICR+PIDX_STRING;
const LONG PIDXMicr_RawData             =   5 + PIDX_MICR+PIDX_STRING;
const LONG PIDXMicr_SerialNumber        =   6 + PIDX_MICR+PIDX_STRING;
const LONG PIDXMicr_TransitNumber       =   7 + PIDX_MICR+PIDX_STRING;

// * Validation Function *

inline BOOL IsValidMicrStringPidx(LONG Pidx)
{
  return ( PIDXMicr_AccountNumber <= Pidx &&
           Pidx <= PIDXMicr_TransitNumber )
    ? TRUE : FALSE ;
}


#endif          // !defined(OPOSMICR_HI)
```

OposMsr.hi : MSR Internal Header File

```
//////////////////////////////////////////////////////////////////////
//
// OposMsr.hi
//
//   MSR header file for OPOS Controls and Service Objects.
//
// Modification history
// -------------------------------------------------------------------
// 95-12-08 OPOS Release 1.0                                     CRM
// 97-06-04 OPOS Release 1.2                                     CRM
//   Add the following properties:
//     ErrorReportingType, ParseDecodedData
// 00-09-24 OPOS Release 1.5                                     BKS
//   Add the following properties:
//     CapTransmitSentinels, Track4Data, TransmitSentinels
//
//////////////////////////////////////////////////////////////////////

#if !defined(OPOSMSR_HI)
#define      OPOSMSR_HI


#include "Opos.hi"
#include "OposMsr.h"


//////////////////////////////////////////////////////////////////////
// Numeric Property Index Values.
//////////////////////////////////////////////////////////////////////

// * Properties *

const LONG PIDXMsr_DecodeData          =   1 + PIDX_MSR+PIDX_NUMBER;
const LONG PIDXMsr_ParseDecodeData     =   2 + PIDX_MSR+PIDX_NUMBER;
const LONG PIDXMsr_TracksToRead        =   3 + PIDX_MSR+PIDX_NUMBER;

//      Added for Release 1.2:
const LONG PIDXMsr_ParseDecodedData    =   2 + PIDX_MSR+PIDX_NUMBER;
//        ParseDecodedData = ParseDecodeData: Support both, due to
//        editing error in the pre-1.2 APG.
const LONG PIDXMsr_ErrorReportingType  =   4 + PIDX_MSR+PIDX_NUMBER;

//      Added for Release 1.5:
```

```
const LONG PIDXMsr_TransmitSentinels    =   5 + PIDX_MSR+PIDX_NUMBER;

// * Capabilities *

const LONG PIDXMsr_CapISO               = 501 + PIDX_MSR+PIDX_NUMBER;
const LONG PIDXMsr_CapJISOne            = 502 + PIDX_MSR+PIDX_NUMBER;
const LONG PIDXMsr_CapJISTwo            = 503 + PIDX_MSR+PIDX_NUMBER;

//      Added for Release 1.5:
const LONG PIDXMsr_CapTransmitSentinels = 504 + PIDX_MSR+PIDX_NUMBER;

// * Validation Functions *

inline BOOL IsValidMsrNumericPidx(LONG Pidx)
{
  return ( PIDXMsr_DecodeData <= Pidx && Pidx <= PIDXMsr_TracksToRead )
    ? TRUE : FALSE ;
}

inline BOOL IsValidMsrNumericPidx12(LONG Pidx)
{
  return ( PIDXMsr_DecodeData <= Pidx &&
           Pidx <= PIDXMsr_ErrorReportingType )
    ? TRUE : FALSE ;
}

inline BOOL IsValidMsrNumericPidx15(LONG Pidx)
{
  return ( PIDXMsr_DecodeData <= Pidx &&
           Pidx <= PIDXMsr_TransmitSentinels )
    ? TRUE : FALSE ;
}

inline BOOL IsValidMsrCapPidx(LONG Pidx)
{
  return ( PIDXMsr_CapISO <= Pidx && Pidx <= PIDXMsr_CapJISTwo )
    ? TRUE : FALSE ;
}

inline BOOL IsValidMsrCapPidx15(LONG Pidx)
{
  return ( PIDXMsr_CapISO <= Pidx &&
           Pidx <= PIDXMsr_CapTransmitSentinels )
    ? TRUE : FALSE ;
}
```

```
//////////////////////////////////////////////////////////////////////
// String Property Index Values.
//////////////////////////////////////////////////////////////////////

// * Properties *

const LONG PIDXMsr_AccountNumber        =   1 + PIDX_MSR+PIDX_STRING;
const LONG PIDXMsr_ExpirationDate       =   2 + PIDX_MSR+PIDX_STRING;
const LONG PIDXMsr_FirstName            =   3 + PIDX_MSR+PIDX_STRING;
const LONG PIDXMsr_MiddleInitial        =   4 + PIDX_MSR+PIDX_STRING;
const LONG PIDXMsr_ServiceCode          =   5 + PIDX_MSR+PIDX_STRING;
const LONG PIDXMsr_Suffix               =   6 + PIDX_MSR+PIDX_STRING;
const LONG PIDXMsr_Surname              =   7 + PIDX_MSR+PIDX_STRING;
const LONG PIDXMsr_Title                =   8 + PIDX_MSR+PIDX_STRING;
const LONG PIDXMsr_Track1Data           =   9 + PIDX_MSR+PIDX_STRING;
const LONG PIDXMsr_Track1DiscretionaryData
                                        =  10 + PIDX_MSR+PIDX_STRING;
const LONG PIDXMsr_Track2Data           =  11 + PIDX_MSR+PIDX_STRING;
const LONG PIDXMsr_Track2DiscretionaryData
                                        =  12 + PIDX_MSR+PIDX_STRING;
const LONG PIDXMsr_Track3Data           =  13 + PIDX_MSR+PIDX_STRING;

//      Added for Release 1.5:
const LONG PIDXMsr_Track4Data           =  14 + PIDX_MSR+PIDX_STRING;

// * Validation Function *

inline BOOL IsValidMsrStringPidx(LONG Pidx)
{
  return ( PIDXMsr_AccountNumber <= Pidx &&
           Pidx <= PIDXMsr_Track3Data )
    ? TRUE : FALSE ;
}

inline BOOL IsValidMsrStringPidx15(LONG Pidx)
{
  return ( PIDXMsr_AccountNumber <= Pidx &&
           Pidx <= PIDXMsr_Track4Data )
    ? TRUE : FALSE ;
}


#endif          // !defined(OPOSMSR_HI)
```

OposPcrw.hi : Point Card Reader Writer Internal Header File

```
/////////////////////////////////////////////////////////////////////
//
// OposPcrw.hi
//
//   Point Card Reader Writer  header file for OPOS Controls and
//   Service Objects.
//
// Modification history
// ------------------------------------------------------------------
// 00-09-24 OPOS Release 1.5                                     BKS
//
/////////////////////////////////////////////////////////////////////

#if !defined(OPOSPCRW_HI)
#define      OPOSPCRW_HI


#include "Opos.hi"
#include "OposPcrw.h"


//////////////////////////////////////////////////////////////////////
// Numeric Property Index Values.
//////////////////////////////////////////////////////////////////////

// * Properties *

const LONG PIDXPcrw_CardState          =   1 + PIDX_PCRW+PIDX_NUMBER;
const LONG PIDXPcrw_CharacterSet       =   2 + PIDX_PCRW+PIDX_NUMBER;
const LONG PIDXPcrw_LineChars          =   3 + PIDX_PCRW+PIDX_NUMBER;
const LONG PIDXPcrw_LineHeight         =   4 + PIDX_PCRW+PIDX_NUMBER;
const LONG PIDXPcrw_LineSpacing        =   5 + PIDX_PCRW+PIDX_NUMBER;
const LONG PIDXPcrw_LineWidth          =   6 + PIDX_PCRW+PIDX_NUMBER;
const LONG PIDXPcrw_MapMode            =   7 + PIDX_PCRW+PIDX_NUMBER;
const LONG PIDXPcrw_MaxLine            =   8 + PIDX_PCRW+PIDX_NUMBER;
const LONG PIDXPcrw_PrintHeight        =   9 + PIDX_PCRW+PIDX_NUMBER;
const LONG PIDXPcrw_ReadState1         =  10 + PIDX_PCRW+PIDX_NUMBER;
const LONG PIDXPcrw_ReadState2         =  11 + PIDX_PCRW+PIDX_NUMBER;
const LONG PIDXPcrw_RecvLength1        =  12 + PIDX_PCRW+PIDX_NUMBER;
const LONG PIDXPcrw_RecvLength2        =  13 + PIDX_PCRW+PIDX_NUMBER;
const LONG PIDXPcrw_SidewaysMaxChars   =  14 + PIDX_PCRW+PIDX_NUMBER;
const LONG PIDXPcrw_SidewaysMaxLines   =  15 + PIDX_PCRW+PIDX_NUMBER;
const LONG PIDXPcrw_TracksToRead       =  16 + PIDX_PCRW+PIDX_NUMBER;
const LONG PIDXPcrw_TracksToWrite       =  17 + PIDX_PCRW+PIDX_NUMBER;
const LONG PIDXPcrw_WriteState1         =  18 + PIDX_PCRW+PIDX_NUMBER;
const LONG PIDXPcrw_WriteState2         =  19 + PIDX_PCRW+PIDX_NUMBER;


// * Capabilities *

const LONG PIDXPcrw_CapBold                = 501 + PIDX_PCRW+PIDX_NUMBER;
const LONG PIDXPcrw_CapCardEntranceSensor = 502 + PIDX_PCRW+PIDX_NUMBER;
const LONG PIDXPcrw_CapCharacterSet       = 503 + PIDX_PCRW+PIDX_NUMBER;
const LONG PIDXPcrw_CapCleanCard          = 504 + PIDX_PCRW+PIDX_NUMBER;
const LONG PIDXPcrw_CapClearPrint         = 505 + PIDX_PCRW+PIDX_NUMBER;
const LONG PIDXPcrw_CapDhigh              = 506 + PIDX_PCRW+PIDX_NUMBER;
const LONG PIDXPcrw_CapDwide              = 507 + PIDX_PCRW+PIDX_NUMBER;
const LONG PIDXPcrw_CapDwideDhigh         = 508 + PIDX_PCRW+PIDX_NUMBER;
const LONG PIDXPcrw_CapItalic             = 509 + PIDX_PCRW+PIDX_NUMBER;
const LONG PIDXPcrw_CapLeft90             = 510 + PIDX_PCRW+PIDX_NUMBER;
const LONG PIDXPcrw_CapPrint              = 511 + PIDX_PCRW+PIDX_NUMBER;
const LONG PIDXPcrw_CapPrintMode          = 512 + PIDX_PCRW+PIDX_NUMBER;
const LONG PIDXPcrw_CapRight90            = 513 + PIDX_PCRW+PIDX_NUMBER;
const LONG PIDXPcrw_CapRotate180          = 514 + PIDX_PCRW+PIDX_NUMBER;
const LONG PIDXPcrw_CapTracksToRead       = 515 + PIDX_PCRW+PIDX_NUMBER;
const LONG PIDXPcrw_CapTracksToWrite      = 516 + PIDX_PCRW+PIDX_NUMBER;


// * Validation Functions *

inline BOOL IsValidPcrwNumericPidx(LONG Pidx)
{
  return ( PIDXPcrw_CardState <= Pidx &&
           Pidx <= PIDXPcrw_WriteState2 )
    ? TRUE : FALSE ;
}

inline BOOL IsValidPcrwNumericPidx15(LONG Pidx)
{
  return IsValidPcrwNumericPidx(Pidx);
}

inline BOOL IsValidPcrwCapPidx(LONG Pidx)
{
  return ( PIDXPcrw_CapBold <= Pidx &&
           Pidx <= PIDXPcrw_CapTracksToWrite )
    ? TRUE : FALSE ;
}
```

```
inline BOOL IsValidPcrwCapPidx15(LONG Pidx)
{
  return IsValidPcrwCapPidx(Pidx);
}


//////////////////////////////////////////////////////////////////////
// String Property Index Values.
//////////////////////////////////////////////////////////////////////

// * Properties *

const LONG PIDXPcrw_CharacterSetList    =  1 + PIDX_PCRW+PIDX_STRING;
const LONG PIDXPcrw_FontTypeFaceList    =  2 + PIDX_PCRW+PIDX_STRING;
const LONG PIDXPcrw_LineCharsList       =  3 + PIDX_PCRW+PIDX_STRING;
const LONG PIDXPcrw_Track1Data          =  4 + PIDX_PCRW+PIDX_STRING;
const LONG PIDXPcrw_Track2Data          =  5 + PIDX_PCRW+PIDX_STRING;
const LONG PIDXPcrw_Track3Data          =  6 + PIDX_PCRW+PIDX_STRING;
const LONG PIDXPcrw_Track4Data          =  7 + PIDX_PCRW+PIDX_STRING;
const LONG PIDXPcrw_Track5Data          =  8 + PIDX_PCRW+PIDX_STRING;
const LONG PIDXPcrw_Track6Data          =  9 + PIDX_PCRW+PIDX_STRING;
const LONG PIDXPcrw_Write1Data          = 10 + PIDX_PCRW+PIDX_STRING;
const LONG PIDXPcrw_Write2Data          = 11 + PIDX_PCRW+PIDX_STRING;
const LONG PIDXPcrw_Write3Data          = 12 + PIDX_PCRW+PIDX_STRING;
const LONG PIDXPcrw_Write4Data          = 13 + PIDX_PCRW+PIDX_STRING;
const LONG PIDXPcrw_Write5Data          = 14 + PIDX_PCRW+PIDX_STRING;
const LONG PIDXPcrw_Write6Data          = 15 + PIDX_PCRW+PIDX_STRING;


// * Validation Functions *

inline BOOL IsValidPcrwStringPidx(LONG Pidx)
{
  return ( PIDXPcrw_CharacterSetList <= Pidx &&
           Pidx <= PIDXPcrw_Write6Data )
    ? TRUE : FALSE ;
}

inline BOOL IsValidPcrwStringPidx15(LONG Pidx)
{
  return IsValidPcrwStringPidx(Pidx);
}


#endif          // !defined(OPOSPCRW_HI)
```

OposPpad.hi: PIN Pad Internal Header File

```
//////////////////////////////////////////////////////////////////////
//
// OposPpad.hi
//
//   PIN Pad header file for OPOS Controls and Service Objects.
//
// Modification history
// ------------------------------------------------------------------
// 98-04-07 OPOS Release 1.3                                    JDB
// 99-12-xx OPOS Release 1.5                                    CRM
//   Remove Amount property index Amount is of type CURRENCY,
//     so it is not accessed by Get/SetPropertyXxx.
// 00-09-24 OPOS Release 1.5                                    BKS
//  Added the Track4Data property
//
//////////////////////////////////////////////////////////////////////

#if !defined(OPOSPPAD_HI)
#define      OPOSPPAD_HI


#include "Opos.hi"
#include "OposPpad.h"


//////////////////////////////////////////////////////////////////////
// Numeric Property Index Values.
//////////////////////////////////////////////////////////////////////

// * Properties *

const LONG PIDXPpad_MaximumPINLength    =   1 + PIDX_PPAD+PIDX_NUMBER;
const LONG PIDXPpad_MinimumPINLength    =   2 + PIDX_PPAD+PIDX_NUMBER;
const LONG PIDXPpad_PINEntryEnabled     =   3 + PIDX_PPAD+PIDX_NUMBER;
const LONG PIDXPpad_Prompt              =   4 + PIDX_PPAD+PIDX_NUMBER;
const LONG PIDXPpad_PromptLanguage      =   5 + PIDX_PPAD+PIDX_NUMBER;
const LONG PIDXPpad_TransactionType     =   6 + PIDX_PPAD+PIDX_NUMBER;


// * Capabilities *
const LONG PIDXPpad_CapDisplay          = 501 + PIDX_PPAD+PIDX_NUMBER;
const LONG PIDXPpad_CapKeyboard         = 502 + PIDX_PPAD+PIDX_NUMBER;
const LONG PIDXPpad_CapLanguage         = 503 + PIDX_PPAD+PIDX_NUMBER;
const LONG PIDXPpad_CapMACCalculation   = 504 + PIDX_PPAD+PIDX_NUMBER;
const LONG PIDXPpad_CapTone             = 505 + PIDX_PPAD+PIDX_NUMBER;


// * Validation Functions *

inline BOOL IsValidPpadNumericPidx(LONG Pidx)
{
  return ( PIDXPpad_MaximumPINLength <= Pidx &&
           Pidx <= PIDXPpad_TransactionType )
    ? TRUE : FALSE ;
}

inline BOOL IsValidPpadNumericPidx13(LONG Pidx)
{
  return IsValidPpadNumericPidx(Pidx);
}

inline BOOL IsValidPpadCapPidx(LONG Pidx)
{
  return ( PIDXPpad_CapDisplay <= Pidx &&
           Pidx <= PIDXPpad_CapTone )
    ? TRUE : FALSE ;
}

inline BOOL IsValidPpadCapPidx13(LONG Pidx)
{
  return IsValidPpadCapPidx(Pidx);
}


//////////////////////////////////////////////////////////////////////
// String Property Index Values.
//////////////////////////////////////////////////////////////////////

// * Properties *

const LONG PIDXPpad_AccountNumber       =   1 + PIDX_PPAD+PIDX_STRING;
const LONG PIDXPpad_AdditionalSecurityInformation
                                        =   2 + PIDX_PPAD+PIDX_STRING;
const LONG PIDXPpad_AvailableLanguagesList
                                        =   3 + PIDX_PPAD+PIDX_STRING;
//  (Unused)                                4 + PIDX_PPAD+PIDX_STRING;
const LONG PIDXPpad_AvailablePromptsList=   5 + PIDX_PPAD+PIDX_STRING;
const LONG PIDXPpad_EncryptedPIN        =   6 + PIDX_PPAD+PIDX_STRING;
const LONG PIDXPpad_MerchantID          =   7 + PIDX_PPAD+PIDX_STRING;
const LONG PIDXPpad_TerminalID         =   8 + PIDX_PPAD+PIDX_STRING;
const LONG PIDXPpad_Track1Data         =   9 + PIDX_PPAD+PIDX_STRING;
const LONG PIDXPpad_Track2Data         =  10 + PIDX_PPAD+PIDX_STRING;
const LONG PIDXPpad_Track3Data         =  11 + PIDX_PPAD+PIDX_STRING;

// Added in Release 1,5
const LONG PIDXPpad_Track4Data         =  12 + PIDX_PPAD+PIDX_STRING;

// * Validation Functions *

inline BOOL IsValidPpadStringPidx(LONG Pidx)
{
  return ( PIDXPpad_AccountNumber <= Pidx &&
           Pidx <= PIDXPpad_Track3Data )
    ? TRUE : FALSE ;
}

inline BOOL IsValidPpadStringPidx13(LONG Pidx)
{
  return IsValidPpadStringPidx(Pidx);
}

inline BOOL IsValidPpadStringPidx15(LONG Pidx)
{
  return ( PIDXPpad_AccountNumber <= Pidx &&
           Pidx <= PIDXPpad_Track4Data )
    ? TRUE : FALSE ;
}


#endif          // !defined(OPOSPPAD_HI)
```

OposPtr.hi : POS Printer Internal Header File

```
/////////////////////////////////////////////////////////////////////
//
// OposPtr.hi
//
//   POS Printer header file for OPOS Controls and Service Objects.
//
// Modification history
// ------------------------------------------------------------------
// 95-12-08 OPOS Release 1.0                                     CRM
// 96-03-18 OPOS Release 1.01                                    CRM
//   Change ...Nearend to ...NearEnd.
//   Change ...Barcode to ...BarCode.
//   Corrected IsValidPtrNumericPidx function.
// 96-04-22 OPOS Release 1.1                                     CRM
//   Add the following properties:
//     CapCharacterSet, CapTransaction, ErrorLevel, RotateSpecial,
//     ErrorString, FontTypefaceList, RecBarCodeRotationList,
//     SlpBarCodeRotationList
// 00-09-24 OPOS Release 1.5                                     BKS
//   Add the following properties:
//     CapJrnCartridgeSensor, CapJrnColor, CapRecCartrdigeSensor,
//     CapRecColor, CapRecMarkFeed, CapSlpBothSidesPrint,
//     CapSlpCartridgeSensor, CapSlpColor, CartridgeNotify,
//     JrnCartridgeState, JrnCurrentCartridge, RecCartridgeState,
//     RecCurrentCartridge, SlpPrintSide, SlpCartridgeState,
//     SlpCurrentCartridge
//
/////////////////////////////////////////////////////////////////////

#if !defined(OPOSPTR_HI)
#define      OPOSPTR_HI


#include "Opos.hi"
#include "OposPtr.h"


/////////////////////////////////////////////////////////////////////
// Numeric Property Index Values.
/////////////////////////////////////////////////////////////////////

// * Properties *
```

```
const LONG PIDXPtr_AsyncMode           =  1 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_CharacterSet        =  2 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_CoverOpen           =  3 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_ErrorStation        =  4 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_FlagWhenIdle        =  5 + PIDX_PTR+PIDX_NUMBER;

const LONG PIDXPtr_JrnEmpty            =  6 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_JrnLetterQuality    =  7 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_JrnLineChars        =  8 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_JrnLineHeight       =  9 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_JrnLineSpacing      = 10 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_JrnLineWidth        = 11 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_JrnNearEnd          = 12 + PIDX_PTR+PIDX_NUMBER;

const LONG PIDXPtr_MapMode             = 13 + PIDX_PTR+PIDX_NUMBER;

const LONG PIDXPtr_RecEmpty            = 14 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_RecLetterQuality    = 15 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_RecLineChars        = 16 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_RecLineHeight       = 17 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_RecLineSpacing      = 18 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_RecLinesToPaperCut  = 19 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_RecLineWidth        = 20 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_RecNearEnd          = 21 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_RecSidewaysMaxChars = 22 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_RecSidewaysMaxLines = 23 + PIDX_PTR+PIDX_NUMBER;

const LONG PIDXPtr_SlpEmpty            = 24 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_SlpLetterQuality    = 25 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_SlpLineChars        = 26 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_SlpLineHeight       = 27 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_SlpLinesNearEndToEnd = 28 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_SlpLineSpacing      = 29 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_SlpLineWidth        = 30 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_SlpMaxLines         = 31 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_SlpNearEnd          = 32 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_SlpSidewaysMaxChars = 33 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_SlpSidewaysMaxLines = 34 + PIDX_PTR+PIDX_NUMBER;

//      Added for Release 1.1:
const LONG PIDXPtr_ErrorLevel          = 35 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_RotateSpecial       = 36 + PIDX_PTR+PIDX_NUMBER;

//      Added for Release 1.5:
const LONG PIDXPtr_CartridgeNotify     = 37 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_JrnCartridgeState   = 38 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_JrnCurrentCartridge  =  39 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_RecCartridgeState    =  40 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_RecCurrentCartridge  =  41 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_SlpPrintSide         =  42 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_SlpCartridgeState    =  43 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_SlpCurrentCartridge  =  44 + PIDX_PTR+PIDX_NUMBER;


// * Capabilities *

const LONG PIDXPtr_CapConcurrentJrnRec  = 501 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_CapConcurrentJrnSlp  = 502 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_CapConcurrentRecSlp  = 503 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_CapCoverSensor       = 504 + PIDX_PTR+PIDX_NUMBER;

const LONG PIDXPtr_CapJrn2Color         = 505 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_CapJrnBold           = 506 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_CapJrnDhigh          = 507 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_CapJrnDwide          = 508 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_CapJrnDwideDhigh     = 509 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_CapJrnEmptySensor    = 510 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_CapJrnItalic         = 511 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_CapJrnNearEndSensor  = 512 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_CapJrnPresent        = 513 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_CapJrnUnderline      = 514 + PIDX_PTR+PIDX_NUMBER;

const LONG PIDXPtr_CapRec2Color         = 515 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_CapRecBarCode        = 516 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_CapRecBitmap         = 517 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_CapRecBold           = 518 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_CapRecDhigh          = 519 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_CapRecDwide          = 520 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_CapRecDwideDhigh     = 521 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_CapRecEmptySensor    = 522 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_CapRecItalic         = 523 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_CapRecLeft90         = 524 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_CapRecNearEndSensor  = 525 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_CapRecPapercut       = 526 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_CapRecPresent        = 527 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_CapRecRight90        = 528 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_CapRecRotate180      = 529 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_CapRecStamp          = 530 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_CapRecUnderline      = 531 + PIDX_PTR+PIDX_NUMBER;

const LONG PIDXPtr_CapSlp2Color         = 532 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_CapSlpBarCode        = 533 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_CapSlpBitmap         = 534 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_CapSlpBold           = 535 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_CapSlpDhigh          = 536 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_CapSlpDwide          = 537 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_CapSlpDwideDhigh     = 538 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_CapSlpEmptySensor    = 539 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_CapSlpFullslip       = 540 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_CapSlpItalic         = 541 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_CapSlpLeft90         = 542 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_CapSlpNearEndSensor  = 543 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_CapSlpPresent        = 544 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_CapSlpRight90        = 545 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_CapSlpRotate180      = 546 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_CapSlpUnderline      = 547 + PIDX_PTR+PIDX_NUMBER;

//      Added for Release 1.1:
const LONG PIDXPtr_CapCharacterSet      = 548 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_CapTransaction       = 549 + PIDX_PTR+PIDX_NUMBER;

//      Added for Release 1.5:
const LONG PIDXPtr_CapJrnCartridgeSensor
                                        = 550 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_CapJrnColor          = 551 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_CapRecCartridgeSensor
                                        = 552 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_CapRecColor          = 553 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_CapRecMarkFeed       = 554 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_CapSlpBothSidesPrint = 555 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_CapSlpCartridgeSensor
                                        = 556 + PIDX_PTR+PIDX_NUMBER;
const LONG PIDXPtr_CapSlpColor          = 557 + PIDX_PTR+PIDX_NUMBER;


// * Validation Functions *

inline BOOL IsValidPtrNumericPidx(LONG Pidx)
{
  return ( PIDXPtr_AsyncMode <= Pidx &&
           Pidx <= PIDXPtr_SlpSidewaysMaxLines )
    ? TRUE : FALSE ;
}

inline BOOL IsValidPtrNumericPidx11(LONG Pidx)
{
  return ( PIDXPtr_AsyncMode <= Pidx &&
           Pidx <= PIDXPtr_RotateSpecial )
    ? TRUE : FALSE ;
}

inline BOOL IsValidPtrNumericPidx15(LONG Pidx)
{
  return ( PIDXPtr_AsyncMode <= Pidx &&
           Pidx <= PIDXPtr_SlpCurrentCartridge )
    ? TRUE : FALSE ;
}

inline BOOL IsValidPtrCapPidx(LONG Pidx)
{
  return ( PIDXPtr_CapConcurrentJrnRec <= Pidx &&
           Pidx <= PIDXPtr_CapSlpUnderline )
    ? TRUE : FALSE ;
}

inline BOOL IsValidPtrCapPidx11(LONG Pidx)
{
  return ( PIDXPtr_CapConcurrentJrnRec <= Pidx &&
           Pidx <= PIDXPtr_CapTransaction )
    ? TRUE : FALSE ;
}

inline BOOL IsValidPtrCapPidx15(LONG Pidx)
{
  return ( PIDXPtr_CapConcurrentJrnRec <= Pidx &&
           Pidx <= PIDXPtr_CapSlpColor )
    ? TRUE : FALSE ;
}


//////////////////////////////////////////////////////////////////////
// String Property Index Values.
//////////////////////////////////////////////////////////////////////

// * Properties *

const LONG PIDXPtr_CharacterSetList     =   1 + PIDX_PTR+PIDX_STRING;
const LONG PIDXPtr_JrnLineCharsList     =   2 + PIDX_PTR+PIDX_STRING;
const LONG PIDXPtr_RecLineCharsList     =   3 + PIDX_PTR+PIDX_STRING;
const LONG PIDXPtr_SlpLineCharsList     =   4 + PIDX_PTR+PIDX_STRING;

//      Added for Release 1.1:
const LONG PIDXPtr_ErrorString          =   5 + PIDX_PTR+PIDX_STRING;
const LONG PIDXPtr_FontTypefaceList     =   6 + PIDX_PTR+PIDX_STRING;
const LONG PIDXPtr_RecBarCodeRotationList
                                        =   7 + PIDX_PTR+PIDX_STRING;
const LONG PIDXPtr_SlpBarCodeRotationList
                                        =   8 + PIDX_PTR+PIDX_STRING;


// * Validation Functions *

inline BOOL IsValidPtrStringPidx(LONG Pidx)
{
  return ( PIDXPtr_CharacterSetList <= Pidx &&
           Pidx <= PIDXPtr_SlpLineCharsList )
    ? TRUE : FALSE ;
}

inline BOOL IsValidPtrStringPidx11(LONG Pidx)
{
  return ( PIDXPtr_CharacterSetList <= Pidx &&
           Pidx <= PIDXPtr_SlpBarCodeRotationList )
    ? TRUE : FALSE ;
}


#endif          // !defined(OPOSPTR_HI)
```

OposPwr.hi : POS Power Internal Header File

```
/////////////////////////////////////////////////////////////////////
//
// OposPwr.hi
//
//   POS Power header file for OPOS Controls and Service Objects.
//
// Modification history
// -------------------------------------------------------------------
// 00-09-24 OPOS Release 1.5                                      BKS
//
/////////////////////////////////////////////////////////////////////

#if !defined(OPOSPWR_HI)
#define      OPOSPWR_HI


#include "Opos.hi"
#include "OposPwr.h"


//////////////////////////////////////////////////////////////////////
// Numeric Property Index Values.
//////////////////////////////////////////////////////////////////////

// * Properties *

const LONG PIDXPwr_EnforcedShutdownDelayTime
                                         =   1 + PIDX_PWR+PIDX_NUMBER;
const LONG PIDXPwr_PowerFailDelayTime    =   2 + PIDX_PWR+PIDX_NUMBER;
const LONG PIDXPwr_QuickChargeMode       =   3 + PIDX_PWR+PIDX_NUMBER;
const LONG PIDXPwr_QuickChargeTime       =   4 + PIDX_PWR+PIDX_NUMBER;
const LONG PIDXPwr_UPSChargeState        =   5 + PIDX_PWR+PIDX_NUMBER;


// * Capabilities *

const LONG PIDXPwr_CapFanAlarm           = 501 + PIDX_PWR+PIDX_NUMBER;
const LONG PIDXPwr_CapHeatAlarm          = 502 + PIDX_PWR+PIDX_NUMBER;
const LONG PIDXPwr_CapQuickCharge        = 503 + PIDX_PWR+PIDX_NUMBER;
const LONG PIDXPwr_CapShutdownPOS        = 504 + PIDX_PWR+PIDX_NUMBER;
const LONG PIDXPwr_CapUPSChargeState     = 505 + PIDX_PWR+PIDX_NUMBER;
```

```
// * Validation Functions *

inline BOOL IsValidPwrNumericPidx(LONG Pidx)
{
  return ( PIDXPwr_EnforcedShutdownDelayTime <= Pidx &&
           Pidx <= PIDXPwr_UPSChargeState )
    ? TRUE : FALSE ;
}

inline BOOL IsValidPwrNumericPidx14(LONG Pidx)
{
  return IsValidPwrNumericPidx(Pidx);
}

inline BOOL IsValidPwrCapPidx(LONG Pidx)
{
  return ( PIDXPwr_CapFanAlarm <= Pidx &&
           Pidx <= PIDXPwr_CapUPSChargeState )
    ? TRUE : FALSE ;
}

inline BOOL IsValidPwrCapPidx15(LONG Pidx)
{
  return IsValidPwrCapPidx(Pidx);
}


//////////////////////////////////////////////////////////////////////
// String Property Index Values.
//////////////////////////////////////////////////////////////////////

// * Validation Functions *

inline BOOL IsValidPwrStringPidx(LONG Pidx)
{
  return FALSE ;
}

inline BOOL IsValidPwrStringPidx15(LONG Pidx)
{
  return IsValidPwrStringPidx(Pidx);
}


#endif          // !defined(OPOSPWR_HI)
```

OposRod.hi: Remote Order Display Internal Header File

```
//////////////////////////////////////////////////////////////////////
//
// OposRod.hi
//
//   Remote Order Display header file for OPOS Controls and
//     Service Objects.
//
// Modification history
// -------------------------------------------------------------------
// 98-03-06 OPOS Release 1.3                                     CRM
//
//////////////////////////////////////////////////////////////////////

#if !defined(OPOSROD_HI)
#define      OPOSROD_HI


#include "Opos.hi"
#include "OposRod.h"


//////////////////////////////////////////////////////////////////////
// Numeric Property Index Values.
//////////////////////////////////////////////////////////////////////

// * Properties *

const LONG PIDXRod_AsyncMode            =   1 + PIDX_ROD+PIDX_NUMBER;
const LONG PIDXRod_AutoToneDuration     =   2 + PIDX_ROD+PIDX_NUMBER;
const LONG PIDXRod_AutoToneFrequency    =   3 + PIDX_ROD+PIDX_NUMBER;
const LONG PIDXRod_CharacterSet         =   4 + PIDX_ROD+PIDX_NUMBER;
const LONG PIDXRod_Clocks               =   5 + PIDX_ROD+PIDX_NUMBER;
const LONG PIDXRod_CurrentUnitID        =   6 + PIDX_ROD+PIDX_NUMBER;
const LONG PIDXRod_ErrorUnits           =   7 + PIDX_ROD+PIDX_NUMBER;
const LONG PIDXRod_EventType            =   8 + PIDX_ROD+PIDX_NUMBER;
const LONG PIDXRod_EventUnitID          =   9 + PIDX_ROD+PIDX_NUMBER;
const LONG PIDXRod_EventUnits           =  10 + PIDX_ROD+PIDX_NUMBER;
const LONG PIDXRod_SystemClocks         =  11 + PIDX_ROD+PIDX_NUMBER;
const LONG PIDXRod_SystemVideoSaveBuffers
                                        =  12 + PIDX_ROD+PIDX_NUMBER;
const LONG PIDXRod_Timeout              =  13 + PIDX_ROD+PIDX_NUMBER;
const LONG PIDXRod_UnitsOnline          =  14 + PIDX_ROD+PIDX_NUMBER;
const LONG PIDXRod_VideoDataCount       =  15 + PIDX_ROD+PIDX_NUMBER;
const LONG PIDXRod_VideoMode            =  16 + PIDX_ROD+PIDX_NUMBER;
const LONG PIDXRod_VideoSaveBuffers     =  17 + PIDX_ROD+PIDX_NUMBER;

// * Capabilities *

const LONG PIDXRod_CapSelectCharacterSet
                                        = 501 + PIDX_ROD+PIDX_NUMBER;
const LONG PIDXRod_CapTone              = 502 + PIDX_ROD+PIDX_NUMBER;
const LONG PIDXRod_CapTouch             = 503 + PIDX_ROD+PIDX_NUMBER;
const LONG PIDXRod_CapTransaction       = 504 + PIDX_ROD+PIDX_NUMBER;

// * Validation Functions *

inline BOOL IsValidRodNumericPidx(LONG Pidx)
{
  return ( PIDXRod_AsyncMode <= Pidx &&
           Pidx <= PIDXRod_VideoSaveBuffers )
    ? TRUE : FALSE ;
}

inline BOOL IsValidRodNumericPidx13(LONG Pidx)
{
  return IsValidRodNumericPidx(Pidx);
}

inline BOOL IsValidRodCapPidx(LONG Pidx)
{
  return ( PIDXRod_CapSelectCharacterSet <= Pidx &&
           Pidx <= PIDXRod_CapTransaction )
    ? TRUE : FALSE ;
}

inline BOOL IsValidRodCapPidx13(LONG Pidx)
{
  return IsValidRodCapPidx(Pidx);
}


//////////////////////////////////////////////////////////////////////
// String Property Index Values.
//////////////////////////////////////////////////////////////////////

// * Properties *

const LONG PIDXRod_CharacterSetList     =   1 + PIDX_ROD+PIDX_STRING;
const LONG PIDXRod_ErrorString          =   2 + PIDX_ROD+PIDX_STRING;
const LONG PIDXRod_EventString          =   3 + PIDX_ROD+PIDX_STRING;
const LONG PIDXRod_VideoModesList       =   4 + PIDX_ROD+PIDX_STRING;

// * Validation Function *

inline BOOL IsValidRodStringPidx(LONG Pidx)
{
  return ( PIDXRod_CharacterSetList <= Pidx &&
           Pidx <= PIDXRod_VideoModesList )
    ? TRUE : FALSE ;
}

inline BOOL IsValidRodStringPidx13(LONG Pidx)
{
  return IsValidRodStringPidx(Pidx);
}


#endif          // !defined(OPOSROD_HI)
```

OposScal.hi : Scale Internal Header File

```
//////////////////////////////////////////////////////////////////////
//
// OposScal.hi
//
//   Scale header file for OPOS Controls and Service Objects.
//
// Modification history
// -------------------------------------------------------------------
// 95-12-08 OPOS Release 1.0                                     CRM
// 96-03-18 OPOS Release 1.01                                    CRM
//   Correct WeightUnits value from 1 to 2.
// 97-06-04 OPOS Release 1.2                                     CRM
//   Add the following properties: CapDisplay, WeightUnit
// 98-03-06 OPOS Release 1.3                                     CRM
//   Add the following properties:
//     CapDisplayText, CapPriceCalculating, CapTareWeight,
//     CapZeroScale, AsyncMode, MaxDisplayTextChars, SalesPrice,
//     TareWeight, UnitPrice
//
//////////////////////////////////////////////////////////////////////

#if !defined(OPOSSCAL_HI)
#define      OPOSSCAL_HI


#include "Opos.hi"
#include "OposScal.h"


//////////////////////////////////////////////////////////////////////
// Numeric Property Index Values.
//////////////////////////////////////////////////////////////////////

// * Properties *

const LONG PIDXScal_MaximumWeight       =   1 + PIDX_SCAL+PIDX_NUMBER;
const LONG PIDXScal_WeightUnits         =   2 + PIDX_SCAL+PIDX_NUMBER;

//      Added for Release 1.2:
const LONG PIDXScal_WeightUnit          =   2 + PIDX_SCAL+PIDX_NUMBER;
//        WeightUnit = WeightUnits: Support both, due to
//        editing error in the pre-1.2 APG.
//      Added for Release 1.3:
const LONG PIDXScal_AsyncMode           =   3 + PIDX_SCAL+PIDX_NUMBER;
const LONG PIDXScal_MaxDisplayTextChars =   4 + PIDX_SCAL+PIDX_NUMBER;
const LONG PIDXScal_TareWeight          =   5 + PIDX_SCAL+PIDX_NUMBER;

// * Capabilities *

//      Added for Release 1.1:
const LONG PIDXScal_CapDisplay          = 501 + PIDX_SCAL+PIDX_NUMBER;

//      Added for Release 1.3:
const LONG PIDXScal_CapDisplayText      = 502 + PIDX_SCAL+PIDX_NUMBER;
const LONG PIDXScal_CapPriceCalculating = 503 + PIDX_SCAL+PIDX_NUMBER;
const LONG PIDXScal_CapTareWeight       = 504 + PIDX_SCAL+PIDX_NUMBER;
const LONG PIDXScal_CapZeroScale        = 505 + PIDX_SCAL+PIDX_NUMBER;

// * Validation Functions *

inline BOOL IsValidScalNumericPidx(LONG Pidx)
{
  return ( PIDXScal_MaximumWeight <= Pidx &&
           Pidx <= PIDXScal_WeightUnits )
    ? TRUE : FALSE ;
}

inline BOOL IsValidScalNumericPidx13(LONG Pidx)
{
  return ( PIDXScal_MaximumWeight <= Pidx &&
           Pidx <= PIDXScal_TareWeight )
    ? TRUE : FALSE ;
}

inline BOOL IsValidScalCapPidx(LONG Pidx)
{
  return FALSE;
}

inline BOOL IsValidScalCapPidx12(LONG Pidx)
{
  return ( PIDXScal_CapDisplay == Pidx )
    ? TRUE : FALSE ;
}

inline BOOL IsValidScalCapPidx13(LONG Pidx)
{
  return ( PIDXScal_CapDisplay <= Pidx &&
           Pidx <= PIDXScal_CapZeroScale )
    ? TRUE : FALSE ;
}


//////////////////////////////////////////////////////////////////////
// String Property Index Values.
//////////////////////////////////////////////////////////////////////

// * Validation Function *

inline BOOL IsValidScalStringPidx(LONG Pidx)
{
  return FALSE;
}


#endif          // !defined(OPOSSCAL_HI)
```

OposScan.hi : Bar Code Scanner Internal Header File

```
//////////////////////////////////////////////////////////////////////
//
// OposScan.hi
//
//   Scanner header file for OPOS Controls and Service Objects.
//
// Modification history
// -------------------------------------------------------------------
// 95-12-08 OPOS Release 1.0                                      CRM
// 97-06-04 OPOS Release 1.2                                      CRM
//   Add the following properties:
//     DecodeData, ScanDataType, ScanDataLabel
//
//////////////////////////////////////////////////////////////////////

#if !defined(OPOSSCAN_HI)
#define      OPOSSCAN_HI


#include "Opos.hi"
#include "OposScan.h"


//////////////////////////////////////////////////////////////////////
// Numeric Property Index Values.
//////////////////////////////////////////////////////////////////////

// * Properties *

//      Added for Release 1.2:
const LONG PIDXScan_DecodeData        =   1 + PIDX_SCAN+PIDX_NUMBER;
const LONG PIDXScan_ScanDataType      =   2 + PIDX_SCAN+PIDX_NUMBER;

// * Validation Functions *

inline BOOL IsValidScanNumericPidx(LONG Pidx)
{
  return FALSE;
}

inline BOOL IsValidScanNumericPidx12(LONG Pidx)
{
  return ( PIDXScan_DecodeData <= Pidx &&
           Pidx <= PIDXScan_ScanDataType )
    ? TRUE : FALSE ;
}

inline BOOL IsValidScanCapPidx(LONG Pidx)
{
  return FALSE;
}


//////////////////////////////////////////////////////////////////////
// String Property Index Values.
//////////////////////////////////////////////////////////////////////

// * Property *

const LONG PIDXScan_ScanData            =   1 + PIDX_SCAN+PIDX_STRING;

//      Added for Release 1.2:
const LONG PIDXScan_ScanDataLabel       =   2 + PIDX_SCAN+PIDX_STRING;

// * Validation Functions *

inline BOOL IsValidScanStringPidx(LONG Pidx)
{
  return ( PIDXScan_ScanData == Pidx )
    ? TRUE : FALSE ;
}

inline BOOL IsValidScanStringPidx12(LONG Pidx)
{
  return ( PIDXScan_ScanData <= Pidx &&
           Pidx <= PIDXScan_ScanDataLabel )
    ? TRUE : FALSE ;
}


#endif          // !defined(OPOSSCAN_HI)
```

OposSig.hi : Signature Capture Internal Header File

```
/////////////////////////////////////////////////////////////////////
//
// OposSig.hi
//
//   Signature Capture header file for OPOS Controls and
//     Service Objects.
//
// Modification history
// ------------------------------------------------------------------
// 95-12-08 OPOS Release 1.0                                     CRM
// 96-03-18 OPOS Release 1.01                                    CRM
//   Change TotalVectors to TotalPoints.
//   Change VectorArray to PointArray.
// 97-06-04 OPOS Release 1.2                                     CRM
//   Add the following properties: CapRealTimeData, RealTimeDataEnabled
//   Correct spelling of TotalPoitns to TotalPoints.
//
/////////////////////////////////////////////////////////////////////

#if !defined(OPOSSIG_HI)
#define      OPOSSIG_HI


#include "Opos.hi"
#include "OposSig.h"


/////////////////////////////////////////////////////////////////////
// Numeric Property Index Values.
/////////////////////////////////////////////////////////////////////

// * Properties *

const LONG PIDXSig_MaximumX            =   1 + PIDX_SIG+PIDX_NUMBER;
const LONG PIDXSig_MaximumY            =   2 + PIDX_SIG+PIDX_NUMBER;
const LONG PIDXSig_TotalPoints         =   3 + PIDX_SIG+PIDX_NUMBER;

//      Added for Release 1.2:
const LONG PIDXSig_RealTimeDataEnabled =   4 + PIDX_SIG+PIDX_NUMBER;

// * Capabilities *

const LONG PIDXSig_CapDisplay          = 501 + PIDX_SIG+PIDX_NUMBER;
const LONG PIDXSig_CapUserTerminated    = 502 + PIDX_SIG+PIDX_NUMBER;

//      Added for Release 1.2:
const LONG PIDXSig_CapRealTimeData      = 503 + PIDX_SIG+PIDX_NUMBER;

// * Validation Functions *

inline BOOL IsValidSigNumericPidx(LONG Pidx)
{
  return ( PIDXSig_MaximumX <= Pidx && Pidx <= PIDXSig_TotalPoints )
    ? TRUE : FALSE ;
}

inline BOOL IsValidSigNumericPidx12(LONG Pidx)
{
  return ( PIDXSig_MaximumX <= Pidx &&
           Pidx <= PIDXSig_RealTimeDataEnabled )
    ? TRUE : FALSE ;
}

inline BOOL IsValidSigCapPidx(LONG Pidx)
{
  return ( PIDXSig_CapDisplay <= Pidx &&
           Pidx <= PIDXSig_CapUserTerminated )
    ? TRUE : FALSE ;
}

inline BOOL IsValidSigCapPidx12(LONG Pidx)
{
  return ( PIDXSig_CapDisplay <= Pidx &&
           Pidx <= PIDXSig_CapRealTimeData )
    ? TRUE : FALSE ;
}


//////////////////////////////////////////////////////////////////////
// String Property Index Values.
//////////////////////////////////////////////////////////////////////

// * Properties *

const LONG PIDXSig_RawData              =   1 + PIDX_SIG+PIDX_STRING;
const LONG PIDXSig_PointArray           =   2 + PIDX_SIG+PIDX_STRING;

// * Validation Function *
```

```
inline BOOL IsValidSigStringPidx(LONG Pidx)
{
  return ( PIDXSig_RawData <= Pidx &&
           Pidx <= PIDXSig_PointArray )
    ? TRUE : FALSE ;
}


#endif          // !defined(OPOSSIG_HI)
```

OposTone.hi : Tone Indicator Internal Header File

```
/////////////////////////////////////////////////////////////////////
//
// OposTone.hi
//
//   Tone Indicator header file for OPOS Controls and Service Objects.
//
// Modification history
// ------------------------------------------------------------------
// 97-06-04 OPOS Release 1.2                                     CRM
//
/////////////////////////////////////////////////////////////////////

#if !defined(OPOSTONE_HI)
#define      OPOSTONE_HI


#include "Opos.hi"
#include "OposTone.h"


/////////////////////////////////////////////////////////////////////
// Numeric Property Index Values.
/////////////////////////////////////////////////////////////////////

// * Properties *

const LONG PIDXTone_AsyncMode          =   1 + PIDX_TONE+PIDX_NUMBER;
const LONG PIDXTone_Tone1Pitch         =   2 + PIDX_TONE+PIDX_NUMBER;
const LONG PIDXTone_Tone1Volume        =   3 + PIDX_TONE+PIDX_NUMBER;
const LONG PIDXTone_Tone1Duration      =   4 + PIDX_TONE+PIDX_NUMBER;
const LONG PIDXTone_Tone2Pitch         =   5 + PIDX_TONE+PIDX_NUMBER;
const LONG PIDXTone_Tone2Volume        =   6 + PIDX_TONE+PIDX_NUMBER;
const LONG PIDXTone_Tone2Duration      =   7 + PIDX_TONE+PIDX_NUMBER;
const LONG PIDXTone_InterToneWait      =   8 + PIDX_TONE+PIDX_NUMBER;

// * Capabilities *

const LONG PIDXTone_CapPitch           = 501 + PIDX_TONE+PIDX_NUMBER;
const LONG PIDXTone_CapVolume          = 502 + PIDX_TONE+PIDX_NUMBER;

// * Validation Functions *

inline BOOL IsValidToneNumericPidx(LONG Pidx)
```

```
{
  return ( PIDXTone_AsyncMode <= Pidx &&
           Pidx <= PIDXTone_InterToneWait )
    ? TRUE : FALSE ;
}

inline BOOL IsValidToneNumericPidx12(LONG Pidx)
{
  return IsValidToneNumericPidx(Pidx);
}

inline BOOL IsValidToneCapPidx(LONG Pidx)
{
  return ( PIDXTone_CapPitch <= Pidx &&
           Pidx <= PIDXTone_CapVolume )
    ? TRUE : FALSE ;
}

inline BOOL IsValidToneCapPidx12(LONG Pidx)
{
  return IsValidToneCapPidx(Pidx);
}


//////////////////////////////////////////////////////////////////////
// String Property Index Values.
//////////////////////////////////////////////////////////////////////

// * Validation Function *

inline BOOL IsValidToneStringPidx(LONG Pidx)
{
  return FALSE ;
}

inline BOOL IsValidToneStringPidx12(LONG Pidx)
{
  return IsValidToneStringPidx(Pidx);
}


#endif          // !defined(OPOSTONE_HI)
```

OposTot.hi : Hard Totals Internal Header File

```
//////////////////////////////////////////////////////////////////////
//
// OposTot.hi
//
//   Hard Totals header file for OPOS Controls and Service Objects.
//
// Modification history
// -------------------------------------------------------------------
// 95-12-08 OPOS Release 1.0                                     CRM
//
//////////////////////////////////////////////////////////////////////

#if !defined(OPOSTOT_HI)
#define      OPOSTOT_HI


#include "Opos.hi"
#include "OposTot.h"


//////////////////////////////////////////////////////////////////////
// Numeric Property Index Values.
//////////////////////////////////////////////////////////////////////

// * Properties *

const LONG PIDXTot_FreeData             =   1 + PIDX_TOT+PIDX_NUMBER;
const LONG PIDXTot_NumberOfFiles        =   2 + PIDX_TOT+PIDX_NUMBER;
const LONG PIDXTot_TotalsSize           =   3 + PIDX_TOT+PIDX_NUMBER;
const LONG PIDXTot_TransactionInProgress
                                        =   4 + PIDX_TOT+PIDX_NUMBER;

// * Capabilities *

const LONG PIDXTot_CapErrorDetection    = 501 + PIDX_TOT+PIDX_NUMBER;
const LONG PIDXTot_CapSingleFile        = 502 + PIDX_TOT+PIDX_NUMBER;
const LONG PIDXTot_CapTransactions      = 503 + PIDX_TOT+PIDX_NUMBER;

// * Validation Functions *

inline BOOL IsValidTotNumericPidx(LONG Pidx)
{
  return ( PIDXTot_FreeData <= Pidx &&
           Pidx <= PIDXTot_TransactionInProgress )
    ? TRUE : FALSE ;
}

inline BOOL IsValidTotCapPidx(LONG Pidx)
{
  return ( PIDXTot_CapErrorDetection <= Pidx &&
           Pidx <= PIDXTot_CapTransactions )
    ? TRUE : FALSE ;
}


//////////////////////////////////////////////////////////////////////
// String Property Index Values.
//////////////////////////////////////////////////////////////////////

// * Validation Function *

inline BOOL IsValidTotStringPidx(LONG Pidx)
{
  return FALSE;
}


#endif          // !defined(OPOSTOT_HI)
```

End of Control Guide