Appendix C  
POS for .NET Implementation Reference

## What is “POS for .NET?” **Updated in Release 1.15**

POS for .NET is a class library implementation of the UnifiedPOS Standard that provides an open device driver architecture for applications utilizing the .NET Framework to easily integrate Point-of-Service (“POS”) hardware on Microsoft Windows Operating Systems.

**Note**: Beginning with POS for .NET 1.10, the POS for .NET version number reflects the version of the version of the UnifiedPOS Specification that it conforms to. Earlier versions of POS for .Net such as POS for .NET 1.0 and POS for .NET 1.1 conform to UnifiedPOS Version 1.8. and UnifiedPOS Version 1.9 respectfully.

Microsoft will not break backwards compatibility with any documented API. Undocumented functionality, including undocumented APIs, file locations, and schemas are subject to change at any time.

The goals of POS for .NET include:

* Defining an architecture for Win32-based POS device access for the .NET Framework, while maintaining a close relationship to certain aspects of the existing OPOS implementation of the UnifiedPOS specification.
* Defining a set of POS device interfaces to support a range of POS applications that incorporate the UnifiedPOS device abstraction. The benefits of the .NET Framework extensions aid in the management of these devices.
* Provide for a migration path for legacy (existing) OPOS device services to function under the .NET Framework, albeit without the feature rich functionality that the .NET Framework potentially offers.

Deliverables available for POS for .NET in addition to this document include:

POS for .NET SDK Documentation: <https://aka.ms/p4dn-docs>

* POS for .NET Runtime and SDK: <https://aka.ms/p4dn-dl>  
  Includes: Class libraries, runtime and code samples

Additional resources for creating POS for .NET service objects from legacy OPOS services: **Updated in Release 1.11**

* A set of software middleware documentation and code, known as a “Shim”, is available that allows for developers to port their legacy OPOS service objects to run under the .NET framework, using existing OPOS naming conventions. The “Shim” is not a Microsoft supported product, does not allow for all the .NET framework benefits, but does allow for an alternative way to migrate to the POS for .NET platform with minimal code changes. A brief description is included in this appendix.

Who Should Read This Section Updated in Release 1.15

This section is intended for application developers who require access to POS-specific peripheral devices and want to implement the UnifiedPOS Standard on a POS for .NET supported Microsoft Windows Operating System. This section is also intended for a programmer who wants to write a POS for .NET Service Object (usually the device manufacturer), or an application developer who desires a better understanding of how to interface with POS for .NET.

This guide assumes that the reader is familiar with the following:

* The UnifiedPOS Device chapters in this document.
* The typical characteristics of POS peripheral devices.
* Microsoft’s .NET Framework terminology and architecture.
* A working knowledge of the OPOS Implementation Reference found in Appendix A in this document. This is helpful to give the reader special insight into the Windows based nuances of peripheral devices implemented under UnifiedPOS.

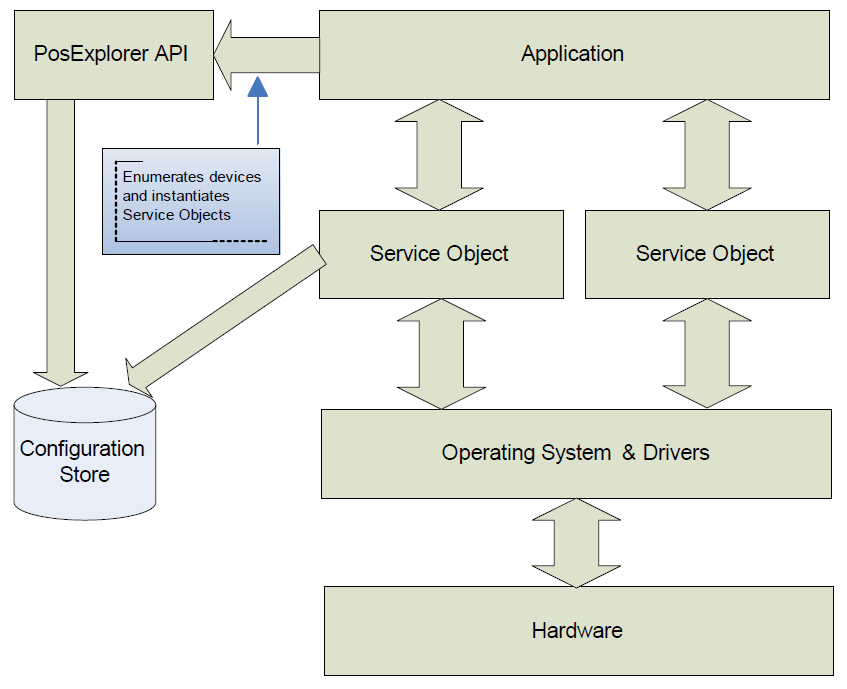
Familiarity with Microsoft Developer Integration tools including the latest version of Visual Studio and at least one of the .NET Application Development languages. Note that as there is no Control Programmer’s Guide (CPG) for POS for .NET, code samples can be found by searching for “POS for .NET SDK” located at: <https://aka.ms/p4dn-docs>.

**Note:** Examples in this Appendix use the Visual C# .NET syntax if method signatures are provided.

## Overview of POS for .NET

The following diagram shows the high level architecture of POS for .NET. An application calls into the **PosExplorer** API to enumerate available POS peripherals and to instantiate service objects for them. Once a service object is instantiated by the PosExplorer API, the application then directly communicates to it. Device-dependent service objects represent state and behavior of the physical peripheral via properties, methods, and events.

Unlike the behavior of an OPOS implementation, in POS for .NET there is no notion of control objects. Instead, the PosExplorer API acts, in some sense, as a sole control object for all device classes. There is a global configuration store where the configuration of POS for .NET is persisted. PosExplorer API reads what logical devices are defined in the system and other related information from the store. Also, configuration of the service objects and physical devices is persisted in the configuration store. Service objects can read and write their properties from and to the store.



It is important to note that provision is made for both legacy OPOS CO/SO’s software code and new .NET base class dependent software code to be used. However, the full rich features of a .NET based service cannot be expected using an OPOS legacy service object scenario. It is fully expected that over time, full-featured .NET enabled devices with full featured .NET designed services will become the preferred implementation for .NET POS applications.

Like OPOS Controls, .NET SO base classes expose properties, methods, and events to a containing Application. The Service Object is a class that implements a device class interface defined by POS for .NET. The Microsoft supplied interfaces provide the class interfaces that serve as the basis for the Applications to interact with a POS peripheral device using properties, methods, and events as defined by the UnifiedPOS standard. Responses are given to the application through method return values and parameters, properties, and events.

POS for .NET Definitions

### Device Class

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

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.

### Service Object *or* SO

A Service Object is a class that implements a device class interface defined by POS for .NET. It exposes properties and methods that are called by an application.

## Key POS for .NET Features

### .NET Interface Classes for POS Peripherals

POS for .NET supplies interface classes for peripheral devices defined in the UnifiedPOS specification. The interface classes provide the entry points as specified in the UnifiedPOS specification, but offer minimal functionality

### Base Classes for Service Objects

POS for .NET supplies fully functional **Base** classes that extend their corresponding **Basic** classes with device-specific members for nine primary UnifiedPOS device types. You could think of these classes as enhanced or extended **Basic** classes. Because **Base** classes provide a nearly complete implementation, Service Object developers should derive from these classes whenever possible.

### Basic Classes for Service Objects

POS for .NET **Basic** classes contain basic functional support for peripheral devices defined in the UnifiedPOS specification. **Basic** classes provide generic support for opening, claiming, and enabling the device, device statistics, and management of delivering events to the application. In addition, each **Basic** class contains a set of inherited and protected methods that can be implemented by the Service Object.

### Plug and Play

POS for .NET helps to bring retail peripherals to the same parity as standard PC desktop peripherals which can use the Plug and Play (PnP) Windows architecture. PnP is a feature of Windows that, with little or no user intervention, automatically installs drivers when their corresponding hardware peripherals are plugged into a PC. Currently PnP is not a feature of a UnifiedPOS implementation but usage of PnP devices is supported along with UnifiedPOS devices. For more information about supporting PnP, see [http://msdn.microsoft.com/library/default.asp?url=/](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwue/html/ch11j.asp)library/en-us/dnwue/html/ch11j.asp.

### Standardized Setup

A standard installation and uninstall procedure support of POS for .NET Service Objects is provided, which negates the requirement for a special service loader install program (as is required in OPOS).

### Device Enumeration

The ability to enumerate all the POS Peripheral devices installed on the system is provided in the POS for .NET services.

### Software-Based Device Statistics

Additional native support for hardware-specific device statistics is available in addition to device statistics that are provided for under UnifiedPOS.

### Support for OPOS Service Objects Updated in Release 1.15

POS for .NET provides for full .NET to COM interoperability as part of the library to avoid depreciating the investment in COM-based Service Objects. See Device Category Support Level table later in this chapter for specific device types supported through legacy OPOS inter-op.

### Service Object Verification Program Updated in Release 1.15

The Service Object Verification Program has been retired. It is the responsibility of the third party authoring POS for .NET service objects to ensure compatibility.

Device Category Support Level Updated in Release 1.15

The following table shows the various classes and the POS for .NET version in which they were initially supported.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Device Category | OPOS  Inter-op | Interface Class | Basic Class | Base Class |
| Belt |  | 1.12 | 1.12 |  |
| BillAcceptor |  | 1.11 | 1.11 |  |
| BillDispenser |  | 1.11 | 1.11 |  |
| Biometrics |  | 1.11 | 1.11 |  |
| BumpBar | 1.14 | 1.0 | 1.0 |  |
| CashChanger | 1.14 | 1.0 | 1.0 |  |
| CashDrawer | 1.0 | 1.0 | 1.0 | 1.0 |
| CAT (Credit Auth Terminal) | 1.12 | 1.0 | 1.0 |  |
| CheckScanner | 1.0 | 1.0 | 1.0 | 1.0 |
| CoinAcceptor |  | 1.11 | 1.11 |  |
| CoinDispenser | 1.1 | 1.0 | 1.0 |  |
| ElectronicJournal |  | 1.11 | 1.11 |  |
| ElectronicValueRW |  | 1.12 | 1.12 |  |
| FiscalPrinter | 1.14 | 1.0 | 1.0 |  |
| Gate |  | 1.12 | 1.12 |  |
| HardTotals | 1.14 | 1.0 | 1.0 |  |
| ImageScanner |  | 1.11 | 1.11 |  |
| ItemDispenser |  | 1.12 | 1.12 |  |
| Keylock | 1.1 | 1.0 | 1.0 |  |
| Lights |  | 1.12 | 1.12 |  |
| LineDisplay | 1.0 | 1.0 | 1.0 | 1.0 |
| MICR (Magnetic Ink Char Recognition) | 1.1 | 1.0 | 1.0 |  |
| MotionSensor | 1.14 | 1.0 | 1.0 |  |
| MSR (Magnetic Stripe Reader) | 1.0 | 1.0 | 1.0 | 1.0 |
| PINPad | 1.0 | 1.0 | 1.0 | 1.0 |
| PointCardRW | 1.14 | 1.0 | 1.0 |  |
| POSKeyboard | 1.0 | 1.0 | 1.0 | 1.0 |
| POSPower | 1.1 | 1.0 | 1.0 |  |
| POSPrinter | 1.0 | 1.0 | 1.0 | 1.0 |
| RemoteOrderDisplay | 1.14 | 1.0 | 1.0 |  |
| RFIDScanner |  | 1.12 | 1.12 | 1.12 |
| Scale | 1.1 | 1.0 | 1.0 |  |
| Scanner (Bar Code Reader) | 1.0 | 1.0 | 1.0 | 1.0 |
| SignatureCapture | 1.1 | 1.0 | 1.0 |  |
| SmartCardRW | 1.14 | 1.0 | 1.0 |  |
| ToneIndicator | 1.1 | 1.0 | 1.0 |  |

## Key Programming Construct Differences from OPOS

### Naming Conventions

The library uses Pascal naming conventions for .NET classes and parameters of methods are camel-case. These conventions are consistent with .NET Guidelines for Class Library Developers. For more information on .NET Guidelines for Class Library Developers, see: <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpgenref/html/cpconnetframeworkdesignguidelines.asp>

### Enumerations Updated in Release 1.13

POS for .NET makes extensive use of enumerations, which serves several purposes. Enumerations force both the application and its Device Service Object to use in-bounds parameters. This method of type checking helps avoid bugs that result from out-of-bounds parameters or from passing return values.

In addition, the use of enumerations eliminates the need for a large list of constants in the name space. Best practices for a library development require range validation for constant data types, something that is automatically provided by using enumerations.

Note that there are cases where the range of acceptable enumeration values is bound; however, the individual number of choices can be quite large. An example is the *timeout* parameter. The possible values are -1 through the size of an Int32. The value of -1 is interpreted as “wait forever” and all values from 0 through the size of an Int32 represent the number of milliseconds before a timeout error occurs. Best practices in this case would be to use a constant (such as -1) to define “wait forever” and to use an Int32 value for the non-wait condition.

The following pages contain a table showing the current OPOS reference implementation constant definitions and the corresponding POS for .NET enumerations.

|  |  |  |  |
| --- | --- | --- | --- |
| UnifiedPOS Name | POS for .NET | | |
| ClassName | Parameter | |
| Type | Name |
| S\_CLOSED | ControlState | enum Constant | Closed |
| S\_IDLE | ControlState | enum Constant | Idle |
| S\_BUSY | ControlState | enum Constant | Busy |
| S\_ERROR | ControlState | enum Constant | Error |
|  |  |  |  |
| SUCCESS | ErrorCode | enum Constant | Success |
| No\_Equivalent\_Defined0000 | ErrorCode | enum Constant | Unspecified |
| E\_CLOSED | ErrorCode | enum Constant | Closed |
| E\_CLAIMED | ErrorCode | enum Constant | Claimed |
| E\_NOTCLAIMED | ErrorCode | enum Constant | NotClaimed |
| E\_NOSERVICE | ErrorCode | enum Constant | NoService |
| E\_DISABLED | ErrorCode | enum Constant | Disabled |
| E\_ILLEGAL | ErrorCode | enum Constant | Illegal |
| E\_NOHARDWARE | ErrorCode | enum Constant | NoHardware |
| E\_OFFLINE | ErrorCode | enum Constant | Offline |
| E\_NOEXIST | ErrorCode | enum Constant | NoExist |
| E\_EXISTS | ErrorCode | enum Constant | Exists |
| E\_FAILURE | ErrorCode | enum Constant | Failure |
| E\_TIMEOUT | ErrorCode | enum Constant | Timeout |
| E\_BUSY | ErrorCode | enum Constant | Busy |
| E\_EXTENDED | ErrorCode | enum Constant | Extended |
|  |  |  |  |
| ESTATS\_ERROR | PosCommon | System.Int32 | ExtendedErrorStatistics |
|  |  |  |  |
| CH\_INTERNAL | HealthCheckLevel | enum Constant | Internal |
| CH\_EXTERNAL | HealthCheckLevel | enum Constant | External |
| CH\_INTERACTIVE | HealthCheckLevel | enum Constant | Interactive |
|  |  |  |  |
| PR\_NONE | PowerReporting | enum Constant | None |
| PR\_STANDARD | PowerReporting | enum Constant | Standard |
| PR\_ADVANCED | PowerReporting | enum Constant | Advanced |
|  |  |  |  |
| PN\_DISABLED | PowerNotification | enum Constant | Disabled |
| PN\_ENABLED | PowerNotification | enum Constant | Enabled |
|  |  |  |  |
| PN\_PS\_UNKNOWN | PowerState | enum Constant | Unknown |
| PS\_ONLINE | PowerState | enum Constant | Online |
| PS\_OFF | PowerState | enum Constant | Off |
| PS\_OFFLINE | PowerState | enum Constant | Offline |
| PS\_OFF\_OFFLINE | PowerState | enum Constant | OffOffline |
|  |  |  |  |
| EL\_OUTPUT | ErrorLocus | enum Constant | Output |
| EL\_INPUT | ErrorLocus | enum Constant | Input |
| EL\_INPUT\_DATA | ErrorLocus | enum Constant | InputData |
|  |  |  |  |
| ER\_RETRY | ErrorResponse | enum Constant | Retry |
| ER\_CLEAR | ErrorResponse | enum Constant | Clear |
| ER\_CONTINUEINPUT | ErrorResponse | enum Constant | ContinueInput |
|  |  |  |  |
| SUE\_POWER\_ONLINE | PosCommon | System.Int32 | StatusPowerOnline |
| SUE\_POWER\_OFF | PosCommon | System.Int32 | StatusPowerOff |
| SUE\_POWER\_OFFLINE | PosCommon | System.Int32 | StatusPowerOffline |
| SUE\_POWER\_OFF\_OFFLINE | PosCommon | System.Int32 | StatusPowerOffOffline |
|  |  |  |  |
| CFV\_FIRMWARE\_DIFFERENT | CompareFirmwareResult | enum Constant | Different |
| CFV\_FIRMWARE\_NEWER | CompareFirmwareResult | enum Constant | Newer |
| CFV\_FIRMWARE\_OLDER | CompareFirmwareResult | enum Constant | Older |
| CFV\_FIRMWARE\_SAME | CompareFirmwareResult | enum Constant | Same |
| CFV\_FIRMWARE\_UNKNOWN | CompareFirmwareResult | enum Constant | Unknown |
|  |  |  |  |
| SUE\_UF\_FAILED\_DEV\_OK | PosCommon | System.Int32 | StatusUpdateFirmwareFailedDeviceOk |
| SUE\_UF\_FAILED\_DEV\_UNRECOVERABLE | PosCommon | System.Int32 | StatusUpdateFirmwareFailedDeviceUnrecoverable |
| SUE\_UF\_FAILED\_DEV\_NEEDS\_FIRMWARE | PosCommon | System.Int32 | StatusUpdateFirmwareFailedDeviceNeedsFirmware |
| SUE\_UF\_FAILED\_DEV\_UNKNOWN | PosCommon | System.Int32 | StatusUpdateFirmwareFailedDeviceUnknown |
| SUE\_UF\_COMPLETE | PosCommon | System.Int32 | StatusUpdateFirmwareComplete |
| SUE\_UF\_COMPLETE\_DEV\_NOT\_RESTORED | PosCommon | System.Int32 | StatusUpdateFirmwareCompleteDeviceNotRestored |
| SUE\_UF\_PROGRESS + 1 to 100 | PosCommon | System.Int32 | StatusUpdateFirmwareProgress |
|  |  |  |  |
| FOREVER | PosCommon | System.Int32 | WaitForever |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| UnifiedPOS Name | POS for .NET | | |
| ClassName | Parameter | |
| Type | Name |
| BB\_UID\_1 | DeviceUnits | enum Constant | Unit1 |
| BB\_UID\_2 | DeviceUnits | enum Constant | Unit2 |
| BB\_UID\_3 | DeviceUnits | enum Constant | Unit3 |
| BB\_UID\_4 | DeviceUnits | enum Constant | Unit4 |
| BB\_UID\_5 | DeviceUnits | enum Constant | Unit5 |
| BB\_UID\_6 | DeviceUnits | enum Constant | Unit6 |
| BB\_UID\_7 | DeviceUnits | enum Constant | Unit7 |
| BB\_UID\_8 | DeviceUnits | enum Constant | Unit8 |
| BB\_UID\_9 | DeviceUnits | enum Constant | Unit9 |
| BB\_UID\_10 | DeviceUnits | enum Constant | Unit10 |
| BB\_UID\_11 | DeviceUnits | enum Constant | Unit11 |
| BB\_UID\_12 | DeviceUnits | enum Constant | Unit12 |
| BB\_UID\_13 | DeviceUnits | enum Constant | Unit13 |
| BB\_UID\_14 | DeviceUnits | enum Constant | Unit14 |
| BB\_UID\_15 | DeviceUnits | enum Constant | Unit15 |
| BB\_UID\_16 | DeviceUnits | enum Constant | Unit16 |
| BB\_UID\_17 | DeviceUnits | enum Constant | Unit17 |
| BB\_UID\_18 | DeviceUnits | enum Constant | Unit18 |
| BB\_UID\_19 | DeviceUnits | enum Constant | Unit19 |
| BB\_UID\_20 | DeviceUnits | enum Constant | Unit20 |
| BB\_UID\_21 | DeviceUnits | enum Constant | Unit21 |
| BB\_UID\_22 | DeviceUnits | enum Constant | Unit22 |
| BB\_UID\_23 | DeviceUnits | enum Constant | Unit23 |
| BB\_UID\_24 | DeviceUnits | enum Constant | Unit24 |
| BB\_UID\_25 | DeviceUnits | enum Constant | Unit25 |
| BB\_UID\_26 | DeviceUnits | enum Constant | Unit26 |
| BB\_UID\_27 | DeviceUnits | enum Constant | Unit27 |
| BB\_UID\_28 | DeviceUnits | enum Constant | Unit28 |
| BB\_UID\_29 | DeviceUnits | enum Constant | Unit29 |
| BB\_UID\_30 | DeviceUnits | enum Constant | Unit30 |
| BB\_UID\_31 | DeviceUnits | enum Constant | Unit31 |
| BB\_UID\_32 | DeviceUnits | enum Constant | Unit32 |
|  |  |  |  |
| BB\_DE\_KEY | BumpBar | System.Int32 | DataEventKey |
|  |  |  |  |
| CASH\_SUE\_DRAWERCLOSED | CashDrawerStatus | enum Constant | Closed |
| CASH\_SUE\_DRAWEROPEN | CashDrawerStatus | enum Constant | Open |
|  |  |  |  |
| CAT\_PAYMENT\_LUMP | PaymentCondition | enum Constant | Lump |
| CAT\_PAYMENT\_BONUS\_1 | PaymentCondition | enum Constant | Bonus1 |
| CAT\_PAYMENT\_BONUS\_2 | PaymentCondition | enum Constant | Bonus2 |
| CAT\_PAYMENT\_BONUS\_3 | PaymentCondition | enum Constant | Bonus3 |
| CAT\_PAYMENT\_BONUS\_4 | PaymentCondition | enum Constant | Bonus4 |
| CAT\_PAYMENT\_BONUS\_5 | PaymentCondition | enum Constant | Bonus5 |
| CAT\_PAYMENT\_INSTALLMENT\_1 | PaymentCondition | enum Constant | Installment1 |
| CAT\_PAYMENT\_INSTALLMENT\_2 | PaymentCondition | enum Constant | Installment2 |
| CAT\_PAYMENT\_INSTALLMENT\_3 | PaymentCondition | enum Constant | Installment3 |
| CAT\_PAYMENT\_BONUS\_COMBINATION\_1 | PaymentCondition | enum Constant | BonusCombination1 |
| CAT\_PAYMENT\_BONUS\_COMBINATION\_2 | PaymentCondition | enum Constant | BonusCombination2 |
| CAT\_PAYMENT\_BONUS\_COMBINATION\_3 | PaymentCondition | enum Constant | BonusCombination3 |
| CAT\_PAYMENT\_BONUS\_COMBINATION\_4 | PaymentCondition | enum Constant | BonusCombination4 |
| CAT\_PAYMENT\_REVOLVING | PaymentCondition | enum Constant | Revolving |
| CAT\_PAYMENT\_DEBIT | PaymentCondition | enum Constant | Debit |
|  |  |  |  |
| CAT\_TRANSACTION\_SALES | CreditTransactionType | enum Constant | Sales |
| CAT\_TRANSACTION\_VOID | CreditTransactionType | enum Constant | Void |
| CAT\_TRANSACTION\_REFUND | CreditTransactionType | enum Constant | Refund |
| CAT\_TRANSACTION\_VOIDPRESALES | CreditTransactionType | enum Constant | VoidPreSales |
| CAT\_TRANSACTION\_COMPLETION | CreditTransactionType | enum Constant | Completion |
| CAT\_TRANSACTION\_PRESALES | CreditTransactionType | enum Constant | PreSales |
| CAT\_TRANSACTION\_CHECKCARD | CreditTransactionType | enum Constant | CheckCard |
|  |  |  |  |
| CAT\_MEDIA\_UNSPECIFIED | PaymentMedia | enum Constant | Unspecified |
| CAT\_MEDIA\_NONDEFINE | PaymentMedia | No\_Equivalent\_Defined |  |
| CAT\_MEDIA\_CREDIT | PaymentMedia | enum Constant | Credit |
| CAT\_MEDIA\_DEBIT | PaymentMedia | enum Constant | Debit |
|  |  |  |  |
| ECAT\_CENTERERROR | Cat | System.Int32 | ExtendedErrorCenterError |
| ECAT\_COMMANDERROR | Cat | System.Int32 | ExtendedErrorCommandError |
| ECAT\_RESET | Cat | System.Int32 | ExtendedErrorReset |
| ECAT\_COMMUNICATIONERROR | Cat | System.Int32 | ExtendedErrorCommunicationError |
| ECAT\_DAILYLOGOVERFLOW | Cat | System.Int32 | ExtendedErrorDailyLogOverflow |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| UnifiedPOS Name | POS for .NET | | |
| ClassName | Parameter | |
| Type | Name |
| CAT\_DL\_NONE | CatLogs | enum Constant | None |
| CAT\_DL\_REPORTING | CatLogs | enum Constant | Reporting |
| CAT\_DL\_SETTLEMENT | CatLogs | enum Constant | Settlement |
| CAT\_DL\_REPORTING\_SETTLEMENT | CatLogs | enum Constant | ReportingAndSettlement |
|  |  |  |  |
| CHAN\_STATUS\_OK | CashChangerStatus | enum Constant | OK |
| CHAN\_STATUS\_EMPTY | CashChangerStatus | enum Constant | Empty |
| CHAN\_STATUS\_NEAREMPTY | CashChangerStatus | enum Constant | NearEmpty |
| CHAN\_STATUS\_EMPTYOK | CashChangerStatus | No\_Equivalent\_Defined |  |
|  |  |  |  |
| No\_Equivalent\_Defined | CashChangerFullStatus | enum Constant | OK |
| CHAN\_STATUS\_FULL | CashChangerFullStatus | enum Constant | Full |
| CHAN\_STATUS\_NEARFULL | CashChangerFullStatus | enum Constant | NearFull |
| CHAN\_STATUS\_FULLOK | CashChangerFullStatus | No\_Equivalent\_Defined |  |
|  |  |  |  |
| CHAN\_STATUS\_JAM | CashChangerStatus | enum Constant | Jam |
| CHAN\_STATUS\_JAMOK | CashChangerStatus | No\_Equivalent\_Defined |  |
|  |  |  |  |
| CHAN\_STATUS\_ASYNC | CashChanger | System.Int32 | StatusAsync |
|  |  |  |  |
| CHAN\_STATUS\_DEPOSIT\_START | CashDepositStatus | enum Constant | Start |
| CHAN\_STATUS\_DEPOSIT\_END | CashDepositStatus | enum Constant | End |
| CHAN\_STATUS\_DEPOSIT\_NONE | CashDepositStatus | enum Constant | None |
| CHAN\_STATUS\_DEPOSIT\_COUNT | CashDepositStatus | enum Constant | Count |
| CHAN\_STATUS\_DEPOSIT\_JAM | CashDepositStatus | enum Constant | Jam |
|  |  |  |  |
| CHAN\_DEPOSIT\_CHANGE | CashDepositAction | enum Constant | Change |
| CHAN\_DEPOSIT\_NOCHANGE | CashDepositAction | enum Constant | NoChange |
| CHAN\_DEPOSIT\_REPAY | CashDepositAction | enum Constant | Repay |
|  |  |  |  |
| CHAN\_DEPOSIT\_PAUSE | CashDepositPause | enum Constant | Pause |
| CHAN\_DEPOSIT\_RESTART | CashDepositPause | enum Constant | Restart |
|  |  |  |  |
| ECHAN\_OVERDISPENSE | CashChanger | System.Int32 | ExtendedErrorOverDispense |
|  |  |  |  |
| CHK\_CCL\_MONO | CheckColors | enum Constant | Mono |
| CHK\_CCL\_GRAYSCALE | CheckColors | enum Constant | GrayScale |
| CHK\_CCL\_16 | CheckColors | enum Constant | Color16 |
| CHK\_CCL\_256 | CheckColors | enum Constant | Color256 |
| CHK\_CCL\_FULL | CheckColors | enum Constant | Full |
|  |  |  |  |
| CHK\_CIF\_NATIVE | CheckImageFormats | enum Constant | Native |
| CHK\_CIF\_TIFF | CheckImageFormats | enum Constant | Tiff |
| CHK\_CIF\_BMP | CheckImageFormats | enum Constant | Bmp |
| CHK\_CIF\_JPEG | CheckImageFormats | enum Constant | Jpeg |
| CHK\_CIF\_GIF | CheckImageFormats | enum Constant | Gif |
|  |  |  |  |
| CHK\_CL\_MONO | CheckColors | enum Constant | Mono |
| CHK\_CL\_GRAYSCALE | CheckColors | enum Constant | GrayScale |
| CHK\_CL\_16 | CheckColors | enum Constant | Color16 |
| CHK\_CL\_256 | CheckColors | enum Constant | Color256 |
| CHK\_CL\_FULL | CheckColors | enum Constant | Full |
|  |  |  |  |
| CHK\_IF\_NATIVE | CheckImageFormats | enum Constant | Native |
| CHK\_IF\_TIFF | CheckImageFormats | enum Constant | Tiff |
| CHK\_IF\_BMP | CheckImageFormats | enum Constant | Bmp |
| CHK\_IF\_JPEG | CheckImageFormats | enum Constant | Jpeg |
| CHK\_IF\_GIF | CheckImageFormats | enum Constant | Gif |
|  |  |  |  |
| CHK\_IMS\_EMPTY | ImageMemoryStatus | enum Constant | Empty |
| CHK\_IMS\_OK | ImageMemoryStatus | enum Constant | OK |
| CHK\_IMS\_FULL | ImageMemoryStatus | enum Constant | Full |
|  |  |  |  |
| CHK\_MM\_DOTS | MapMode | enum Constant | Dots |
| CHK\_MM\_TWIPS | MapMode | enum Constant | Twips |
| CHK\_MM\_ENGLISH | MapMode | enum Constant | English |
| CHK\_MM\_METRIC | MapMode | enum Constant | Metric |
|  |  |  |  |
| CHK\_CLR\_ALL | CheckImageClear | enum Constant | All |
| CHK\_CLR\_BY\_FILEID | CheckImageClear | enum Constant | FileId |
| CHK\_CLR\_BY\_FILEINDEX | CheckImageClear | enum Constant | FileIndex |
| CHK\_CLR\_BY\_IMAGETAGDATA | CheckImageClear | enum Constant | ImageTagData |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| UnifiedPOS Name | POS for .NET | | |
| ClassName | Parameter | |
| Type | Name |
| CHK\_CROP\_AREA\_ENTIRE\_IMAGE | CheckScanner | System.Int32 | CropEntireImage |
| CHK\_CROP\_AREA\_RESET\_ALL | CheckScanner | System.Int32 | CropResetAll |
| CHK\_CROP\_AREA\_RIGHT | CheckScanner | System.Int32 | CropRight |
| CHK\_CROP\_AREA\_BOTTOM | CheckScanner | System.Int32 | CropBottom |
|  |  |  |  |
| CHK\_LOCATE\_BY\_FILEID | CheckImageLocate | enum Constant | FileId |
| CHK\_LOCATE\_BY\_FILEINDEX | CheckImageLocate | enum Constant | FileIndex |
| CHK\_LOCATE\_BY\_IMAGETAGDATA | CheckImageLocate | enum Constant | ImageTagData |
|  |  |  |  |
| CHK\_SUE\_SCANCOMPLETE | CheckScannerStatus | enum Constant | ScanComplete |
|  |  |  |  |
| ECHK\_NOCHECK | CheckScanner | System.Int32 | ExtendedErrorNoCheck |
| ECHK\_CHECK | CheckScanner | System.Int32 | ExtendedErrorCheck |
| ECHK\_NOROOM | CheckScanner | System.Int32 | ExtendedErrorNoRoom |
|  |  |  |  |
| COIN\_STATUS\_OK | CoinDispenserStatus | enum Constant | OK |
| COIN\_STATUS\_EMPTY | CoinDispenserStatus | enum Constant | Empty |
| COIN\_STATUS\_NEAREMPTY | CoinDispenserStatus | enum Constant | NearEmpty |
| COIN\_STATUS\_JAM | CoinDispenserStatus | enum Constant | Jam |
|  |  |  |  |
| DISP\_CB\_NOBLINK | DisplayBlink | enum Constant | None |
| DISP\_CB\_BLINKALL | DisplayBlink | enum Constant | All |
| DISP\_CB\_BLINKEACH | DisplayBlink | enum Constant | Each |
|  |  |  |  |
| DISP\_CCS\_NUMERIC | CharacterSetCapability | enum Constant | Numeric |
| DISP\_CCS\_ALPHA | CharacterSetCapability | enum Constant | Alpha |
| DISP\_CCS\_ASCII | CharacterSetCapability | enum Constant | Ascii |
| DISP\_CCS\_KANA | CharacterSetCapability | enum Constant | Kana |
| DISP\_CCS\_KANJI | CharacterSetCapability | enum Constant | Kanji |
| DISP\_CCS\_UNICODE | CharacterSetCapability | enum Constant | Unicode |
|  |  |  |  |
| DISP\_CCT\_NONE | DisplayCursors | enum Constant | None |
| DISP\_CCT\_FIXED | DisplayCursors | enum Constant | Fixed |
| DISP\_CCT\_BLOCK | DisplayCursors | enum Constant | Block |
| DISP\_CCT\_HALFBLOCK | DisplayCursors | enum Constant | HalfBlock |
| DISP\_CCT\_UNDERLINE | DisplayCursors | enum Constant | Underline |
| DISP\_CCT\_REVERSE | DisplayCursors | enum Constant | Reverse |
| DISP\_CCT\_OTHER | DisplayCursors | enum Constant | Other |
| DISP\_CCT\_BLINK | DisplayCursors | enum Constant | Blink |
|  |  |  |  |
| DISP\_CRB\_NONE | DisplayReadBack | enum Constant | None |
| DISP\_CRB\_SINGLE | DisplayReadBack | enum Constant | Single |
|  |  |  |  |
| DISP\_CR\_NONE | DisplayReverse | enum Constant | None |
| DISP\_CR\_REVERSEALL | DisplayReverse | enum Constant | All |
| DISP\_CR\_REVERSEEACH | DisplayReverse | enum Constant | Each |
|  |  |  |  |
| DISP\_CS\_UNICODE | PosCommon | System.Int32 | CharacterSetUnicode |
| DISP\_CS\_ASCII | PosCommon | System.Int32 | CharacterSetAscii |
| DISP\_CS\_WINDOWS | PosCommon | System.Int32 | No\_Equivalent\_Defined |
| DISP\_CS\_ANSI | PosCommon | System.Int32 | CharacterSetAnsi |
| DISP\_CT\_NONE | DisplayCursors | enum Constant | None |
|  |  |  |  |
| DISP\_CT\_FIXED | DisplayCursors | enum Constant | Fixed |
| DISP\_CT\_BLOCK | DisplayCursors | enum Constant | Block |
| DISP\_CT\_HALFBLOCK | DisplayCursors | enum Constant | HalfBlock |
| DISP\_CT\_UNDERLINE | DisplayCursors | enum Constant | Underline |
| DISP\_CT\_REVERSE | DisplayCursors | enum Constant | Reverse |
| DISP\_CT\_OTHER | DisplayCursors | enum Constant | Other |
| DISP\_CT\_BLINK | DisplayCursors | enum Constant | Blink |
|  |  |  |  |
| DISP\_MT\_NONE | DisplayMarqueeType | enum Constant | None |
| DISP\_MT\_UP | DisplayMarqueeType | enum Constant | Up |
| DISP\_MT\_DOWN | DisplayMarqueeType | enum Constant | Down |
| DISP\_MT\_LEFT | DisplayMarqueeType | enum Constant | Left |
| DISP\_MT\_RIGHT | DisplayMarqueeType | enum Constant | Right |
| DISP\_MT\_INIT | DisplayMarqueeType | enum Constant | Init |
|  |  |  |  |
| DISP\_MF\_WALK | DisplayMarqueeFormat | enum Constant | Walk |
| DISP\_MF\_PLACE | DisplayMarqueeFormat | enum Constant | Place |
|  |  |  |  |
| DISP\_DT\_NORMAL | DisplayTextMode | enum Constant | Normal |
| DISP\_DT\_BLINK | DisplayTextMode | enum Constant | Blink |
| DISP\_DT\_REVERSE | DisplayTextMode | enum Constant | Reverse |
| DISP\_DT\_BLINK\_REVERSE | DisplayTextMode | enum Constant | BlinkReverse |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| UnifiedPOS Name | POS for .NET | | |
| ClassName | Parameter | |
| Type | Name |
| DISP\_ST\_UP | DisplayScrollText | enum Constant | Up |
| DISP\_ST\_DOWN | DisplayScrollText | enum Constant | Down |
| DISP\_ST\_LEFT | DisplayScrollText | enum Constant | Left |
| DISP\_ST\_RIGHT | DisplayScrollText | enum Constant | Right |
|  |  |  |  |
| DISP\_SD\_OFF | DisplaySetDescriptor | enum Constant | Off |
| DISP\_SD\_ON | DisplaySetDescriptor | enum Constant | On |
| DISP\_SD\_BLINK | DisplaySetDescriptor | enum Constant | Blink |
|  |  |  |  |
| DISP\_BM\_ASIS | LineDisplay | System.Int32 | DisplayBitmapAsIs |
| DISP\_BM\_LEFT | LineDisplay | System.Int32 | DisplayBitmapLeft |
| DISP\_BM\_CENTER | LineDisplay | System.Int32 | DisplayBitmapCenter |
| DISP\_BM\_RIGHT | LineDisplay | System.Int32 | DisplayBitmapRight |
| DISP\_BM\_TOP | LineDisplay | System.Int32 | DisplayBitmapTop |
| DISP\_BM\_BOTTOM | LineDisplay | System.Int32 | DisplayBitmapBottom |
| EDISP\_TOOBIG | LineDisplay | System.Int32 | ExtendedErrorTooBig |
| EDISP\_BADFORMAT | LineDisplay | System.Int32 | ExtendedErrorBadFormat |
|  |  |  |  |
| FPTR\_S\_JOURNAL | FiscalPrinterStations | enum Constant | Journal |
| FPTR\_S\_RECEIPT | FiscalPrinterStations | enum Constant | Receipt |
| FPTR\_S\_SLIP | FiscalPrinterStations | enum Constant | Slip |
| FPTR\_S\_JOURNAL\_RECEIPT | FiscalPrinterStations | enum Constant | JournalReceipt |
| No\_Equivalent\_Defined | FiscalPrinterStations | enum Constant | JournalSlip |
| No\_Equivalent\_Defined | FiscalPrinterStations | enum Constant | ReceiptSlip |
|  |  |  |  |
| FPTR\_AC\_BRC | FiscalCurrency | enum Constant | BrazilianCruceiro |
| FPTR\_AC\_BGL | FiscalCurrency | enum Constant | BulgarianLev |
| FPTR\_AC\_EUR | FiscalCurrency | enum Constant | Euro |
| FPTR\_AC\_GRD | FiscalCurrency | enum Constant | GreekDrachma |
| FPTR\_AC\_HUF | FiscalCurrency | enum Constant | HungarianForint |
| FPTR\_AC\_ITL | FiscalCurrency | enum Constant | ItalianLira |
| FPTR\_AC\_PLZ | FiscalCurrency | enum Constant | PolishZloty |
| FPTR\_AC\_ROL | FiscalCurrency | enum Constant | RomanianLeu |
| FPTR\_AC\_RUR | FiscalCurrency | enum Constant | RussianRouble |
| FPTR\_AC\_TRL | FiscalCurrency | enum Constant | TurkishLira |
|  |  |  |  |
| FPTR\_CID\_FIRST | FiscalContractorId | enum Constant | First |
| FPTR\_CID\_SECOND | FiscalContractorId | enum Constant | Second |
| FPTR\_CID\_SINGLE | FiscalContractorId | enum Constant | Single |
|  |  |  |  |
| FPTR\_CC\_BRAZIL | FiscalCountryCodes | enum Constant | Brazil |
| FPTR\_CC\_GREECE | FiscalCountryCodes | enum Constant | Greece |
| FPTR\_CC\_HUNGARY | FiscalCountryCodes | enum Constant | Hungary |
| FPTR\_CC\_ITALY | FiscalCountryCodes | enum Constant | Italy |
| FPTR\_CC\_POLAND | FiscalCountryCodes | enum Constant | Poland |
| FPTR\_CC\_TURKEY | FiscalCountryCodes | enum Constant | Turkey |
| FPTR\_CC\_RUSSIA | FiscalCountryCodes | enum Constant | Russia |
| FPTR\_CC\_BULGARIA | FiscalCountryCodes | enum Constant | Bulgaria |
| FPTR\_CC\_ROMANIA | FiscalCountryCodes | enum Constant | Romania |
|  |  |  |  |
| FPTR\_DT\_CONF | FiscalDateType | enum Constant | Configuration |
| FPTR\_DT\_EOD | FiscalDateType | enum Constant | EndOfDay |
| FPTR\_DT\_RESET | FiscalDateType | enum Constant | Reset |
| FPTR\_DT\_RTC | FiscalDateType | enum Constant | RealTimeClock |
| FPTR\_DT\_VAT | FiscalDateType | enum Constant | VatChange |
|  |  |  |  |
| FPTR\_EL\_NONE | FiscalErrorLevel | enum Constant | None |
| FPTR\_EL\_RECOVERABLE | FiscalErrorLevel | enum Constant | Recoverable |
| FPTR\_EL\_FATAL | FiscalErrorLevel | enum Constant | Fatal |
| FPTR\_EL\_BLOCKED | FiscalErrorLevel | enum Constant | Blocked |
|  |  |  |  |
| FPTR\_PS\_MONITOR | FiscalPrinterState | enum Constant | Monitor |
| FPTR\_PS\_FISCAL\_RECEIPT | FiscalPrinterState | enum Constant | FiscalReceipt |
| FPTR\_PS\_FISCAL\_RECEIPT\_TOTAL | FiscalPrinterState | enum Constant | FiscalReceiptTotal |
| FPTR\_PS\_FISCAL\_RECEIPT\_ENDING | FiscalPrinterState | enum Constant | FiscalReceiptEnding |
| FPTR\_PS\_FISCAL\_DOCUMENT | FiscalPrinterState | enum Constant | FiscalDocument |
| FPTR\_PS\_FIXED\_OUTPUT | FiscalPrinterState | enum Constant | FixedOutput |
| FPTR\_PS\_ITEM\_LIST | FiscalPrinterState | enum Constant | ItemList |
| FPTR\_PS\_LOCKED | FiscalPrinterState | enum Constant | Locked |
| FPTR\_PS\_NONFISCAL | FiscalPrinterState | enum Constant | NonFiscal |
| FPTR\_PS\_REPORT | FiscalPrinterState | enum Constant | Report |
|  |  |  |  |
| FPTR\_RS\_RECEIPT | FiscalReceiptStation | enum Constant | Receipt |
| FPTR\_RS\_SLIP | FiscalReceiptStation | enum Constant | Slip |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| UnifiedPOS Name | POS for .NET | | |
| ClassName | Parameter | |
| Type | Name |
| FPTR\_RT\_CASH\_IN | FiscalReceiptType | enum Constant | CashIn |
| FPTR\_RT\_CASH\_OUT | FiscalReceiptType | enum Constant | CashOut |
| FPTR\_RT\_GENERIC | FiscalReceiptType | enum Constant | Generic |
| FPTR\_RT\_SALES | FiscalReceiptType | enum Constant | Sales |
| FPTR\_RT\_SERVICE | FiscalReceiptType | enum Constant | Service |
| FPTR\_RT\_SIMPLE\_INVOICE | FiscalReceiptType | enum Constant | SimpleInvoice |
|  |  |  |  |
| FPTR\_MT\_ADVANCE | FiscalMessageType | enum Constant | Advance |
| FPTR\_MT\_ADVANCE\_PAID | FiscalMessageType | enum Constant | AdvancePaid |
| FPTR\_MT\_AMOUNT\_TO\_BE\_PAID | FiscalMessageType | enum Constant | AmountToBePaid |
| FPTR\_MT\_AMOUNT\_TO\_BE\_PAID\_BACK | FiscalMessageType | enum Constant | AmountToBePaidBack |
| FPTR\_MT\_CARD | FiscalMessageType | enum Constant | Card |
| FPTR\_MT\_CARD\_NUMBER | FiscalMessageType | enum Constant | CardNumber |
| FPTR\_MT\_CARD\_TYPE | FiscalMessageType | enum Constant | CardType |
| FPTR\_MT\_CASH | FiscalMessageType | enum Constant | Cash |
| FPTR\_MT\_CASHIER | FiscalMessageType | enum Constant | Cashier |
| FPTR\_MT\_CASH\_REGISTER\_NUMBER | FiscalMessageType | enum Constant | CashRegisterNumber |
| FPTR\_MT\_CHANGE | FiscalMessageType | enum Constant | Change |
| FPTR\_MT\_CHEQUE | FiscalMessageType | enum Constant | Cheque |
| FPTR\_MT\_CLIENT\_NUMBER | FiscalMessageType | enum Constant | ClientNumber |
| FPTR\_MT\_CLIENT\_SIGNATURE | FiscalMessageType | enum Constant | ClientSignature |
| FPTR\_MT\_COUNTER\_STATE | FiscalMessageType | enum Constant | CounterState |
| FPTR\_MT\_CREDIT\_CARD | FiscalMessageType | enum Constant | CreditCard |
| FPTR\_MT\_CURRENCY | FiscalMessageType | enum Constant | Currency |
| FPTR\_MT\_CURRENCY\_VALUE | FiscalMessageType | enum Constant | CurrencyValue |
| FPTR\_MT\_DEPOSIT | FiscalMessageType | enum Constant | Deposit |
| FPTR\_MT\_DEPOSIT\_RETURNED | FiscalMessageType | enum Constant | DepositReturned |
| FPTR\_MT\_DOT\_LINE | FiscalMessageType | enum Constant | DotLine |
| FPTR\_MT\_DRIVER\_NUMB | FiscalMessageType | enum Constant | DriverNumber |
| FPTR\_MT\_EMPTY\_LINE | FiscalMessageType | enum Constant | EmptyLine |
| FPTR\_MT\_FREE\_TEXT | FiscalMessageType | enum Constant | FreeText |
| FPTR\_MT\_FREE\_TEXT\_WITH\_DAY\_LIMIT | FiscalMessageType | enum Constant | FreeTextWithDayLimit |
| FPTR\_MT\_GIVEN\_DISCOUNT | FiscalMessageType | enum Constant | GivenDiscount |
| FPTR\_MT\_LOCAL\_CREDIT | FiscalMessageType | enum Constant | LocalCredit |
| FPTR\_MT\_MILEAGE\_KM | FiscalMessageType | enum Constant | MileageKilometers |
| FPTR\_MT\_NOTE | FiscalMessageType | enum Constant | Note |
| FPTR\_MT\_PAID | FiscalMessageType | enum Constant | Paid |
| FPTR\_MT\_PAY\_IN | FiscalMessageType | enum Constant | PayIn |
| FPTR\_MT\_POINT\_GRANTED | FiscalMessageType | enum Constant | PointGranted |
| FPTR\_MT\_POINTS\_BONUS | FiscalMessageType | enum Constant | PointsBonus |
| FPTR\_MT\_POINTS\_RECEIPT | FiscalMessageType | enum Constant | PointsReceipt |
| FPTR\_MT\_POINTS\_TOTAL | FiscalMessageType | enum Constant | PointsTotal |
| FPTR\_MT\_PROFITED | FiscalMessageType | enum Constant | Profited |
| FPTR\_MT\_RATE | FiscalMessageType | enum Constant | Rate |
| FPTR\_MT\_REGISTER\_NUMB | FiscalMessageType | enum Constant | RegisterNumber |
| FPTR\_MT\_SHIFT\_NUMBER | FiscalMessageType | enum Constant | ShiftNumber |
| FPTR\_MT\_STATE\_OF\_AN\_ACCOUNT | FiscalMessageType | enum Constant | StateOfAnAccount |
| FPTR\_MT\_SUBSCRIPTION | FiscalMessageType | enum Constant | Subscription |
| FPTR\_MT\_TABLE | FiscalMessageType | enum Constant | Table |
| FPTR\_MT\_THANK\_YOU\_FOR\_LOYALTY | FiscalMessageType | enum Constant | ThankYouForLoyalty |
| FPTR\_MT\_TRANSACTION\_NUMB | FiscalMessageType | enum Constant | TransactionNumber |
| FPTR\_MT\_VALID\_TO | FiscalMessageType | enum Constant | ValidTo |
| FPTR\_MT\_VOUCHER | FiscalMessageType | enum Constant | Voucher |
| FPTR\_MT\_VOUCHER\_PAID | FiscalMessageType | enum Constant | VoucherPaid |
| FPTR\_MT\_VOUCHER\_VALUE | FiscalMessageType | enum Constant | VoucherValue |
| FPTR\_MT\_WITH\_DISCOUNT | FiscalMessageType | enum Constant | WithDiscount |
| FPTR\_MT\_WITHOUT\_UPLIFT | FiscalMessageType | enum Constant | WithoutUplift |
|  |  |  |  |
| FPTR\_SS\_FULL\_LENGTH | FiscalSlipSelection | enum Constant | FullLength |
| FPTR\_SS\_VALIDATION | FiscalSlipSelection | enum Constant | Validation |
|  |  |  |  |
| FPTR\_TT\_DOCUMENT | FiscalTotalizerType | enum Constant | Document |
| FPTR\_TT\_DAY | FiscalTotalizerType | enum Constant | Day |
| FPTR\_TT\_RECEIPT | FiscalTotalizerType | enum Constant | Receipt |
| FPTR\_TT\_GRAND | FiscalTotalizerType | enum Constant | Grand |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| UnifiedPOS Name | POS for .NET | | |
| ClassName | Parameter | |
| Type | Name |
| FPTR\_GD\_CURRENT\_TOTAL | FiscalData | enum Constant | CurrentTotal |
| FPTR\_GD\_DAILY\_TOTAL | FiscalData | enum Constant | DailyTotal |
| FPTR\_GD\_RECEIPT\_NUMBER | FiscalData | enum Constant | ReceiptNumber |
| FPTR\_GD\_REFUND | FiscalData | enum Constant | Refund |
| FPTR\_GD\_NOT\_PAID | FiscalData | enum Constant | NotPaid |
| FPTR\_GD\_MID\_VOID | FiscalData | enum Constant | NumberOfVoidedReceipts |
| FPTR\_GD\_Z\_REPORT | FiscalData | enum Constant | ZReport |
| FPTR\_GD\_GRAND\_TOTAL | FiscalData | enum Constant | GrandTotal |
| FPTR\_GD\_PRINTER\_ID | FiscalData | enum Constant | PrinterId |
| FPTR\_GD\_FIRMWARE | FiscalData | enum Constant | Firmware |
| FPTR\_GD\_RESTART | FiscalData | enum Constant | Restart |
| FPTR\_GD\_REFUND\_VOID | FiscalData | enum Constant | RefundVoid |
| FPTR\_GD\_NUMB\_CONFIG\_BLOCK | FiscalData | enum Constant | NumberOfConfigurationBlocks |
| FPTR\_GD\_NUMB\_CURRENCY\_BLOCK | FiscalData | enum Constant | NumberOfCurrencyBlocks |
| FPTR\_GD\_NUMB\_HDR\_BLOCK | FiscalData | enum Constant | NumberOfHeaderBlocks |
| FPTR\_GD\_NUMB\_RESET\_BLOCK | FiscalData | enum Constant | NumberOfResetBlocks |
| FPTR\_GD\_NUMB\_VAT\_BLOCK | FiscalData | enum Constant | NumberOfVatBlocks |
| FPTR\_GD\_FISCAL\_DOC | FiscalData | enum Constant | FiscalDocument |
| FPTR\_GD\_FISCAL\_DOC\_VOID | FiscalData | enum Constant | FiscalDocumentVoid |
| FPTR\_GD\_FISCAL\_REC | FiscalData | enum Constant | FiscalReceipt |
| FPTR\_GD\_FISCAL\_REC\_VOID | FiscalData | enum Constant | FiscalReceiptVoid |
| FPTR\_GD\_NONFISCAL\_DOC | FiscalData | enum Constant | NonFiscalDocument |
| FPTR\_GD\_NONFISCAL\_DOC\_VOID | FiscalData | enum Constant | NonFiscalDocumentVoid |
| FPTR\_GD\_NONFISCAL\_REC | FiscalData | enum Constant | NonFiscalReceipt |
| FPTR\_GD\_SIMP\_INVOICE | FiscalData | enum Constant | SimplifiedInvoice |
| FPTR\_GD\_TENDER | FiscalData | enum Constant | Tender |
| FPTR\_GD\_LINECOUNT | FiscalData | enum Constant | LineCount |
| FPTR\_GD\_DESCRIPTION\_LENGTH | FiscalData | enum Constant | DescriptionLength |
|  |  |  |  |
| FPTR\_PDL\_CASH | FiscalPrinter | System.Int32 | PaymentDescriptionCash |
| FPTR\_PDL\_CHEQUE | FiscalPrinter | System.Int32 | PaymentDescriptionCheque |
| FPTR\_PDL\_CHITTY | FiscalPrinter | System.Int32 | PaymentDescriptionChitty |
| FPTR\_PDL\_COUPON | FiscalPrinter | System.Int32 | PaymentDescriptionCoupon |
| FPTR\_PDL\_CURRENCY | FiscalPrinter | System.Int32 | PaymentDescriptionCurrency |
| FPTR\_PDL\_DRIVEN\_OFF | FiscalPrinter | System.Int32 | PaymentDescriptionDrivenOff |
| FPTR\_PDL\_EFT\_IMPRINTER | FiscalPrinter | System.Int32 | PaymentDescriptionEftImprinter |
| FPTR\_PDL\_EFT\_TERMINAL | FiscalPrinter | System.Int32 | PaymentDescriptionEftTerminal |
| FPTR\_PDL\_TERMINAL\_IMPRINTER | FiscalPrinter | System.Int32 | PaymentDescriptionTerminalImprinter |
| FPTR\_PDL\_FREE\_GIFT | FiscalPrinter | System.Int32 | PaymentDescriptionFreeGift |
| FPTR\_PDL\_GIRO | FiscalPrinter | System.Int32 | PaymentDescriptionGiro |
| FPTR\_PDL\_HOME | FiscalPrinter | System.Int32 | PaymentDescriptionHome |
| FPTR\_PDL\_IMPRINTER\_WITH\_ISSUER | FiscalPrinter | System.Int32 | PaymentDescriptionImprinterWithIssuer |
| FPTR\_PDL\_LOCAL\_ACCOUNT | FiscalPrinter | System.Int32 | PaymentDescriptionLocalAccount |
| FPTR\_PDL\_LOCAL\_ACCOUNT\_CARD | FiscalPrinter | System.Int32 | PaymentDescriptionLocalAccountCard |
| FPTR\_PDL\_PAY\_CARD | FiscalPrinter | System.Int32 | PaymentDescriptionPayCard |
| FPTR\_PDL\_PAY\_CARD\_MANUAL | FiscalPrinter | System.Int32 | PaymentDescriptionPayCardManual |
| FPTR\_PDL\_PREPAY | FiscalPrinter | System.Int32 | PaymentDescriptionPrepay |
| FPTR\_PDL\_PUMP\_TEST | FiscalPrinter | System.Int32 | PaymentDescriptionPumpTest |
| FPTR\_PDL\_SHORT\_CREDIT | FiscalPrinter | System.Int32 | PaymentDescriptionShortCredit |
| FPTR\_PDL\_STAFF | FiscalPrinter | System.Int32 | PaymentDescriptionStaff |
| FPTR\_PDL\_VOUCHER | FiscalPrinter | System.Int32 | PaymentDescriptionVoucher |
|  |  |  |  |
| FPTR\_LC\_ITEM | FiscalPrinter | System.Int32 | LineCountItem |
| FPTR\_LC\_ITEM\_VOID | FiscalPrinter | System.Int32 | LineCountItemVoid |
| FPTR\_LC\_DISCOUNT | FiscalPrinter | System.Int32 | LineCountDiscount |
| FPTR\_LC\_DISCOUNT\_VOID | FiscalPrinter | System.Int32 | LineCountDiscountVoid |
| FPTR\_LC\_SURCHARGE | FiscalPrinter | System.Int32 | LineCountSurcharge |
| FPTR\_LC\_SURCHARGE\_VOID | FiscalPrinter | System.Int32 | LineCountSurchargeVoid |
| FPTR\_LC\_REFUND | FiscalPrinter | System.Int32 | LineCountRefund |
| FPTR\_LC\_REFUND\_VOID | FiscalPrinter | System.Int32 | LineCountRefundVoid |
| FPTR\_LC\_SUBTOTAL\_DISCOUNT | FiscalPrinter | System.Int32 | LineCountSubtotalDiscount |
| FPTR\_LC\_SUBTOTAL\_DISCOUNT\_VOID | FiscalPrinter | System.Int32 | LineCountSubtotalDiscountVoid |
| FPTR\_LC\_SUBTOTAL\_SURCHARGE | FiscalPrinter | System.Int32 | LineCountSubtotalSurcharge |
| FPTR\_LC\_SUBTOTAL\_SURCHARGE\_VOID | FiscalPrinter | System.Int32 | LineCountSubtotalSurchargeVoid |
| FPTR\_LC\_COMMENT | FiscalPrinter | System.Int32 | LineCountComment |
| FPTR\_LC\_SUBTOTAL | FiscalPrinter | System.Int32 | LineCountSubtotal |
| FPTR\_LC\_TOTAL | FiscalPrinter | System.Int32 | LineCountTotal |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| UnifiedPOS Name | POS for .NET | | |
| ClassName | Parameter | |
| Type | Name |
| FPTR\_DL\_ITEM | FiscalPrinter | System.Int32 | DescriptionLengthItem |
| FPTR\_DL\_ITEM\_ADJUSTMENT | FiscalPrinter | System.Int32 | DescriptionLengthItemAdjustment |
| FPTR\_DL\_ITEM\_FUEL | FiscalPrinter | System.Int32 | DescriptionLengthItemFuel |
| FPTR\_DL\_ITEM\_FUEL\_VOID | FiscalPrinter | System.Int32 | DescriptionLengthItemFuelVoid |
| FPTR\_DL\_NOT\_PAID | FiscalPrinter | System.Int32 | DescriptionLengthNotPaid |
| FPTR\_DL\_PACKAGE\_ADJUSTMENT | FiscalPrinter | System.Int32 | DescriptionLengthPackageAdjustment |
| FPTR\_DL\_REFUND | FiscalPrinter | System.Int32 | DescriptionLengthRefund |
| FPTR\_DL\_REFUND\_VOID | FiscalPrinter | System.Int32 | DescriptionLengthRefundVoid |
| FPTR\_DL\_SUBTOTAL\_ADJUSTMENT | FiscalPrinter | System.Int32 | DescriptionLengthSubtotalAdjustment |
| FPTR\_DL\_TOTAL | FiscalPrinter | System.Int32 | DescriptionLengthTotal |
| FPTR\_DL\_VOID | FiscalPrinter | System.Int32 | DescriptionLengthVoid |
| FPTR\_DL\_VOID\_ITEM | FiscalPrinter | System.Int32 | DescriptionLengthVoidItem |
|  |  |  |  |
| FPTR\_GT\_GROSS | FiscalTotalizer | enum Constant | Gross |
| FPTR\_GT\_NET | FiscalTotalizer | enum Constant | Net |
| FPTR\_GT\_DISCOUNT | FiscalTotalizer | enum Constant | Discount |
| FPTR\_GT\_DISCOUNT\_VOID | FiscalTotalizer | enum Constant | DiscountVoid |
| FPTR\_GT\_ITEM | FiscalTotalizer | enum Constant | Item |
| FPTR\_GT\_ITEM\_VOID | FiscalTotalizer | enum Constant | ItemVoid |
| FPTR\_GT\_NOT\_PAID | FiscalTotalizer | enum Constant | NotPaid |
| FPTR\_GT\_REFUND | FiscalTotalizer | enum Constant | Refund |
| FPTR\_GT\_REFUND\_VOID | FiscalTotalizer | enum Constant | RefundVoid |
| FPTR\_GT\_SUBTOTAL\_DISCOUNT | FiscalTotalizer | enum Constant | SubtotalDiscount |
| FPTR\_GT\_SUBTOTAL\_DISCOUNT\_VOID | FiscalTotalizer | enum Constant | SubtotalDiscountVoid |
| FPTR\_GT\_SUBTOTAL\_SURCHARGES | FiscalTotalizer | enum Constant | SubtotalSurcharges |
| FPTR\_GT\_SUBTOTAL\_SURCHARGES\_VOID | FiscalTotalizer | enum Constant | SubtotalSurchargesVoid |
| FPTR\_GT\_SURCHARGE | FiscalTotalizer | enum Constant | Surcharge |
| FPTR\_GT\_SURCHARGE\_VOID | FiscalTotalizer | enum Constant | SurchargeVoid |
| FPTR\_GT\_VAT | FiscalTotalizer | enum Constant | Vat |
| FPTR\_GT\_VAT\_CATEGORY | FiscalTotalizer | enum Constant | VatCategory |
|  |  |  |  |
| FPTR\_AT\_AMOUNT\_DISCOUNT | FiscalAdjustment | enum Constant | AmountDiscount |
| FPTR\_AT\_AMOUNT\_SURCHARGE | FiscalAdjustment | enum Constant | AmountSurcharge |
| FPTR\_AT\_PERCENTAGE\_DISCOUNT | FiscalAdjustment | enum Constant | PercentageDiscount |
| FPTR\_AT\_PERCENTAGE\_SURCHARGE | FiscalAdjustment | enum Constant | PercentageSurcharge |
|  |  |  |  |
| FPTR\_RT\_ORDINAL | FiscalReport | enum Constant | Ordinal |
| FPTR\_RT\_DATE | FiscalReport | enum Constant | Date |
|  |  |  |  |
| FPTR\_SC\_EURO | FiscalCurrency | enum Constant | Euro |
|  |  |  |  |
| FPTR\_SUE\_COVER\_OPEN | PrinterStatus | enum Constant | CoverOpen |
| FPTR\_SUE\_COVER\_OK | PrinterStatus | enum Constant | CoverOK |
| FPTR\_SUE\_JRN\_COVER\_OPEN | PrinterStatus | enum Constant | JournalCoverOpen |
| FPTR\_SUE\_JRN\_COVER\_OK | PrinterStatus | enum Constant | JournalCoverOK |
| FPTR\_SUE\_REC\_COVER\_OPEN | PrinterStatus | enum Constant | ReceiptCoverOpen |
| FPTR\_SUE\_REC\_COVER\_OK | PrinterStatus | enum Constant | ReceiptCoverOK |
| FPTR\_SUE\_SLP\_COVER\_OPEN | PrinterStatus | enum Constant | SlipCoverOpen |
| FPTR\_SUE\_SLP\_COVER\_OK | PrinterStatus | enum Constant | SlipCoverOK |
|  |  |  |  |
| FPTR\_SUE\_JRN\_EMPTY | PrinterStatus | enum Constant | JournalEmpty |
| FPTR\_SUE\_JRN\_NEAREMPTY | PrinterStatus | enum Constant | JournalNearEmpty |
| FPTR\_SUE\_JRN\_PAPEROK | PrinterStatus | enum Constant | JournalPaperOK |
|  |  |  |  |
| FPTR\_SUE\_REC\_EMPTY | PrinterStatus | enum Constant | ReceiptEmpty |
| FPTR\_SUE\_REC\_NEAREMPTY | PrinterStatus | enum Constant | ReceiptNearEmpty |
| FPTR\_SUE\_REC\_PAPEROK | PrinterStatus | enum Constant | ReceiptPaperOK |
|  |  |  |  |
| FPTR\_SUE\_SLP\_EMPTY | PrinterStatus | enum Constant | SlipEmpty |
| FPTR\_SUE\_SLP\_NEAREMPTY | PrinterStatus | enum Constant | SlipNearEmpty |
| FPTR\_SUE\_SLP\_PAPEROK | PrinterStatus | enum Constant | SlipPaperOK |
|  |  |  |  |
| FPTR\_SUE\_IDLE | PrinterStatus | enum Constant | Idle |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| UnifiedPOS Name | POS for .NET | | |
| ClassName | Parameter | |
| Type | Name |
| EFPTR\_COVER\_OPEN | FiscalPrinter | System.Int32 | ExtendedErrorCoverOpen |
| EFPTR\_JRN\_EMPTY | FiscalPrinter | System.Int32 | ExtendedErrorJournalEmpty |
| EFPTR\_REC\_EMPTY | FiscalPrinter | System.Int32 | ExtendedErrorReceiptEmpty |
| EFPTR\_SLP\_EMPTY | FiscalPrinter | System.Int32 | ExtendedErrorSlipEmpty |
| EFPTR\_SLP\_FORM | FiscalPrinter | System.Int32 | ExtendedErrorSlipForm |
| EFPTR\_MISSING\_DEVICES | FiscalPrinter | System.Int32 | ExtendedErrorMissingDevices |
| EFPTR\_WRONG\_STATE | FiscalPrinter | System.Int32 | ExtendedErrorWrongState |
| EFPTR\_TECHNICAL\_ASSISTANCE | FiscalPrinter | System.Int32 | ExtendedErrorTechnicalAssistance |
| EFPTR\_CLOCK\_ERROR | FiscalPrinter | System.Int32 | ExtendedErrorClockError |
| EFPTR\_FISCAL\_MEMORY\_DISCONNECTED | FiscalPrinter | System.Int32 | ExtendedErrorMemoryDisconnected |
| EFPTR\_FISCAL\_MEMORY\_FULL | FiscalPrinter | System.Int32 | ExtendedErrorMemoryFull |
| EFPTR\_FISCAL\_TOTALS\_ERROR | FiscalPrinter | System.Int32 | ExtendedErrorTotalsError |
| EFPTR\_BAD\_ITEM\_QUANTITY | FiscalPrinter | System.Int32 | ExtendedErrorBadItemQuantity |
| EFPTR\_BAD\_ITEM\_AMOUNT | FiscalPrinter | System.Int32 | ExtendedErrorBadItemAmount |
| EFPTR\_BAD\_ITEM\_DESCRIPTION | FiscalPrinter | System.Int32 | ExtendedErrorBadItemDescription |
| EFPTR\_RECEIPT\_TOTAL\_OVERFLOW | FiscalPrinter | System.Int32 | ExtendedErrorReceiptTotalOverflow |
| EFPTR\_BAD\_VAT | FiscalPrinter | System.Int32 | ExtendedErrorBadVat |
| EFPTR\_BAD\_PRICE | FiscalPrinter | System.Int32 | ExtendedErrorBadPrice |
| EFPTR\_BAD\_DATE | FiscalPrinter | System.Int32 | ExtendedErrorBadDate |
| EFPTR\_NEGATIVE\_TOTAL | FiscalPrinter | System.Int32 | ExtendedErrorNegativeTotal |
| EFPTR\_WORD\_NOT\_ALLOWED | FiscalPrinter | System.Int32 | ExtendedErrorWordNotAllowed |
| EFPTR\_BAD\_LENGTH | FiscalPrinter | System.Int32 | ExtendedErrorBadLength |
| EFPTR\_MISSING\_SET\_CURRENCY | FiscalPrinter | System.Int32 | ExtendedErrorMissingSetCurrency |
|  |  |  |  |
| KBD\_ET\_DOWN | KeyboardEventType | enum Constant | Down |
| KBD\_ET\_DOWN\_UP | KeyboardEventType | enum Constant | DownUp |
|  |  |  |  |
| KBD\_KET\_KEYDOWN | KeyEvent | enum Constant | Down |
| KBD\_KET\_KEYUP | KeyEvent | enum Constant | Up |
|  |  |  |  |
| LOCK\_KP\_ANY | Keylock | System.Int32 | PositionAny |
| LOCK\_KP\_LOCK | Keylock | System.Int32 | PositionLocked |
| LOCK\_KP\_NORM | Keylock | System.Int32 | PositionNormal |
| LOCK\_KP\_SUPR | Keylock | System.Int32 | PositionSupervisor |
|  |  |  |  |
| MICR\_CT\_PERSONAL | CheckType | enum Constant | Personal |
| MICR\_CT\_BUSINESS | CheckType | enum Constant | Business |
| MICR\_CT\_UNKNOWN | CheckType | enum Constant | Unknown |
|  |  |  |  |
| MICR\_CC\_USA | CheckCountryCode | enum Constant | Usa |
| MICR\_CC\_CANADA | CheckCountryCode | enum Constant | Canada |
| MICR\_CC\_MEXICO | CheckCountryCode | enum Constant | Mexico |
| MICR\_CC\_UNKNOWN | CheckCountryCode | enum Constant | Unknown Check Font E-13B |
| MICR\_CC\_CMC7 | CheckCountryCode | enum Constant | Unknown Check Font CMC-7 |
| MICR\_CC\_OTHER | CheckCountryCode | enum Constant | Unknown Check Font OCR-A or OCR\_B |
|  |  |  |  |
| EMICR\_NOCHECK | Micr | System.Int32 | ExtendedErrorNoCheck |
| EMICR\_CHECK | Micr | System.Int32 | ExtendedErrorCheck |
| EMICR\_BADDATA | Micr | System.Int32 | ExtendedErrorBadData |
| EMICR\_NODATA | Micr | System.Int32 | ExtendedErrorNoData |
| EMICR\_BADSIZE | Micr | System.Int32 | ExtendedErrorBadSize |
| EMICR\_JAM | Micr | System.Int32 | ExtendedErrorJam |
| EMICR\_CHECKDIGIT | Micr | System.Int32 | ExtendedErrorCheckDigit |
| EMICR\_COVEROPEN | Micr | System.Int32 | ExtendedErrorCoverOpen |
|  |  |  |  |
| MOTION\_M\_PRESENT | MotionSensor | System.Int32 | StatusMotionPresent |
| MOTION\_M\_ABSENT | MotionSensor | System.Int32 | StatusMotionAbsent |
|  |  |  |  |
| MSR\_TR\_1 | MsrTracks | enum Constant | Track1 |
| MSR\_TR\_2 | MsrTracks | enum Constant | Track2 |
| MSR\_TR\_3 | MsrTracks | enum Constant | Track3 |
| MSR\_TR\_4 | MsrTracks | enum Constant | Track4 |
| MSR\_TR\_1\_2 | MsrTracks | enum Constant | Tracks12 |
| MSR\_TR\_1\_3 | MsrTracks | enum Constant | Tracks13 |
| MSR\_TR\_1\_4 | MsrTracks | enum Constant | Tracks14 |
| MSR\_TR\_2\_3 | MsrTracks | enum Constant | Tracks23 |
| MSR\_TR\_2\_4 | MsrTracks | enum Constant | Tracks24 |
| MSR\_TR\_3\_4 | MsrTracks | enum Constant | Tracks34 |
| MSR\_TR\_1\_2\_3 | MsrTracks | enum Constant | Tracks123 |
| MSR\_TR\_1\_2\_4 | MsrTracks | enum Constant | Tracks124 |
| MSR\_TR\_1\_3\_4 | MsrTracks | enum Constant | Tracks134 |
| MSR\_TR\_2\_3\_4 | MsrTracks | enum Constant | Tracks234 |
| MSR\_TR\_1\_2\_3\_4 | MsrTracks | enum Constant | Tracks1234 |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| UnifiedPOS Name | POS for .NET | | |
| ClassName | Parameter | |
| Type | Name |
| MSR\_ERT\_CARD | MsrErrorReporting | enum Constant | Card |
| MSR\_ERT\_TRACK | MsrErrorReporting | enum Constant | Track |
|  |  |  |  |
| No\_Equivalent\_Defined | Msr | System.Int32 | ExtendedErrorSuccess |
| No\_Equivalent\_Defined | Msr | System.Int32 | ExtendedErrorFailure |
| EMSR\_START | Msr | System.Int32 | ExtendedErrorStart |
| EMSR\_END | Msr | System.Int32 | ExtendedErrorEnd |
| EMSR\_PARITY | Msr | System.Int32 | ExtendedErrorParity |
| EMSR\_LRC | Msr | System.Int32 | ExtendedErrorLrc |
|  |  |  |  |
| No\_Equivalent\_Defined | CharacterSetCapability | enum Constant | Numeric |
| PCRW\_CCS\_ALPHA | CharacterSetCapability | enum Constant | Alpha |
| PCRW\_CCS\_ASCII | CharacterSetCapability | enum Constant | Ascii |
| PCRW\_CCS\_KANA | CharacterSetCapability | enum Constant | Kana |
| PCRW\_CCS\_KANJI | CharacterSetCapability | enum Constant | Kanji |
| PCRW\_CCS\_UNICODE | CharacterSetCapability | enum Constant | Unicode |
|  |  |  |  |
| PCRW\_STATE\_NOCARD | PointCardState | enum Constant | NoCard |
| PCRW\_STATE\_REMAINING | PointCardState | enum Constant | Remaining |
| PCRW\_STATE\_INRW | PointCardState | enum Constant | Inserted |
|  |  |  |  |
| PCRW\_TRACK1 | PointCardRWTracks | enum Constant | Track1 |
| PCRW\_TRACK2 | PointCardRWTracks | enum Constant | Track2 |
| PCRW\_TRACK3 | PointCardRWTracks | enum Constant | Track3 |
| PCRW\_TRACK4 | PointCardRWTracks | enum Constant | Track4 |
| PCRW\_TRACK5 | PointCardRWTracks | enum Constant | Track5 |
| PCRW\_TRACK6 | PointCardRWTracks | enum Constant | Track6 |
|  |  |  |  |
| PCRW\_CS\_UNICODE | PosCommon | System.Int32 | CharacterSetUnicode |
| PCRW\_CS\_ASCII | PosCommon | System.Int32 | CharacterSetAscii |
| PCRW\_CS\_WINDOWS | PosCommon | System.Int32 | No\_Equivalent\_Defined |
| PCRW\_CS\_ANSI | PosCommon | System.Int32 | CharacterSetAnsi |
|  |  |  |  |
| PCRW\_MM\_DOTS | MapMode | enum Constant | Dots |
| PCRW\_MM\_TWIPS | MapMode | enum Constant | Twips |
| PCRW\_MM\_ENGLISH | MapMode | enum Constant | English |
| PCRW\_MM\_METRIC | MapMode | enum Constant | Metric |
|  |  |  |  |
| EPCRW\_READ | PointCardRW | System.Int32 | ExtendedErrorRead |
| EPCRW\_WRITE | PointCardRW | System.Int32 | ExtendedErrorWrite |
| EPCRW\_JAM | PointCardRW | System.Int32 | ExtendedErrorJam |
| EPCRW\_MOTOR | PointCardRW | System.Int32 | ExtendedErrorMotor |
| EPCRW\_COVER | PointCardRW | System.Int32 | ExtendedErrorCover |
| EPCRW\_PRINTER | PointCardRW | System.Int32 | ExtendedErrorPrinter |
| EPCRW\_RELEASE | PointCardRW | System.Int32 | ExtendedErrorRelease |
| EPCRW\_DISPLAY | PointCardRW | System.Int32 | ExtendedErrorDisplay |
| EPCRW\_NOCARD | PointCardRW | System.Int32 | ExtendedErrorNoCard |
|  |  |  |  |
| No\_Equivalent\_Defined | PointCardReadWriteState | enum Constant | Success |
| EPCRW\_START | PointCardReadWriteState | enum Constant | Start |
| EPCRW\_END | PointCardReadWriteState | enum Constant | End |
| EPCRW\_PARITY | PointCardReadWriteState | enum Constant | Parity |
| EPCRW\_ENCODE | PointCardReadWriteState | enum Constant | Encode |
| EPCRW\_LRC | PointCardReadWriteState | enum Constant | LrcError |
| EPCRW\_VERIFY | PointCardReadWriteState | enum Constant | Verify |
| No\_Equivalent\_Defined | PointCardReadWriteState | enum Constant | Failure |
|  |  |  |  |
| PCRW\_RP\_NORMAL | PrintRotation | enum Constant | Normal |
| PCRW\_RP\_RIGHT90 | PrintRotation | enum Constant | Right90 |
| PCRW\_RP\_LEFT90 | PrintRotation | enum Constant | Left90 |
| PCRW\_RP\_ROTATE180 | PrintRotation | enum Constant | Rotate180 |
|  |  |  |  |
| PCRW\_SUE\_NOCARD | PointCardRW | System.Int32 | StatusNoCard |
| PCRW\_SUE\_REMAINING | PointCardRW | System.Int32 | StatusRemaining |
| PCRW\_SUE\_INRW | PointCardRW | System.Int32 | StatusInserted |
|  |  |  |  |
| No\_Equivalent\_Defined | PointCardKinds | enum Constant | PrintingArea |
| No\_Equivalent\_Defined | PointCardKinds | enum Constant | MagneticTracks |
| No\_Equivalent\_Defined | PointCardKinds | enum Constant | PrintingAreaAndMagneticTracks |
|  |  |  |  |
| PPAD\_DISP\_UNRESTRICTED | PinPadDisplay | enum Constant | Unrestricted |
| PPAD\_DISP\_PINRESTRICTED | PinPadDisplay | enum Constant | PinRestricted |
| PPAD\_DISP\_RESTRICTED\_LIST | PinPadDisplay | enum Constant | RestrictedList |
| PPAD\_DISP\_RESTRICTED\_ORDER | PinPadDisplay | enum Constant | RestrictedOrder |
| PPAD\_DISP\_NONE | PinPadDisplay | enum Constant | None |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| UnifiedPOS Name | POS for .NET | | |
| ClassName | Parameter | |
| Type | Name |
| PPAD\_MSG\_ENTERPIN | PinPadMessage | enum Constant | EnterPin |
| PPAD\_MSG\_PLEASEWAIT | PinPadMessage | enum Constant | PleaseWait |
| PPAD\_MSG\_ENTERVALIDPIN | PinPadMessage | enum Constant | EnterValidPin |
| PPAD\_MSG\_RETRIESEXCEEDED | PinPadMessage | enum Constant | RetriesExceeded |
| PPAD\_MSG\_APPROVED | PinPadMessage | enum Constant | Approved |
| PPAD\_MSG\_DECLINED | PinPadMessage | enum Constant | Declined |
| PPAD\_MSG\_CANCELED | PinPadMessage | enum Constant | Canceled |
| PPAD\_MSG\_AMOUNTOK | PinPadMessage | enum Constant | AmountOK |
| PPAD\_MSG\_NOTREADY | PinPadMessage | enum Constant | NotReady |
| PPAD\_MSG\_IDLE | PinPadMessage | enum Constant | Idle |
| PPAD\_MSG\_SLIDE\_CARD | PinPadMessage | enum Constant | SlideCard |
| PPAD\_MSG\_INSERTCARD | PinPadMessage | enum Constant | InsertCard |
| PPAD\_MSG\_SELECTCARDTYPE | PinPadMessage | enum Constant | SelectCardType |
| PPAD\_LANG\_NONE | PinPadLanguage | enum Constant | None |
| PPAD\_LANG\_ONE | PinPadLanguage | enum Constant | One |
| PPAD\_LANG\_PINRESTRICTED | PinPadLanguage | enum Constant | PinRestricted |
| PPAD\_LANG\_UNRESTRICTED | PinPadLanguage | enum Constant | Unrestricted |
|  |  |  |  |
| PPAD\_TRANS\_DEBIT | EftTransactionType | enum Constant | Debit |
| PPAD\_TRANS\_CREDIT | EftTransactionType | enum Constant | Credit |
| PPAD\_TRANS\_INQ | EftTransactionType | enum Constant | Inquiry |
| PPAD\_TRANS\_RECONCILE | EftTransactionType | enum Constant | Reconcile |
| PPAD\_TRANS\_ADMIN | EftTransactionType | enum Constant | Admin |
|  |  |  |  |
| PPAD\_EFT\_NORMAL | EftTransactionControl | enum Constant | Normal |
| PPAD\_EFT\_ABNORMAL | EftTransactionControl | enum Constant | Abnormal |
|  |  |  |  |
| PPAD\_SUCCESS | PinEntryStatus | enum Constant | Success |
| PPAD\_CANCEL | PinEntryStatus | enum Constant | Cancel |
| No\_Equivalent\_Defined | PinEntryStatus | enum Constant | Timeout |
| No\_Equivalent\_Defined | PinEntryStatus | enum Constant | BadKey |
| No\_Equivalent\_Defined | PinPadSystem | enum Constant | MasterSession |
| No\_Equivalent\_Defined | PinPadSystem | enum Constant | Dukpt |
| No\_Equivalent\_Defined | PinPadSystem | enum Constant | Apacs40 |
| No\_Equivalent\_Defined | PinPadSystem | enum Constant | AS2805 |
| No\_Equivalent\_Defined | PinPadSystem | enum Constant | Hgepos |
| No\_Equivalent\_Defined | PinPadSystem | enum Constant | Jdebit2 |
|  |  |  |  |
| EPPAD\_BAD\_KEY | PinPad | System.Int32 | ExtendedErrorBadKey |
|  |  |  |  |
| No\_Equivalent\_Defined | PrinterStation | enum Constant | None |
| PTR\_S\_JOURNAL | PrinterStation | enum Constant | Journal |
| PTR\_S\_RECEIPT | PrinterStation | enum Constant | Receipt |
| PTR\_S\_SLIP | PrinterStation | enum Constant | Slip |
|  |  |  |  |
| PTR\_S\_JOURNAL\_RECEIPT | FiscalPrinterStations | enum Constant | JournalReceipt |
| PTR\_S\_JOURNAL\_SLIP | FiscalPrinterStations | enum Constant | JournalSlip |
| PTR\_S\_RECEIPT\_SLIP | FiscalPrinterStations | enum Constant | ReceiptSlip |
|  |  |  |  |
| PTR\_TWO\_RECEIPT\_JOURNAL | PrinterStation | enum Constant | TwoReceiptJournal |
| PTR\_TWO\_SLIP\_JOURNAL | PrinterStation | enum Constant | TwoSlipJournal |
| PTR\_TWO\_SLIP\_RECEIPT | PrinterStation | enum Constant | TwoSlipReceipt |
|  |  |  |  |
| No\_Equivalent\_Defined | CharacterSetCapability | enum Constant | Numeric |
| PTR\_CCS\_ALPHA | CharacterSetCapability | enum Constant | Alpha |
| PTR\_CCS\_ASCII | CharacterSetCapability | enum Constant | Ascii |
| PTR\_CCS\_KANA | CharacterSetCapability | enum Constant | Kana |
| PTR\_CCS\_KANJI | CharacterSetCapability | enum Constant | Kanji |
| PTR\_CCS\_UNICODE | CharacterSetCapability | enum Constant | Unicode |
|  |  |  |  |
| PTR\_CS\_UNICODE | PosCommon | System.Int32 | CharacterSetUnicode |
| PTR\_CS\_ASCII | PosCommon | System.Int32 | CharacterSetAscii |
| PTR\_CS\_WINDOWS | PosCommon | System.Int32 | No\_Equivalent\_Defined |
| PTR\_CS\_ANSI | PosCommon | System.Int32 | CharacterSetAnsi |
|  |  |  |  |
| PTR\_EL\_NONE | PrinterErrorLevel | enum Constant | None |
| PTR\_EL\_RECOVERABLE | PrinterErrorLevel | enum Constant | Recoverable |
| PTR\_EL\_FATAL | PrinterErrorLevel | enum Constant | Fatal |
|  |  |  |  |
| PTR\_MM\_DOTS | MapMode | enum Constant | Dots |
| PTR\_MM\_TWIPS | MapMode | enum Constant | Twips |
| PTR\_MM\_ENGLISH | MapMode | enum Constant | English |
| PTR\_MM\_METRIC | MapMode | enum Constant | Metric |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| UnifiedPOS Name | POS for .NET | | |
| ClassName | Parameter | |
| Type | Name |
| No\_Equivalent\_Defined | PrinterColors | enum Constant | None |
| PTR\_COLOR\_PRIMARY | PrinterColors | enum Constant | Primary |
| PTR\_COLOR\_CUSTOM1 | PrinterColors | enum Constant | Custom1 |
| PTR\_COLOR\_CUSTOM2 | PrinterColors | enum Constant | Custom2 |
| PTR\_COLOR\_CUSTOM3 | PrinterColors | enum Constant | Custom3 |
| PTR\_COLOR\_CUSTOM4 | PrinterColors | enum Constant | Custom4 |
| PTR\_COLOR\_CUSTOM5 | PrinterColors | enum Constant | Custom5 |
| PTR\_COLOR\_CUSTOM6 | PrinterColors | enum Constant | Custom6 |
| PTR\_COLOR\_CYAN | PrinterColors | enum Constant | Cyan |
| PTR\_COLOR\_MAGENTA | PrinterColors | enum Constant | Magenta |
| PTR\_COLOR\_YELLOW | PrinterColors | enum Constant | Yellow |
| PTR\_COLOR\_FULL | PrinterColors | enum Constant | Full |
|  |  |  |  |
| PTR\_CART\_UNKNOWN | PrinterCartridgeStates | enum Constant | Unknown |
| PTR\_CART\_OK | PrinterCartridgeStates | enum Constant | OK |
| PTR\_CART\_REMOVED | PrinterCartridgeStates | enum Constant | Removed |
| PTR\_CART\_EMPTY | PrinterCartridgeStates | enum Constant | Empty |
| PTR\_CART\_NEAREND | PrinterCartridgeStates | enum Constant | NearEnd |
| PTR\_CART\_CLEANING | PrinterCartridgeStates | enum Constant | Cleaning |
| PTR\_CN\_DISABLED | PrinterCartridgeNotify | enum Constant | Disabled |
| PTR\_CN\_ENABLED | PrinterCartridgeNotify | enum Constant | Enabled |
|  |  |  |  |
| PTR\_CP\_FULLCUT | PosPrinter | System.Int32 | PrinterCutPaperFullCut |
| PTR\_BC\_LEFT | PosPrinter | System.Int32 | PrinterBarCodeLeft |
| PTR\_BC\_CENTER | PosPrinter | System.Int32 | PrinterBarCodeCenter |
| PTR\_BC\_RIGHT | PosPrinter | System.Int32 | PrinterBarCodeRight |
|  |  |  |  |
| PTR\_BC\_TEXT\_NONE | BarCodeTextPosition | enum Constant | None |
| PTR\_BC\_TEXT\_ABOVE | BarCodeTextPosition | enum Constant | Above |
| PTR\_BC\_TEXT\_BELOW | BarCodeTextPosition | enum Constant | Below |
|  |  |  |  |
| No\_Equivalent\_Defined | BarCodeSymbology | enum Constant | Unknown |
| PTR\_BCS\_UPCA | BarCodeSymbology | enum Constant | Upca |
| PTR\_BCS\_UPCE | BarCodeSymbology | enum Constant | Upce |
| PTR\_BCS\_JAN8 | BarCodeSymbology | enum Constant | EanJan8 |
| PTR\_BCS\_EAN8 | BarCodeSymbology | enum Constant | No\_Equivalent\_Defined |
| PTR\_BCS\_JAN13 | BarCodeSymbology | enum Constant | EanJan13 |
| PTR\_BCS\_EAN13 | BarCodeSymbology | enum Constant | No\_Equivalent\_Defined |
| PTR\_BCS\_TF | BarCodeSymbology | enum Constant | TF |
| PTR\_BCS\_ITF | BarCodeSymbology | enum Constant | Itf |
| PTR\_BCS\_Codabar | BarCodeSymbology | enum Constant | Codabar |
| PTR\_BCS\_Code39 | BarCodeSymbology | enum Constant | Code39 |
| PTR\_BCS\_Code93 | BarCodeSymbology | enum Constant | Code93 |
| PTR\_BCS\_Code128 | BarCodeSymbology | enum Constant | Code128 |
| PTR\_BCS\_UPCA\_S | BarCodeSymbology | enum Constant | Upcas |
| PTR\_BCS\_UPCE\_S | BarCodeSymbology | enum Constant | Upces |
| PTR\_BCS\_UPCD1 | BarCodeSymbology | enum Constant | Upcd1 |
| PTR\_BCS\_UPCD2 | BarCodeSymbology | enum Constant | Upcd2 |
| PTR\_BCS\_UPCD3 | BarCodeSymbology | enum Constant | Upcd3 |
| PTR\_BCS\_UPCD4 | BarCodeSymbology | enum Constant | Upcd4 |
| PTR\_BCS\_UPCD5 | BarCodeSymbology | enum Constant | Upcd5 |
| PTR\_BCS\_EAN8\_S | BarCodeSymbology | enum Constant | Ean8S |
| PTR\_BCS\_EAN13\_S | BarCodeSymbology | enum Constant | Ean13S |
| PTR\_BCS\_EAN128 | BarCodeSymbology | enum Constant | Ean128 |
| PTR\_BCS\_OCRA | BarCodeSymbology | enum Constant | Ocra |
| PTR\_BCS\_OCRB | BarCodeSymbology | enum Constant | Ocrb |
| PTR\_BCS\_Code128\_Parsed | BarCodeSymbology | enum Constant | Code128 Parsed |
| PTR\_BCS\_RSS14 | BarCodeSymbology | enum Constant | Rss14 – Deprecated v1.12 |
| PTR\_BCS\_RSS\_EXPANDED | BarCodeSymbology | enum Constant | RssExpanded – Deprecated v1.12 |
| PTR\_BCS\_GS1DATABAR | BarCodeSymbology | enum Constant | GS1 DataBar Omnidirectional |
| PTR\_BCS\_GS1DATABAR\_S | BarCodeSymbology | enum Constant | GS1 DataBar Stacked Omnidirectional |
| PTR\_BCS\_GS1DATABAR\_E | BarCodeSymbology | enum Constant | GS1 DataBar Expanded |
| PTR\_BCS\_GS1DATABAR\_E\_S | BarCodeSymbology | enum Constant | GS1 DataBar Expanded Stacked |
| No\_Equivalent\_Defined | BarCodeSymbology | enum Constant | Cca |
| No\_Equivalent\_Defined | BarCodeSymbology | enum Constant | Ccb |
| No\_Equivalent\_Defined | BarCodeSymbology | enum Constant | Ccc |
| PTR\_BCS\_PDF417 | BarCodeSymbology | enum Constant | Pdf417 |
| PTR\_BCS\_MAXICODE | BarCodeSymbology | enum Constant | Maxicode |
| PTR\_BCS\_DATAMATRIX | BarCodeSymbology | enum Constant | Data Matrix |
| PTR\_BCS-QRCODE | BarCodeSymbology | enum Constant | QR Code |
| PTR\_BCS\_UQRCODE | BarCodeSymbology | enum Constant | Micro QR Code |
| PTR\_BCS\_AXTEC | BarCodeSymbology | enum Constant | Axtec |
| PTR\_BCS\_UPDF417 | BarCodeSymbology | enum Constant | Micro Pdf417 |
| PTR\_BCS\_OTHER | BarCodeSymbology | enum Constant | Other |

|  |  |  |  |
| --- | --- | --- | --- |
| UnifiedPOS Name | POS for .NET | | |
| ClassName | Parameter | |
| Type | Name |
| PTR\_BM\_ASIS | PosPrinter | System.Int32 | PrinterBitmapAsIs |
| PTR\_BM\_LEFT | PosPrinter | System.Int32 | PrinterBitmapLeft |
| PTR\_BM\_CENTER | PosPrinter | System.Int32 | PrinterBitmapCenter |
| PTR\_BM\_RIGHT | PosPrinter | System.Int32 | PrinterBitmapRight |
|  |  |  |  |
| PTR\_RP\_NORMAL | PrintRotation | enum Constant | Normal |
| PTR\_RP\_RIGHT90 | PrintRotation | enum Constant | Right90 |
| PTR\_RP\_LEFT90 | PrintRotation | enum Constant | Left90 |
| PTR\_RP\_ROTATE180 | PrintRotation | enum Constant | Rotate180 |
| PTR\_RP\_BARCODE | PrintRotation | enum Constant | Barcode |
| PTR\_RP\_BITMAP | PrintRotation | enum Constant | Bitmap |
|  |  |  |  |
| PTR\_L\_TOP | PrinterLogoLocation | enum Constant | Top |
| PTR\_L\_BOTTOM | PrinterLogoLocation | enum Constant | Bottom |
|  |  |  |  |
| PTR\_TP\_TRANSACTION | PrinterTransactionControl | enum Constant | Transaction |
| PTR\_TP\_NORMAL | PrinterTransactionControl | enum Constant | Normal |
|  |  |  |  |
| No\_Equivalent\_Defined | PrinterMarkFeeds | enum Constant | None |
| PTR\_MF\_TO\_TAKEUP | PrinterMarkFeeds | enum Constant | Takeup |
| PTR\_MF\_TO\_CUTTER | PrinterMarkFeeds | enum Constant | Cutter |
| PTR\_MF\_TO\_CURRENT\_TOF | PrinterMarkFeeds | enum Constant | CurrentTopOfForm |
| PTR\_MF\_TO\_NEXT\_TOF | PrinterMarkFeeds | enum Constant | NextTopOfForm |
|  |  |  |  |
| PTR\_PS\_UNKNOWN | PrinterSide | enum Constant | Unknown |
| PTR\_PS\_SIDE1 | PrinterSide | enum Constant | Side1 |
| PTR\_PS\_SIDE2 | PrinterSide | enum Constant | Side2 |
| PTR\_PS\_OPPOSITE | PrinterSide | enum Constant | Opposite |
|  |  |  |  |
| PTR\_SUE\_COVER\_OPEN | PrinterStatus | enum Constant | CoverOpen |
| PTR\_SUE\_COVER\_OK | PrinterStatus | enum Constant | CoverOK |
| PTR\_SUE\_JRN\_EMPTY | PrinterStatus | enum Constant | JournalEmpty |
| PTR\_SUE\_JRN\_NEAREMPTY | PrinterStatus | enum Constant | JournalNearEmpty |
| PTR\_SUE\_JRN\_PAPEROK | PrinterStatus | enum Constant | JournalPaperOK |
| PTR\_SUE\_REC\_EMPTY | PrinterStatus | enum Constant | ReceiptEmpty |
| PTR\_SUE\_REC\_NEAREMPTY | PrinterStatus | enum Constant | ReceiptNearEmpty |
| PTR\_SUE\_REC\_PAPEROK | PrinterStatus | enum Constant | ReceiptPaperOK |
| PTR\_SUE\_SLP\_EMPTY | PrinterStatus | enum Constant | SlipEmpty |
| PTR\_SUE\_SLP\_NEAREMPTY | PrinterStatus | enum Constant | SlipNearEmpty |
| PTR\_SUE\_SLP\_PAPEROK | PrinterStatus | enum Constant | SlipPaperOK |
| PTR\_SUE\_JRN\_CARTRIDGE\_EMPTY | PrinterStatus | enum Constant | JournalCartridgeEmpty |
| PTR\_SUE\_JRN\_CARTRIDGE\_NEAREMPTY | PrinterStatus | enum Constant | JournalCartridgeNearEmpty |
| PTR\_SUE\_JRN\_HEAD\_CLEANING | PrinterStatus | enum Constant | JournalHeadCleaning |
| PTR\_SUE\_JRN\_CARTRIDGE\_OK | PrinterStatus | enum Constant | JournalCartridgeOK |
| PTR\_SUE\_REC\_CARTRIDGE\_EMPTY | PrinterStatus | enum Constant | ReceiptCartridgeEmpty |
| PTR\_SUE\_REC\_CARTRIDGE\_NEAREMPTY | PrinterStatus | enum Constant | ReceiptCartridgeNearEmpty |
| PTR\_SUE\_REC\_HEAD\_CLEANING | PrinterStatus | enum Constant | ReceiptHeadCleaning |
| PTR\_SUE\_REC\_CARTRIDGE\_OK | PrinterStatus | enum Constant | ReceiptCartridgeOK |
| PTR\_SUE\_SLP\_CARTRIDGE\_EMPTY | PrinterStatus | enum Constant | SlipCartridgeEmpty |
| PTR\_SUE\_SLP\_CARTRIDGE\_NEAREMPTY | PrinterStatus | enum Constant | SlipCartridgeNearEmpty |
| PTR\_SUE\_SLP\_HEAD\_CLEANING | PrinterStatus | enum Constant | SlipHeadCleaning |
| PTR\_SUE\_SLP\_CARTRIDGE\_OK | PrinterStatus | enum Constant | SlipCartridgeOK |
| PTR\_SUE\_JRN\_COVER\_OPEN | PrinterStatus | enum Constant | JournalCoverOpen |
| PTR\_SUE\_JRN\_COVER\_OK | PrinterStatus | enum Constant | JournalCoverOK |
| PTR\_SUE\_REC\_COVER\_OPEN | PrinterStatus | enum Constant | ReceiptCoverOpen |
| PTR\_SUE\_REC\_COVER\_OK | PrinterStatus | enum Constant | ReceiptCoverOK |
| PTR\_SUE\_SLP\_COVER\_OPEN | PrinterStatus | enum Constant | SlipCoverOpen |
| PTR\_SUE\_SLP\_COVER\_OK | PrinterStatus | enum Constant | SlipCoverOK |
| PTR\_SUE\_IDLE | PrinterStatus | enum Constant | Idle |
|  |  |  |  |
| EPTR\_COVER\_OPEN | PosPrinter | System.Int32 | ExtendedErrorCoverOpen |
|  |  |  |  |
| EPTR\_JRN\_EMPTY | PosPrinter | System.Int32 | ExtendedErrorJrnEmpty |
| EPTR\_REC\_EMPTY | PosPrinter | System.Int32 | ExtendedErrorRecEmpty |
| EPTR\_SLP\_EMPTY | PosPrinter | System.Int32 | ExtendedErrorSlpEmpty |
| EPTR\_SLP\_FORM | PosPrinter | System.Int32 | ExtendedErrorSlpForm |
| EPTR\_TOOBIG | PosPrinter | System.Int32 | ExtendedErrorTooBig |
| EPTR\_BADFORMAT | PosPrinter | System.Int32 | ExtendedErrorBadFormat |

|  |  |  |  |
| --- | --- | --- | --- |
| UnifiedPOS Name | POS for .NET | | |
| ClassName | Parameter | |
| Type | Name |
| EPTR\_JRN\_CARTRIDGE\_REMOVED | PosPrinter | System.Int32 | ExtendedErrorJrnCartridgeRemoved |
| EPTR\_JRN\_CARTRIDGE\_EMPTY | PosPrinter | System.Int32 | ExtendedErrorJrnCartridgeEmpty |
| EPTR\_JRN\_HEAD\_CLEANING | PosPrinter | System.Int32 | ExtendedErrorJrnHeadCleaning |
| EPTR\_REC\_CARTRIDGE\_REMOVED | PosPrinter | System.Int32 | ExtendedErrorRecCartridgeRemoved |
| EPTR\_REC\_CARTRIDGE\_EMPTY | PosPrinter | System.Int32 | ExtendedErrorRecCartridgeEmpty |
| EPTR\_REC\_HEAD\_CLEANING | PosPrinter | System.Int32 | ExtendedErrorRecHeadCleaning |
| EPTR\_SLP\_CARTRIDGE\_REMOVED | PosPrinter | System.Int32 | ExtendedErrorSlpCartridgeRemoved |
| EPTR\_SLP\_CARTRIDGE\_EMPTY | PosPrinter | System.Int32 | ExtendedErrorSlpCartridgeEmpty |
| EPTR\_SLP\_HEAD\_CLEANING | PosPrinter | System.Int32 | ExtendedErrorSlpHeadCleaning |
|  |  |  |  |
| PWR\_UPS\_FULL | UpsChargeStates | enum Constant | Full |
| PWR\_UPS\_WARNING | UpsChargeStates | enum Constant | Warning |
| PWR\_UPS\_LOW | UpsChargeStates | enum Constant | Low |
| PWR\_UPS\_CRITICAL | UpsChargeStates | enum Constant | Critical |
|  |  |  |  |
| PWR\_SUE\_UPS\_FULL | PosPower | System.Int32 | StatusUpsFull |
| PWR\_SUE\_UPS\_WARNING | PosPower | System.Int32 | StatusUpsWarning |
| PWR\_SUE\_UPS\_LOW | PosPower | System.Int32 | StatusUpsLow |
| PWR\_SUE\_UPS\_CRITICAL | PosPower | System.Int32 | StatusUpsCritical |
| PWR\_SUE\_FAN\_STOPPED | PosPower | System.Int32 | StatusFanStopped |
| PWR\_SUE\_FAN\_RUNNING | PosPower | System.Int32 | StatusFanRunning |
| PWR\_SUE\_TEMPERATURE\_HIGH | PosPower | System.Int32 | StatusTemperatureHigh |
| PWR\_SUE\_TEMPERATURE\_OK | PosPower | System.Int32 | StatusTemperatureOK |
| PWR\_SUE\_SHUTDOWN | PosPower | System.Int32 | StatusShutDown |
|  |  |  |  |
| ROD\_UID\_1 | DeviceUnits | enum Constant | nit1 |
| ROD\_UID\_2 | DeviceUnits | enum Constant | Unit2 |
| ROD\_UID\_3 | DeviceUnits | enum Constant | Unit3 |
| ROD\_UID\_4 | DeviceUnits | enum Constant | Unit4 |
| ROD\_UID\_5 | DeviceUnits | enum Constant | Unit5 |
| ROD\_UID\_6 | DeviceUnits | enum Constant | Unit6 |
| ROD\_UID\_7 | DeviceUnits | enum Constant | Unit7 |
| ROD\_UID\_8 | DeviceUnits | enum Constant | Unit8 |
| ROD\_UID\_9 | DeviceUnits | enum Constant | Unit9 |
| ROD\_UID\_10 | DeviceUnits | enum Constant | Unit10 |
| ROD\_UID\_11 | DeviceUnits | enum Constant | Unit11 |
| ROD\_UID\_12 | DeviceUnits | enum Constant | Unit12 |
| ROD\_UID\_13 | DeviceUnits | enum Constant | Unit13 |
| ROD\_UID\_14 | DeviceUnits | enum Constant | Unit14 |
| ROD\_UID\_15 | DeviceUnits | enum Constant | Unit15 |
| ROD\_UID\_16 | DeviceUnits | enum Constant | Unit16 |
| ROD\_UID\_17 | DeviceUnits | enum Constant | Unit17 |
| ROD\_UID\_18 | DeviceUnits | enum Constant | Unit18 |
| ROD\_UID\_19 | DeviceUnits | enum Constant | Unit19 |
| ROD\_UID\_20 | DeviceUnits | enum Constant | Unit20 |
| ROD\_UID\_21 | DeviceUnits | enum Constant | Unit21 |
| ROD\_UID\_22 | DeviceUnits | enum Constant | Unit22 |
| ROD\_UID\_23 | DeviceUnits | enum Constant | Unit23 |
| ROD\_UID\_24 | DeviceUnits | enum Constant | Unit24 |
| ROD\_UID\_25 | DeviceUnits | enum Constant | Unit25 |
| ROD\_UID\_26 | DeviceUnits | enum Constant | Unit26 |
| ROD\_UID\_27 | DeviceUnits | enum Constant | Unit27 |
| ROD\_UID\_28 | DeviceUnits | enum Constant | Unit28 |
| ROD\_UID\_29 | DeviceUnits | enum Constant | Unit29 |
| ROD\_UID\_30 | DeviceUnits | enum Constant | Unit30 |
| ROD\_UID\_31 | DeviceUnits | enum Constant | Unit31 |
| ROD\_UID\_32 | DeviceUnits | enum Constant | Unit32 |
|  |  |  |  |
| ROD\_ATTR\_BLINK | VideoAttributes | enum Constant | Blink |
| ROD\_ATTR\_BG\_BLACK | VideoAttributes | enum Constant | BackgroundBlack |
| ROD\_ATTR\_BG\_BLUE | VideoAttributes | enum Constant | BackgroundBlue |
| ROD\_ATTR\_BG\_GREEN | VideoAttributes | enum Constant | BackgroundGreen |
| ROD\_ATTR\_BG\_CYAN | VideoAttributes | enum Constant | BackgroundCyan |
| ROD\_ATTR\_BG\_RED | VideoAttributes | enum Constant | BackgroundRed |
| ROD\_ATTR\_BG\_MAGENTA | VideoAttributes | enum Constant | BackgroundMagenta |
| ROD\_ATTR\_BG\_BROWN | VideoAttributes | enum Constant | BackgroundBrown |
| ROD\_ATTR\_BG\_GRAY | VideoAttributes | enum Constant | BackgroundGray |
| ROD\_ATTR\_INTENSITY | VideoAttributes | enum Constant | ntensity |
| ROD\_ATTR\_FG\_BLACK | VideoAttributes | enum Constant | ForegroundBlack |
| ROD\_ATTR\_FG\_BLUE | VideoAttributes | enum Constant | ForegroundBlue |
| ROD\_ATTR\_FG\_GREEN | VideoAttributes | enum Constant | ForegroundGreen |
| ROD\_ATTR\_FG\_CYAN | VideoAttributes | enum Constant | ForegroundCyan |
| ROD\_ATTR\_FG\_RED | VideoAttributes | enum Constant | ForegroundRed |
| ROD\_ATTR\_FG\_MAGENTA | VideoAttributes | enum Constant | ForegroundMagenta |
| ROD\_ATTR\_FG\_BROWN | VideoAttributes | enum Constant | ForegroundBrown |
| ROD\_ATTR\_FG\_GRAY | VideoAttributes | enum Constant | ForegroundGray |

|  |  |  |  |
| --- | --- | --- | --- |
| UnifiedPOS Name | POS for .NET | | |
| ClassName | Parameter | |
| Type | Name |
| ROD\_BDR\_SINGLE | BorderType | enum Constant | Single |
| ROD\_BDR\_DOUBLE | BorderType | enum Constant | Double |
| ROD\_BDR\_SOLID | BorderType | enum Constant | Solid |
|  |  |  |  |
| ROD\_CLK\_START | ClockFunction | enum Constant | Start |
| ROD\_CLK\_PAUSE | ClockFunction | enum Constant | Pause |
| ROD\_CLK\_RESUME | ClockFunction | enum Constant | Resume |
| ROD\_CLK\_MOVE | ClockFunction | enum Constant | Move |
| ROD\_CLK\_STOP | ClockFunction | enum Constant | Stop |
|  |  |  |  |
| ROD\_CRS\_LINE | VideoCursorType | enum Constant | Line |
| ROD\_CRS\_LINE\_BLINK | VideoCursorType | enum Constant | LineBlink |
| ROD\_CRS\_BLOCK | VideoCursorType | enum Constant | Block |
| ROD\_CRS\_BLOCK\_BLINK | VideoCursorType | enum Constant | BlockBlink |
| ROD\_CRS\_OFF | VideoCursorType | enum Constant | Off |
|  |  |  |  |
| ROD\_CS\_UNICODE | PosCommon | System.Int32 | CharacterSetUnicode |
| ROD\_CS\_ASCII | PosCommon | System.Int32 | CharacterSetAscii |
| ROD\_CS\_WINDOWS | PosCommon | System.Int32 | No\_Equivalent\_Defined |
| ROD\_CS\_ANSI | PosCommon | System.Int32 | CharacterSetAnsi |
|  |  |  |  |
| ROD\_TD\_TRANSACTION | RemoteOderDisplayTransaction | enum Constant | Transaction |
| ROD\_TD\_NORMAL | RemoteOderDisplayTransaction | enum Constant | Normal |
|  |  |  |  |
| ROD\_UA\_SET | VideoAttributeCommand | enum Constant | Set |
| ROD\_UA\_INTENSITY\_ON | VideoAttributeCommand | enum Constant | IntensityOn |
| ROD\_UA\_INTENSITY\_OFF | VideoAttributeCommand | enum Constant | IntensityOff |
| ROD\_UA\_REVERSE\_ON | VideoAttributeCommand | enum Constant | ReverseOn |
| ROD\_UA\_REVERSE\_OFF | VideoAttributeCommand | enum Constant | ReverseOff |
| ROD\_UA\_BLINK\_ON | VideoAttributeCommand | enum Constant | BlinkOn |
| ROD\_UA\_BLINK\_OFF | VideoAttributeCommand | enum Constant | BlinkOff |
|  |  |  |  |
| ROD\_DE\_TOUCH\_DOWN | RemoteOrderDisplayEventTypes | enum Constant | TouchDown |
| ROD\_DE\_TOUCH\_MOVE | RemoteOrderDisplayEventTypes | enum Constant | TouchMove |
| ROD\_DE\_TOUCH\_UP | RemoteOrderDisplayEventTypes | enum Constant | TouchUp |
|  |  |  |  |
| EROD\_BADCLK | RemoteOrderDisplay | System.Int32 | ExtendedErrorBadClock |
| EROD\_NOCLOCKS | RemoteOrderDisplay | System.Int32 | ExtendedErrorNoClocks |
| EROD\_NOREGION | RemoteOrderDisplay | System.Int32 | ExtendedErrorNoRegion |
| EROD\_NOROOM | RemoteOrderDisplay | System.Int32 | ExtendedErrorNoRoom |
| EROD\_NOBUFFERS | RemoteOrderDisplay | System.Int32 | ExtendedErrorNoBuffers |
|  |  |  |  |
| SCAL\_WU\_GRAM | WaitUnit | enum Constant | Gram |
| SCAL\_WU\_KILOGRAM | WaitUnit | enum Constant | Kilogram |
| SCAL\_WU\_OUNCE | WaitUnit | enum Constant | Ounce |
| SCAL\_WU\_POUND | WaitUnit | enum Constant | Pound |
|  |  |  |  |
| ESCAL\_OVERWEIGHT | Scale | System.Int32 | ExtendedErrorOverWeight |
|  |  |  |  |
| SCAN\_SDT\_UNKNOWN | BarCodeSymbology | enum Constant | Unknown |
| SCAN\_SDT\_UPCA | BarCodeSymbology | enum Constant | Upca |
| SCAN\_SDT\_UPCE | BarCodeSymbology | enum Constant | Upce |
| SCAN\_SDT\_JAN8 | BarCodeSymbology | enum Constant | EanJan8 |
| SCAN\_SDT\_EAN8 | BarCodeSymbology | enum Constant | No\_Equivalent\_Defined |
| SCAN\_SDT\_JAN13 | BarCodeSymbology | enum Constant | EanJan13 |
| SCAN\_SDT\_EAN13 | BarCodeSymbology | enum Constant | No\_Equivalent\_Defined |
| SCAN\_SDT\_TF | BarCodeSymbology | enum Constant | TF |
| SCAN\_SDT\_ITF | BarCodeSymbology | enum Constant | Itf |
| SCAN\_SDT\_Codabar | BarCodeSymbology | enum Constant | Codabar |
| SCAN\_SDT\_Code39 | BarCodeSymbology | enum Constant | Code39 |
| SCAN\_SDT\_Code93 | BarCodeSymbology | enum Constant | Code93 |
| SCAN\_SDT\_Code128 | BarCodeSymbology | enum Constant | Code128 |
| SCAN\_SDT\_UPCA\_S | BarCodeSymbology | enum Constant | Upcas |
| SCAN\_SDT\_UPCE\_S | BarCodeSymbology | enum Constant | Upces |
| SCAN\_SDT\_UPCD1 | BarCodeSymbology | enum Constant | Upcd1 |
| SCAN\_SDT\_UPCD2 | BarCodeSymbology | enum Constant | Upcd2 |
| SCAN\_SDT\_UPCD3 | BarCodeSymbology | enum Constant | Upcd3 |
| SCAN\_SDT\_UPCD4 | BarCodeSymbology | enum Constant | Upcd4 |
| SCAN\_SDT\_UPCD5 | BarCodeSymbology | enum Constant | Upcd5 |
| SCAN\_SDT\_EAN8\_S | BarCodeSymbology | enum Constant | Ean8S |
| SCAN\_SDT\_EAN13\_S | BarCodeSymbology | enum Constant | Ean13S |
| SCAN\_SDT\_EAN128 | BarCodeSymbology | enum Constant | Ean128 |

|  |  |  |  |
| --- | --- | --- | --- |
| UnifiedPOS Name | POS for .NET | | |
| ClassName | Parameter | |
| Type | Name |
| SCAN\_SDT\_OCRA | BarCodeSymbology | enum Constant | Ocra |
| SCAN\_SDT\_OCRB | BarCodeSymbology | enum Constant | Ocrb |
| SCAN\_SDT\_RSS14 | BarCodeSymbology | enum Constant | Rss14 – Deprecated v1.12 |
| SCAN\_SDT\_RSS\_EXPANDED | BarCodeSymbology | enum Constant | RssExpanded – Deprecated b1.12 |
| SCAN\_SDT\_GS1DATABAR | BarCodeSymbology | enum Constant | GS1DataBar Omnidirectional |
| SCAN\_SDT\_GS1DATABAR\_E | BarCodeSymbology | enum Constant | GS1 DataBar Expanded |
| SCAN\_SDT\_CCA | BarCodeSymbology | enum Constant | Cca |
| SCAN\_SDT\_CCB | BarCodeSymbology | enum Constant | Ccb |
| SCAN\_SDT\_CCC | BarCodeSymbology | enum Constant | Ccc |
| SCAN\_SDT\_PDF417 | BarCodeSymbology | enum Constant | Pdf417 |
| SCAN\_SDT\_MAXICODE | BarCodeSymbology | enum Constant | Maxicode |
| SCAN\_SDT\_OTHER | BarCodeSymbology | enum Constant | Other |
|  |  |  |  |
| SC\_CMODE\_TRANS | SmartCardInterfaceModes | enum Constant | Transaction |
| SC\_CMODE\_BLOCK | SmartCardInterfaceModes | enum Constant | Block |
| SC\_CMODE\_APDU | SmartCardInterfaceModes | enum Constant | Apdu |
| SC\_CMODE\_XML | SmartCardInterfaceModes | enum Constant | Xml |
| SC\_CMODE\_ISO | SmartCardIsoEmvModes | enum Constant | Iso |
| SC\_CMODE\_EMV | SmartCardIsoEmvModes | enum Constant | Emv |
|  |  |  |  |
| SC\_CTRANS\_PROTOCOL\_T0 | SmartCardTransactionProtocols | enum Constant | T0 |
| SC\_CTRANS\_PROTOCOL\_T1 | SmartCardTransactionProtocols | enum Constant | T1 |
|  |  |  |  |
| SC\_MODE\_TRANS | SmartCardInterfaceModes | enum Constant | Transaction |
| SC\_MODE\_BLOCK | SmartCardInterfaceModes | enum Constant | Block |
| SC\_MODE\_APDU | SmartCardInterfaceModes | enum Constant | Apdu |
| SC\_MODE\_XML | SmartCardInterfaceModes | enum Constant | Xml |
| SC\_MODE\_ISO | SmartCardIsoEmvModes | enum Constant | Iso |
| SC\_MODE\_EMV | SmartCardIsoEmvModes | enum Constant | Emv |
|  |  |  |  |
| SC\_TRANS\_PROTOCOL\_T0 | SmartCardTransactionProtocols | enum Constant | T0 |
| SC\_TRANS\_PROTOCOL\_T1 | SmartCardTransactionProtocols | enum Constant | T1 |
|  |  |  |  |
| SC\_READ\_DATA | SmartCardReadAction | enum Constant | ReadData |
| SC\_READ\_PROGRAM | SmartCardReadAction | enum Constant | ReadProgram |
| SC\_EXECUTE\_AND\_READ\_DATA | SmartCardReadAction | enum Constant | ExecuteAndReadData |
| SC\_XML\_READ\_BLOCK\_DATA | SmartCardReadAction | enum Constant | XmlReadBlockData |
| SC\_STORE\_DATA | SmartCardWriteAction | enum Constant | StoreData |
| SC\_STORE\_PROGRAM | SmartCardWriteAction | enum Constant | StoreProgram |
| SC\_EXECUTE\_DATA | SmartCardWriteAction | enum Constant | ExecuteData |
| SC\_XML\_BLOCK\_DATA | SmartCardWriteAction | enum Constant | XmlBlockData |
| SC\_SECURITY\_FUSE | SmartCardWriteAction | enum Constant | SecurityFuse |
| SC\_RESET | SmartCardWriteAction | enum Constant | Reset |
|  |  |  |  |
| SC\_SUE\_NO\_CARD | No\_Equivalent\_Defined | No\_Equivalent\_Defined |  |
| SC\_SUE\_CARD\_PRESENT | No\_Equivalent\_Defined | No\_Equivalent\_Defined |  |
|  |  |  |  |
| ESC\_READ | SmartCardRW | System.Int32 | ExtendedErrorRead |
| ESC\_WRITE | SmartCardRW | System.Int32 | ExtendedErrorWrite |
| ESC\_TORN | SmartCardRW | System.Int32 | ExtendedErrorTorn |
| ESC\_NO\_CARD | SmartCardRW | System.Int32 | ExtendedErrorNoCard |
|  |  |  |  |
| ETOT\_NOROOM | HardTotals | System.Int32 | ExtendedErrorNoRoom |
| ETOT\_VALIDATION | HardTotals | System.Int32 | ExtendedErrorValidation |
|  |  |  |  |
| STAT\_BarcodePrintedCount | PosPrinter | System.String | StatisticBarcodePrintedCount |
| STAT\_BumpCount | BumpBar | System.String | StatisticBumpCount |
| STAT\_CommunicationErrorCount | PosCommon | System.String | StatisticCommunicationErrorCount |
| No\_Equivalent\_Defined | PosCommon | System.String | StatisticDeviceCategory |
| STAT\_DrawerFailedOpenCount | CashDrawer | System.String | StatisticDrawerFailedOpenCount |
| STAT\_DrawerGoodOpenCount | CashDrawer | System.String | StatisticDrawerGoodOpenCount |
| STAT\_FailedDataParseCount | Micr | System.String | StatisticFailedDataParseCount |
| STAT\_FailedPaperCutCount | PosPrinter | System.String | StatisticFailedPaperCutCount |
| STAT\_FailedPrintSideChangeCount | PosPrinter | System.String | StatisticFailedPrintSideChangeCount |
| STAT\_FailedReadCount | Micr | System.String | StatisticFailedReadCount |
| No\_Equivalent\_Defined | Msr | System.String | StatisticFailedReadCount |
| STAT\_FailedSignatureReadCount | SignatureCapture | System.String | StatisticFailedSignatureReadCount |
| No\_Equivalent\_Defined | PosCommon | System.String | StatisticFirmwareRevision |
| STAT\_FormInsertionCount | PosPrinter | System.String | StatisticFormInsertionCount |
| STAT\_GoodReadCount | Micr | System.String | StatisticGoodReadCount |
| No\_Equivalent\_Defined | Msr | System.String | StatisticGoodReadCount |
| STAT\_GoodScanCount | Scanner | System.String | StatisticGoodScanCount |
| STAT\_GoodSignatureReadCount | SignatureCapture | System.String | StatisticGoodSignatureReadCount |
| STAT\_GoodWeightReadCount | Scale | System.String | StatisticGoodWeightReadCount |
| STAT\_HomeErrorCount | PosPrinter | System.String | StatisticHomeErrorCount |
| STAT\_HoursPoweredCount | PosCommon | System.String | StatisticHoursPoweredCount |
| UnifiedPOS Name | POS for .NET | | |
| ClassName | Parameter | |
| Type | Name |
| No\_Equivalent\_Defined | PosCommon | System.String | StatisticInstallationDate |
| No\_Equivalent\_Defined | PosCommon | System.String | StatisticInterface |
| STAT\_InvalidPINEntryCount | PinPad | System.String | StatisticInvalidPINEntryCount |
| STAT\_JournalCharacterPrintedCount | PosPrinter | System.String | StatisticJournalCharacterPrintedCount |
| No\_Equivalent\_Defined | PosPrinter | System.String | StatisticJournalCoverOpenCount |
| STAT\_JournalLinePrintedCount | PosPrinter | System.String | StatisticJournalLinePrintedCount |
| STAT\_KeyPressedCount | PosKeyBoard | System.String | StatisticKeyPressedCount |
| STAT\_LockPositionChangeCount | Keylock | System.String | StatisticLockPositionChangeCount |
| No\_Equivalent\_Defined | PosCommon | System.String | StatisticManufactureDate |
| No\_Equivalent\_Defined | PosCommon | System.String | StatisticManufacturerName |
| STAT\_MaximumTempReachedCount | PosPrinter | System.String | StatisticMaximumTempReachedCount |
| No\_Equivalent\_Defined | PosCommon | System.String | StatisticMechanicalRevision |
| No\_Equivalent\_Defined | PosCommon | System.String | StatisticModelName |
| STAT\_MotionEventCount | MotionSensor | System.String | StatisticMotionEventCount |
| STAT\_NVRAMWriteCount | PosPrinter | System.String | StatisticNVRAMWriteCount |
| STAT\_OnlineTransitionCount | LineDisplay | System.String | StatisticOnlineTransitionCount |
| STAT\_PaperCutCount | PosPrinter | System.String | StatisticPaperCutCount |
| STAT\_PrinterFaultCount | PosPrinter | System.String | StatisticPrinterFaultCount |
| STAT\_PrintSideChangeCount | PosPrinter | System.String | StatisticPrintSideChangeCount |
| STAT\_ReceiptCharacterPrintedCount | PosPrinter | System.String | StatisticReceiptCharacterPrintedCount |
| STAT\_ReceiptCoverOpenCount | PosPrinter | System.String | StatisticReceiptCoverOpenCount |
| STAT\_ReceiptLineFeedCount | PosPrinter | System.String | StatisticReceiptLineFeedCount |
| STAT\_ReceiptLinePrintedCount | PosPrinter | System.String | StatisticReceiptLinePrintedCount |
| No\_Equivalent\_Defined | PosCommon | System.String | StatisticSerialNumber |
| STAT\_SlipCharacterPrintedCount | PosPrinter | System.String | StatisticSlipCharacterPrintedCount |
| STAT\_SlipCoverOpenCount | PosPrinter | System.String | StatisticSlipCoverOpenCount |
| STAT\_SlipLineFeedCount | PosPrinter | System.String | StatisticSlipLineFeedCount |
| STAT\_SlipLinePrintedCount | PosPrinter | System.String | StatisticSlipLinePrintedCount |
| STAT\_StampFiredCount | PosPrinter | System.String | StatisticStampFiredCount |
| STAT\_ToneSoundedCount | ToneIndicator | System.String | StatisticToneSoundedCount |
| No\_Equivalent\_Defined | PosCommon | System.String | StatisticUnifiedPOSVersion |
| STAT\_UnreadableCardCount | Msr | System.String | StatisticUnreadableCardCount |
| STAT\_ValidPINEntryCount | PinPad | System.String | StatisticValidPINEntryCount |

Structures

POS for .NET defines structure types to aggregate data values that are returned by method calls. This is required since parameters in POS for .NET are **In** only. On the other hand, structure types are used in POS for .NET to provide a more type-safe handling for aggregated data. Structural strings containing several data values that are returned by a UnifiedPOS property or method are broken into members of a new defined structure type.

Structures are like classes. However, structures have value semantics and they do not require heap allocation. The language concept of structures is described in the MSDN Library documentation.

The following structures are defined in POS for .NET.

#### CashCount Structure

The structure ***CashCount*** contains the dispensing cash units and counts.

##### Structure Properties

|  |  |
| --- | --- |
| Name | Description |
| Count | Holds the number bills or coins. |
| NominalValue | Holds the face value. |
| Type | Defines whether the currency is bills or coins. |

##### Used by

* **CashChanger.DepositCounts** Property as item type of the returned array, the POS for .NET method has the following signature:

public abstract CashCount[] DepositCounts

* **CashChanger.DispenseCash** Method parameter array item type for the parameter *CashCounts*, the POS for .NET method has the following signature:

public abstract void DispenseCash( CashCount[] cashCounts )

#### CashCounts Structure

The structure ***CashCounts*** aggregates an array of items of type **CashCount** whether a cash discrepancy is given or not.

##### Structure Properties

|  |  |
| --- | --- |
| Name | Description |
| Counts | Holds the CashCount data. |
| Discrepancy | If TRUE, there is some cash that could not be included in a CashCount; otherwise FALSE. |

##### Used by

* **CashChanger.ReadCashCounts** Method as return value type, the POS for .NET method has the following signature:

public abstract CashCounts ReadCashCounts()

#### CashUnits Structure

Holds the cash units supported in the **CashChanger**. The cash units are stored in two separate String arrays for bills and coins.

##### Structure Properties

|  |  |
| --- | --- |
| Name | Description |
| Bills | Holds the number of each type of bill. |
| Coins | Holds the number of each type of coin. |

##### Used by

* **CashChanger.DepositCashList** Property as return value type, the POS for .NET method has the following signature:

public abstract CashUnits DepositCashList

* **CashChanger.CurrenyCashList** Property as return value type, the POS for .NET method has the following signature:

public abstract CashUnits CurrencyCashList

* **CashChanger.ExitCashList** Property as return value type, the POS for .NET method has the following signature:

public abstract CashUnits ExitCashList

#### DirectIOData Structure

The structure ***DirectIOData*** aggregates values that are returned by the **DirectIO** method.

##### Structure Properties

|  |  |
| --- | --- |
| Name | Description |
| Data | Specific value varies by Command and Service Object. |
| Object | Specific object varies by Command and Service Object. |

##### Used by

* **PosCommon.DirectIO** Method as return value type, the POS for .NET method has the following signature:

public abstract DirectIOData DirectIO( int command, int data, object obj )

#### FiscalDataItem Structure

The structure ***FiscalDataItem*** aggregates values that are returned by the **GetData** method of the **FiscalPrinter** category.

##### Structure Properties

|  |  |
| --- | --- |
| Name | Description |
| Data | Character string describing data. |
| ItemOption | Optional additional parameter. Consult the Service Object vendor's documentation for more information about how to use this argument. |

##### Used by

* **FiscalPrinter.GetData** Method as return value type, the POS for .NET method has the following signature:

public abstract FiscalDataItem GetData(FiscalData dataItem, int itemOption)

#### TotalsFileInfo Structure

The structure ***TotalsFileInfo*** aggregates file information for the **HardTotals** device category.

##### Structure Properties

|  |  |
| --- | --- |
| Name | Description |
| Handle | Handle to the totals file. |
| Size | Totals file size. |

##### Used by

* **Totals.Find** Method as return value type, the POS for .NET method has the following signature:

public abstract TotalsFileInfo Find( string fileName )

#### VatInfo Structure

The structure ***VatInfo*** aggregates VAT information used in the **FiscalPrinter** category.

##### Structure Properties

|  |  |
| --- | --- |
| Name | Description |
| Amount | Indicates the VAT amount. |
| Id | VAT identifier. |

##### Used by

* **FiscalPrinter.PrintRecPackageAdjustVoid** Method as array item type of the parameter *vatAdjustments*, the POS for .NET method has the following signature:

public abstract void PrintRecPackageAdjustVoid( FiscalAdjustmentType adjustmentType, VatInfo[] vatAdjustments )

* **FiscalPrinter.PrintRecPackageAdjustment** Method array item type of the parameter *vatAdjustments*, the POS for .NET method has the following signature:

public abstract void PrintRecPackageAdjustment( FiscalAdjustmentType adjustmentType, string description, VatInfo[] vatAdjustments )

#### VideoMode Structure

The structure ***VideoMode*** holds the video modes supported for the video unit used by the **RemoteOrderDisplay** device category.

##### Structure Properties

|  |  |
| --- | --- |
| Name | Description |
| Colors | The number of colors. |
| Columns | The number of columns. |
| IsColor | TRUE if video is color; otherwise, FALSE |
| Rows | The number of rows. |

##### Used by

* **RemoteOrderDisplay.VideoModesList** Property as item type of the returned array, the POS for .NET method has the following signature:

public abstract VideoMode[] VideoModesList

### Complete Class Libraries Provided

#### Interface Classes

* Interface libraries provide no code functionality. They represent the interface to the device class only. There are Interface classes for each of the device classes defined within UnifiedPOS.
* The interfaces meet or provide extensions to the UnifiedPOS specification standards.
* The interface classes define all the constants needed for management of device statistics, status reporting via events, and standard error conditions.
* The interface classes define all the enumerations needed for all device classes.

#### Basic Classes

* Basic classes inherit from the Interface classes and implement the common functionality across device classes. For example, the Basic classes implement the **Open()**, **Claim()**, and **Release()** methods. There are Basic classes for each of the device classes defined within UnifiedPOS.
* The Basic classes not only manage all common properties and methods, they manage event delivery to the application, retrieval and storage of device statistics, manage error handling for all classes of errors, and provide functionality for notifying the Service Object of hardware state change conditions.

#### Base Classes

* Base classes inherit from Basic classes and implement device class specific functionality across device classes. The Device Service Object provider is left to implement only the hardware-specific functionality.
* Base classes build on the basic class functionality by providing implementations for all event types (as well as managing event delivery), increment and manage all device statistics, manage validation of property and parameter values (and deliver errors, as needed, to the application), update all device-specific properties according to specification guidelines as part of delivering data events to the application, plus provide a flexible structure of protected methods and helper classes that the application can use if it chooses to provide its own hardware-specific functionality.

### Return Values

Many POS for .NET API calls return a value. For example, the common method

string CheckHealth (HealthCheckLevel *level*);

returns a string describing the health level. Parameters in POS for .NET are **In** only.

### Returning Properties

Often, an application method call will result in the change of a property value or the method will return some status value as defined within the UnifiedPOS specification.

For example, assume the following case:

An ISV calls a method that may change the value of a specific property. Further processing is dependent upon the new value of the property. In the OPOS implementation of UnifiedPOS, the ISV would first make a method call and then call another method that would return the value of the property.

MethodThatChangesAProperty()

Dim MyProperty as Property

GetPropertyValue(MyProperty)

//GetPropertyValue has a

// byref parameter

Select MyProperty

case ….

In POS for .NET, the ISV would call the method and test the returned value as follows (Visual Basic .NET):

Select MethodThatChangesAProperty()

Case ….

### Returning Lists

Often, a method will return a list of values. In OPOS, methods that return lists do so by returning strings that are comma-delimited (regardless of the data type of the list item). The application must construct the string and do any necessary conversion of the data items to a string, adding commas as delimiters.

The application will have to parse the string and cast the data items into the type associated by the list.

Example:

CHAR nChar = “,”;

int x;

int y = 0;

CHAR\* pMyElements[];

CHAR\* psCurrent;

for(x=0;x<len(sReturn);x++)

{

if(sReturn[x] == nChar)

{

pMyElements[y] = psCurrent;

y++;

}

psCurrent += sReturn[x];

}

//assumes all return types should be strings if not,

//cast to appropriate data type is required

In POS for .NET, arrays are native data types. There is no need to cast the data elements to a coerced type. Further, arrays provide their own iterate functions to allow easy access to all items in the list.

//use each item as needed

SomeMethod(ReturnedArray[0]);

SomeOtherMethod(ReturnedArray[1]);

NOTE:

From the SO, the following code demonstrates returning a clone – necessary to preserve data safety.

return SomeArrary.Clone();

The reasons to return arrays instead of compound strings are as follows:

* Arrays are native data types in .NET and they can be enumerated with a FOR EACH statement.
* Building and parsing delimited strings introduces more code that must be maintained and increases the chance of introducing bugs.
* Clarity of intent of the code is clearer when arrays are used.

Example:

To further illustrate the differences between UnifiedPOS, OPOS, and POS for .NET, refer to the property **PosPrinter**.**CharacterSetList**. This property has the following signature in UnifiedPOS:

CharacterSetList: string { read-only, access after open }

The property in UnifiedPOS returns a string with a comma separated list of code page numbers. The application program must parse the string to extract the code page numbers and must convert them to integer values if needed. In POS for .NET, the property **PosPrinter**.**CharacterSetList** has the following method signature:

public abstract int[] CharacterSetList

This returns the list of code page numbers as an integer array. There is no need for parsing a string and converting code page numbers to integer values. This approach is more type safe and easier to handle for application programmers.

Key Parameter Differences

POS for .NET makes use of enumerations versus OPOS use of constants.

POS for .NET makes use of array data typing versus OPOS use of compound strings.

POS for .NET makes use of native integer types.

POS for .NET makes use of “right-sizing” variables (using variables that match the type of data they represent) rather than OPOS use of data types for values that require more bytes than would ever be necessary to contain the proper meaning and expected range.

POS for .NET divides a UnifiedPOS method into multiple POS for .NET methods if it contains a parameter that can contain only 2 or 3 values. E.g., the **FiscalPrinter** method **printReport** has the following signature under UnifiedPOS:

printReport(reportType: *int32*, startNum: *string*, endNum: *string*): void

The parameter **reportType** can have only one of the following values - FPTR\_RT\_ORDINAL or FPTR\_RT\_DATE. For FPTR\_RT\_DATE the two following parameters must be interpreted as date strings otherwise both values must be used as integer values.

In POS for .NET the **reportType** parameter is omitted. Instead two new methods have been introduced defining **printReport**() with different signatures. These are more type safe.

void PrintReport(DateTime *startDate*, DateTime *endDate*)

**void PrintReport**(**int** *startNumber*, **int** *endNumber*)

The following table lists the method/parameter differences in POS for .NET compared to the corresponding UnifiedPOS method/parameters. Methods differing only by the usage of an Enumeration type are not listed.

|  |  |
| --- | --- |
| UnifiedPOS Method | POS for .NET |
| **CashChanger** |  |
| **dispenseCash(cashCounts: string): void** | **void DispenseCash(CashCount[] cashCounts)** |
| **FiscalPrinter** |  |
| **getData(dataItem: int32, inout optArgs: int32, inout data: string): void** | **FiscalDataItem GetData(FiscalData dataItem, int**  **itemOption)** |
| **printPeriodicTotalsReport(date1: string, date2: string): void** | **void PrintPeriodicTotalsReport(DateTime startingDate, DateTime endingDate)** |
| **printRecItem(description: string, price: currency, quantity: int32, vatInfo: int32, unitPrice: currency, unitName: string): void** | **void PrintRecItem(string description, decimal price, decimal quantity, int vatId, decimal unitPrice, string unitName)** |
| **printRecPackageAdjustment(adjustmentType: int32, description: string, vatAdjustment: string): void** | **void PrintRecPackageAdjustment**( **FiscalAdjustmentType** *adjustmentType*, **string** *description*, **VatInfo[]** *vatAdjustments*) |
| **printRecPackageAdjustVoid**(**adjustmentType:** *int32*, **vatAdjustment**: *string*): **void** | **void PrintRecPackageAdjustVoid**( **FiscalAdjustmentType** *adjustmentType*, **VatInfo[]** *vatAdjustments*) |
| **printReport**(**reportType**: *int32*, **startNum**: *string*, **endNum**: *string*): **void** | **void** **PrintReport**(**DateTime** *startDate*, **DateTime** *endDate*) |
| **printReport**(**reportType**: *int32*, **startNum**: *string*, **endNum**: *string*): **void** | **void** **PrintReport**(**int** *startNumber*, **int** *endNumber*) |
| **printReport**(**reportType**: *int32*, **startNum**: *string*, **endNum**: *string*): **void** | **void** **PrintReport**(**int** *startNumber*) |
| **setDate**(**date**: *string*): **void** | **void** **SetDate**(**DateTime** *newDate*) |
| **setVatValue**(**vatID**: *int32*, **vatValue**: *string*): **void** | **void** **SetVatValue**(**int** *vatId*, **decimal** *vatRate*) |

## Key Property Signature Differences

There are several properties which have different POS for .NET signatures compared to UnifiedPOS. They use arrays or structures instead of comma separated lists inside strings. The following table shows these properties.

|  |  |
| --- | --- |
| **UnifiedPOS Property** | **POS for .NET Signature** |
| **CashChanger** |  |
| **CurrencyCodeList** | public abstract string[] CurrencyCodeList |
| **CurrencyCashList** | public abstract CashUnits CurrencyCashList |
| **DepositCodeList** | public abstract string[] DepositCodeList |
| **DepositCounts** | public abstract CashCount[] DepositCounts |
| **ExitCashList** | public abstract CashUnits ExitCashList |
| **CheckScanner** |  |
| **QualityList** | public abstract int[] QualityList |
| **FiscalPrinter** |  |
| **PredefinedPaymentLines** | public abstract string[] PredefinedPaymentLines |
| **POSPrinter** |  |
| **CharacterSetList** | public abstract int[] CharacterSetList |
| **FontTypefaceList** | public abstract string[] FontTypefaceList |
| **RecBarCodeRotationList** | public abstract Rotation[] RecBarCodeRotationList |
| **RecBitmapRotationList** | public abstract Rotation[] RecBitmapRotationList |
| **SlpBarCodeRotationList** | public abstract Rotation[] SlpBarCodeRotationList |
| **SlpBitmapRotationList** | public abstract Rotation[] SlpBitmapRotationList |
| **RemoteOrderDisplay** |  |
| **VideoModesList** | public abstract VideoMode[] VideoModesList |

## More Information

Samples are available in the POS for .NET Software Development Kit (SDK) which is available for download at https://aka.ms/p4dn-dl.

PosExplorer API

**PosExplorer** is used by applications to acquire a list of installed POS devices, open—or create instances of—service objects for those devices and receive Plug-n-Play events when the devices are connected or disconnected from the system.

## **PosExplorer Properties**

PosRegistryKey Property

Syntax public static string PosRegistryKey {read-only}

Remarks Holds the POS for .NET configuration root registry key relative to HKEY\_LOCAL\_MACHINE.

StatisticsFile Property

Syntax public static string StatisticsFile {read-only}

**Remarks** Holds the path to the file in which device statistics is persisted.

SynchronizingObject Property

Syntax public ISynchronizeInvoke SynchronizingObject {read-write}

**Remarks** Sets or holds the **ISynchronizeInvoke** object.

## PosExplorer Methods

CreateInstance Method

Syntax public PosDevice CreateInstance(DeviceInfo *device*)

**Remarks** Instantiates the device based on the information supplied by the property values of the **DeviceInfo** object.

Parameter Description

*device* An object that describes the device you want to create an instance of, and which is an instance of the DeviceInfo class. DeviceInfo contains properties such as Compatibility, Description, HardwareID, and so on, for the device.

GetDevice Method (string)

Syntax public DeviceInfo GetDevice(string *type*)

**Remarks** Retrieves a device of the specified type.

Parameter Description

*type* A string that contains one of the UnifiedPOS device types, as defined by the **DeviceType** helper class.

There must be only one device of that type currently in the system, or if there is more than one, one must have been configured as the default device. If there is more than one device of the specified type and no device has been configured as the default device, a **PosLibraryException** will be thrown.

This signature of **GetDevice** represents the simplest case for retrieving and instantiating a device in the POS for .NET system. To retrieve one device and instantiate its service object, the application must only:

* Create an instance of **PosExplorer**;
* Call **GetDevice** using the above method signature; and
* Call CreateInstance.

POS for .NET initializes the device of the type specified or, if there is more than one device of that type, the pre-configured default device for that type.

GetDevice Method (string, string)

Syntax public DeviceInfo GetDevice(string *type*, string *logicalName*)

**Remarks** Retrieves a device of the specified type and name (or alias).

Parameter Description

*type* A string that contains one of the UnifiedPOS device types, as defined by the **DeviceType** helper class.

*logicalName* The logical name or alias of the device.

GetDevices Method

Syntax public DeviceCollection GetDevices()

**Remarks** Retrieves all POS devices currently installed in the system.

GetDevices Method (DeviceCompatibilities)

Syntax public DeviceCollection GetDevices(DeviceCompatibilities *compatibility*)

**Remarks** Retrieves all POS devices currently installed in the system, based on a compatibility level.

Parameter Description

*compatibility* **DeviceCompatibilities** enumeration that indicates compatibility level.

GetDevices Method (string)

Syntax public DeviceCollection GetDevices(string *type*)

**Remarks** Retrieves all POS devices of the specified type.

Parameter Description

*type* A string that contains one of the UnifiedPOS device types, as defined by the **DeviceType** helper class.

GetDevices Method (string, DeviceCompatibilities)

Syntax public DeviceCollection GetDevices(string *type*, DeviceCompatibilities *compatibility*)

**Remarks** Retrieves all POS devices of the specified type based on a compatibility level.

Parameter Description

*type* A string that contains one of the UnifiedPOS device types, as defined by the **DeviceType** helper class.

*compatibility* **DeviceCompatibilities** enumeration that indicates compatibility level.

Refresh Method

Syntax public void Refresh()

**Remarks** Re-enumerates the list of attached POS devices and rebuilds the internal data structures.

## PosExplorer Events

DeviceAddedEvent Event

Syntax public event DeviceChangedEventHandler DeviceAddedEvent;

**Remarks** Notifies the application when a POS device has been added to the system.

**DeviceAddedEvent** only notifies for POS devices for which there is a service object installed.

The event handler receives an argument of type **DeviceChangedEventArgs** which contains a **DeviceInfo** object for the added device.

DeviceRemovedEvent Event

Syntax public event DeviceChangedEventHandler DeviceRemovedEvent;

**Remarks** Notifies the application when a POS device has been removed from the system.

**DeviceRemovedEvent** only notifies for POS devices for which there is a service object installed.

The event handler receives an argument of type **DeviceChangedEventArgs** which contains a **DeviceInfo** object for the removed device.

## Global Configuration

**PosExplorer** reads the global configuration file (config.xml), which enables application developers to specify aliases for Plug-n-Play and non Plug-n-Play devices, and to define physical devices for non Plug-n-Play Service Objects.

The global configuration file also enables application developers to define more than one physical device associated with a non Plug-n-Play Service Object and to assign aliases and device paths (such as COM ports) to them. This enables Application Developers to target non Plug-n-Play Service Objects to specific physical devices.

Service Object Registry

In OPOS, configuration information for Service Objects is stored in the registry. In POS for .NET, configuration information is stored in Config.xml. POS for .NET enables seamless access to registry information for COM Service Objects through PosExplorer; the Common Control Library does the work of gathering registry configuration information.

## Consuming Service Objects

### OPOS

Control Objects represent the application interface to its matching Service Object. The UnifiedPOS standard does not provide any code for Control Objects. However, it does suggest that the OPOS Control objects located at [http://](http://www.monroecs.com/oposccos.htm)www.monroecs.com/oposccos.htm be used or at the very least tested against. In addition, the site holds an ActiveX® Control that is an aggregation of all device classes. This is commonly referred to as the Common Controls Objects.

Under OPOS it is customary practice for IHVs, ISVs, and OEMs to create their own versions of Control Objects and to not use or test the referenced Common Control Objects. This has led to compatibility issues between hardware, services, and application code.

The OPOS implementation consists of the following steps:

* Instantiate an instance of the Control Object
* Call the Control Objects:
  + Open to load the Service Object by name
  + Claim
  + Enable

Note that on a device-by-device basis, there may be properties that must be read or set before interacting with the device for device-specific functionality.

### POS for .NET

To instantiate a Service Object in POS for .NET, do the following:

* Instantiate the PosExplorer object.
* Use the PosExplorer.GetDevice or GetDevices method to obtain a list of one or more DeviceInfo objects that represent devices attached to the machine.
* Call PosExplorer.CreateInstance, passing in the DeviceInfo for the device you want to load.
* Call methods/properties on the Service Object returned by the previous step.

The supplied **PosExplorer** tool is a helper class that acts as a Service Object Factory. The developer will instantiate:

Sample POSExplorer.GetDevice(…);

This approach provides the following benefits:

* Achieves infrastructure required to support feature set (see POS for .NET features).
* Simplifies an application: One section of code can be used to dynamically instantiate a Service Object.
* For most cases it eliminates the need for detailed knowledge of the specific brand of hardware peripheral.
* An application can easily get a list of available POS peripherals attached to the device (Available for Plug-n-Play).
* For an application there is no difference between .NET SOs and OPOS SOs.

## **Writing Service Objects**

### POS for .NET

There are three different approaches available:

* Derive the Service Object from the Interface class
* Derive the Service Object from the Basic class
* Derive the Service Object from the Base class

There are various levels of work required for the Service Object writer for each approach. For example, deriving from the Interface class requires the most amount of code to be implemented by the service application yet gives it the most control over the operation of the Service Object. By deriving from the Basic class, the service application only must implement the core functionality of the device. The Basic class already provides the common functionality. Deriving from the Base class leaves the service application with only having to implement the specific hardware functionality; the basic functionality of the device class has already been provided.

## Status, State Model, and Exceptions

The status, error code, and state models are built around several common enumerations, events, and a property, described below:

#### StatusUpdateEvent

An event fired when some class-specific state or status variable has changed.

#### ControlState

An enumeration containing the current state. Possible values are:

* Closed
* Idle
* Busy
* Error

#### Exceptions

Every POS for .NET method invocation and property access may throw a **PosControlException** upon failure, except for accesses to the properties **DeviceControlVersion**, **DeviceControlDescription**, and **State**. No other types of exceptions will be thrown.

PosControlException is defined in the namespace Microsoft.PointOfService and extends System.Exception.

|  |  |
| --- | --- |
| **Name** | **Description** |
| *E*[*rrorCode*](http://msdn.microsoft.com/library/en-us/ccl/html/P_Microsoft_PointOfService_PosControlException_ErrorCode.asp) | *ErrorCode* causing the error exception. See the list of Error Codes on page 0-20. |
| *E*[*rrorCodeExtended*](http://msdn.microsoft.com/library/en-us/ccl/html/P_Microsoft_PointOfService_PosControlException_ErrorCodeExtended.asp) | Extended Error Code causing the error exception. This may contain a Service-specific value. |

##### Public Properties

The constructor variations are defined as follows:

PosControlException (string message, ErrorCode errorCode)

PosControlException (string message, ErrorCode errorCode, Exception innerException)

PosControlException (string message, ErrorCode errorCode, int errorCodeExtended)

PosControlException (string message, ErrorCode errorCode, int errorCodeExtended, Exception *innerException*)

The parameters are defined as follows:

Parameter Description

*errorCode* The POS for .NET error code. Access is through the **ErrorCode** getter method.

*errorCodeExtended* May contain an extended error code. If not provided by the selected constructor, then is set to zero. Access is through the **ErrorCodeExtended** getter method.

*message* A text description of the error. If not provided by the selected constructor, then one is formed from the *errorCode* and *errorCodeExtended* parameters. Access is through the superclass’ getter method **Message** or method **ToString**.

*innerException* Original exception. If the POS for .NET Service caught a non-POS for .NET exception, then an appropriate *errorCode* is selected and the original exception is referenced by this parameter. Otherwise, it is set to null. Access is through the inherited getter method **InnerException**.

**Device Sharing Model**

The POS for .NET device sharing model supports devices that are to be used exclusively by one application at a time, as well as devices that may be partially or fully shared by multiple applications. All POS for .NET service objects may be opened by more than one application at a given time. Some or many of the activities that an application can perform with the service object, however, may be restricted to an application that claims access to the device.

#### Exclusive-Use Devices

The most common device type is called an “exclusive-use device”. An example is the POSPrinter. Due to physical or operational characteristics, this device can only be used by one application at a time. The application must call the **Claim** method to gain exclusive access to the device before most methods, properties, or events are legal. Until the device is claimed, calling methods or setting properties cause an *Illegal* error, and events are not fired to the application.

Should two closely cooperating applications want to treat an exclusive-use device in a shared manner, then one application may claim the device for a short sequence of operations, then release it so that the other application may use it.

When the **Claim** method is called again, settable device characteristics are restored to their condition at **Release**. Examples of restored characteristics are the LineDisplay's brightness, the MSR's tracks to read, and the POSPrinter's characters per line. **State** characteristics are not restored, such as the POSPrinter's sensor properties. Instead, these are updated to their current values.

#### Sharable Devices

Some devices are “sharable devices”. An example is the Keylock. A sharable device allows multiple applications to call its methods and access its properties. Also, it may fire its events to all applications that have opened it. A sharable device may still limit access to some methods or properties to an application that has claimed it or may fire some events only to this application.

Events Updated in Release 1.12

POS for .NET implements UnifiedPOS events as standard .NET events with multicast delegates.

The events inform an application of various activities or changes with a device, or when a device is added or removed. The event types are as follows:

|  |  |
| --- | --- |
| **Event** | **Description** |
| **DataEvent** | Input data has been placed into device class-specific properties |
| **ErrorEvent** | An error has occurred during event-driven input or asynchronous output. |
| **StatusUpdateEvent** | Reports a change in the device’s status. |
| **OutputCompleteEvent** | An asynchronous output has successfully completed. |
| **DirectIOEvent** | This event may be defined by a Service Object provider for purposes not covered by the specification. |

The Service Object queues events as they occur. Queued events are delivered to the application when conditions are correct. Conditions that delay the delivery of events include:

* The application has set the property **FreezeEvents** to TRUE.
* The event type is **DataEvent** or an input **ErrorEvent**, but the property **DataEventEnabled** is FALSE.

Unless specified otherwise, properties that convey device state information (e.g., **JrnEmpty** and **DrawerOpened**) are kept current while the device is enabled, regardless of the setting of the **FreezeEvents** property.

**Note:** The following event terminology is used in this document.

*Queue* When the Service Object determines that an event needs to be fired to the application, it queues the event on an internal event queue.

*Deliver* When the event queue is non-empty, and all conditions are met for the top event on the queue, this event is removed from the queue and delivered to the application.

*Fire* The combination of queuing and delivering an event. Sometimes, the term is used more loosely and may only refer to one of these steps. The reader should differentiate these cases by context.

Rules on the management of the queue of events are:

* The Service Object can only queue new events while the device is enabled.
* The Service Object can deliver queued events until the application calls the **Release** method (for exclusive-use devices) or the **Close** method (for any device), at which time any remaining events are deleted.
* For input devices, the **ClearInput** method clears data and input error events. While within an event handler, the application may access properties and call methods. However, the application must not call the **Release** or **Close** methods from an event handler, because **Release** may shut down event handling (possibly including a thread that caused the event to be delivered) and **Close** must shut down event handling before returning.

Input Model Updated in Release 1.12

The POS for .NET input model supports event-driven input. Event-driven input allows input data to be received after **DeviceEnabled** is set to TRUE. Received data is queued as a **DataEvent**, which is delivered to the application when preconditions are correct. If the **AutoDisable** property is TRUE when data is received, then the control will automatically disable itself, setting **DeviceEnabled** to FALSE. This will inhibit the Service Object from queuing further input and, when possible, physically disable the device.

When the application is ready to receive input from the device, it sets the **DataEventEnabled** property to TRUE. Then, when input is received (usually because of a hardware interrupt), the Control enqueues and delivers a **DataEvent**. (If input has already been enqueued, the **DataEvent** will be delivered.) This event may include input status information through a numeric parameter. The Control places the input data plus other information as needed into device-specific properties just before the event is fired.

Just before delivering this event, the Control disables further data events by setting the **DataEventEnabled** property to FALSE. This causes subsequent input data to be enqueued by the Control while the application processes the current input and associated properties. When the application has finished the current input and is ready for more data, it re-enables events by setting **DataEventEnabled** to TRUE.

If the input device is an exclusive-use device, the application must both claim and enable the device before the device begins reading input.

For sharable input devices, one or more applications must open and enable the device before the device begins reading input. An application must call the **Claim** method to request exclusive access to the device before the Control will send data to it using the **DataEvent**. If event-driven input is received, but no application has claimed the device, then the input is buffered until an application claims the device (and the **DataEventEnabled** property is TRUE). This behavior allows orderly sharing of the device between multiple applications, effectively passing the input focus between them.

If the Control encounters an error while gathering or processing event-driven input, then the Control changes its state to Error, and enqueues one or two **ErrorEvent**s to alert the application of the error condition. This event (or events) is not delivered until the **DataEventEnabled** property is TRUE, so that orderly application sequencing occurs.

Unlike a **DataEvent**, the Control does not disable further **DataEvent**s or input **ErrorEvent**s; it leaves the **DataEventEnabled** property value at TRUE. Note that the application may set **DataEventEnabled** to FALSE within its event handler if subsequent input events need to be disabled for a period.

Error events are delivered with the following loci:

**InputData** – Only queued if the error occurred while one or more **DataEvent** events are queued. It is enqueued ahead of all **DataEvent**s. This event gives the application the ability to immediately clear the input, or to optionally alert the user to the error and process the buffered input.

The latter case may be useful with a Scanner Control. The user can be immediately alerted to the error so that no further items are scanned until the error is resolved. Any previously scanned items can then be successfully processed before error recovery is performed.

**Input** – Delivered when an error has occurred and there is no data available. (A typical implementation would place it at the tail of the event queue.) If some input data was already enqueued when the error occurred, then an **ErrorEvent** with the locus **InputData** was queued and delivered first, and then this error event is delivered after all **DataEvent**s have been fired. (If an “**InputData**” event was delivered and the application event handler responded with a “Clear”, then this “**Input**” event is not delivered.)

The Control exits the Error state when one of the following occurs:

* The application returns from the Input **ErrorEvent**.
* The application returns from the InputData **ErrorEvent** with a Clear **ErrorResponse**.
* The application calls the **ClearInput** method.

For some Controls, the Application must call a method to begin event-driven input. After the input is received by the Control, then typically no additional input will be received until the method is called again to reinitiate input. Examples are the MICR and Signature Capture devices. This variation of event driven input is sometimes called “asynchronous input.”

The **DataCount** property can be read to obtain the number of **DataEvent**s queued by the Control.

All input queued by a Control can be deleted by calling the **ClearInput** method. **ClearInput** can be called after **Open** for sharable devices and after **Claim** for exclusive-use devices.

The general event-driven input model does not specifically rule out the definition of device classes containing methods or properties that return input data directly. Some device classes will define such methods and properties to operate in a more intuitive or flexible manner. An example is the Keylock device. This type of input is sometimes called “synchronous input.”

Output Model

The POS for .NET output model consists of two output types: synchronous and asynchronous. A device class can support one or both types, and neither type.

### Synchronous Output

This type of output is preferred when device output can be performed quickly. Its merit is simplicity.

The application calls a class-specific method to perform output. The service object does not return until the output is completed.

### Asynchronous Output Updated in Release 1.12

This type of output is preferred when device output requires slow hardware interactions. Its merit is perceived responsiveness, because the application can perform other work while the device is performing the output.

The application calls a class-specific method to start the output. The Service Object buffers the request in program memory, for delivery to the Physical Device as soon as the Physical Device can receive and process it, sets the **OutputId** property to an identifier for this request, and returns as soon as possible. When the device completes the request successfully, POS for .NET fires an **OutputCompleteEvent**. A parameter of this event contains the **OutputId** of the completed request.

If an error occurs while performing an asynchronous request, an **ErrorEvent** is fired. The application’s event handler can either retry the outstanding output or clear it. The Service Object is in the Error state while the **ErrorEvent** is in progress. (Note that if the condition causing the error was not corrected, then the Service Object can immediately reenter the Error state and fire another **ErrorEvent**.) Asynchronous output is performed on a first-in, first-out basis. All buffered output data, including all asynchronous output, can be deleted by calling **ClearOutput**. **OutputCompleteEvent**s are not fired for cleared output. This method also stops any output that may be in progress (when possible).

If an error occurs while processing a request, an **ErrorEvent** is enqueued which will be delivered to the application after the events already enqueued, including **OutputCompleteEvent**s (according to the normal Event delivery rules on page 0-19). No further asynchronous output will occur until the event has been delivered to the application. If the **ErrorResponse** is Clear, then outstanding asynchronous output is cleared. If the **ErrorResponse** is Retry, then output is retried; note that if several outputs were simultaneously in progress at the time that the error was detected, then the Service may need to retry all these outputs.

**Device Power Reporting Model**

Applications frequently need to know the power state of the devices they use. This state is managed by the **PowerState** enumeration.

Note: This model is not intended to report PC or POS Terminal power conditions (such as “on battery” and “battery low”). Reporting of these conditions is now managed by the **PosPower** enumeration.

### Model

POS for .NET segments device power into four states:

**Online** The device is powered on and ready for use. This is the “operational” state.

**Off** The device is powered off or detached from the terminal. This is a “non-operational” state.

**Offline** The device is powered on but is either not ready or not able to respond to requests. It may need to be placed online by pressing a button, or it may not be responding to terminal requests. This is a “non-operational” state.

In addition, one combination state is defined:

**OffOffline** The device is either off or offline, and the Service Object cannot distinguish these states.

Power reporting only occurs while the device is open, claimed (if the device is exclusive-use), and enabled.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Note** – Enabled/Disabled vs. Power States

These states are different and usually independent. POS for .NET defines “disabled” / “enabled” as a logical state, whereas the power state is a physical state. A device may be logically “enabled” but physically “offline”. It may also be logically “disabled” but physically “online”. Regardless of the physical power state, POS for .NET only reports the state while the device is enabled. (This restriction is necessary because a Service Object typically can only communicate with the device while enabled.) If a device is “offline”, then a Service Object may choose to fail an attempt to “enable” the device. However, once enabled, the Service Object may not disable a device based on its power state.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Power Reporting Properties**

The POS for .NET device power reporting model adds the following common elements across all device classes:

**CapPowerReporting** property: Identifies the reporting capabilities of the device. This property is a **PowerReporting** enumeration value:

**None** The Service Object cannot determine the state of the device. Therefore, no power reporting is possible.

**Standard** The Service Object can determine and report two of the power states – OffOffline (that is, off or offline) and Online.

**Advanced** The Service Object can determine and report all three power states – Online, Offline, and Off.

**PowerState** enumeration: Maintained by the Service Object at the current power condition, if it can be determined. This value can be one of:

* Unknown
* Online
* Off
* Offline
* OffOffline

**PowerNotify** property: The Application can set this property to enable power reporting via **StatusUpdateEvent**s and the **PowerState** enumeration. This property can only be set before the device is enabled (that is, before **DeviceEnabled** is set to TRUE). This restriction allows simpler implementation of power notification with no adverse effects on the application. The application is either prepared to receive notifications or does not want them and has no need to switch between these cases. This property returns a **PowerNotification** enumeration, the value of which is either Disabled or Enabled.

### Power Reporting Requirements for DeviceEnabled

The following semantics are added to **DeviceEnabled** when **CapPowerReporting** is not None, and **PowerNotify** is Enabled:

When the Control changes from **DeviceEnabled** FALSE to TRUE, then begin monitoring the power state:

If the device is Online, then:

* **PowerState** is set to Online.
* A StatusUpdateEvent is fired with StatusUpdateEventArgs.Status property set to Online.

If the device power state is Off, Offline, or OffOffline, then the Control can choose to fail the enable, throwing a **PosControlException** and setting **ErrorCode** to NoHardware or OffLine.

However, if there are no other conditions that cause the enable to fail, and the Control chooses to return success for the enable, then:

* **PowerState** is set to Off, Offline, or OffOffline.
* A **StatusUpdateEvent** is fired with the **StatusUpdateEventArgs**.**Status** property set to PowerOff, Offline, or OffOffline.

Device Information Reporting Model

POS Applications, as well as System Management agents, frequently need to monitor the current configuration and usage metrics of the various POS devices that are attached to the POS terminal.

Examples of configuration data are the device’s serial number, firmware version, and connection type. Examples of usage data for the POSPrinter device are the Number of Lines Printed, Number of Hours Running, Number of paper cuts, and so on. Examples of usage data for the Scanner device are the Number of scans, Number of Hours Running, etc. Examples of usage data for the MSR device are the Number of successful swipes, Number of swipes resulting in errors, Number of Hours Running, etc.

In some cases, the data may be accumulated and stored within the device itself. In other cases, the data may be accumulated by the Service and stored, possibly on the POS terminal or store controller.

For multiple applications (for example a POS application and a System Management application) to obtain statistics from the same device, proper care must be taken by both applications so that the device can be made accessible when required. This is done by using the **Claim** method and by setting **DeviceEnabled** to TRUE when access to a device is required and then setting **DeviceEnabled** to FALSE and using the **Release** method when access to the device is no longer needed. Coordination of device access via this mechanism is the responsibility of the applications themselves.

### Statistics Reporting Properties and Methods

The UnifiedPOS device information reporting model adds the following common properties and methods across all device classes.

* **CapStatisticsReporting** property. Identifies the reporting capabilities of the device. When **CapStatisticsReporting** is FALSE, then no statistical data regarding the device is available. This is equivalent to Services compatible with prior versions of the specification. When **CapStatisticsReporting** is TRUE, then some statistical data for the device is available.
* **CapUpdateStatistics** property. Defines whether gathered statistics (or some of them) can be reset/updated by the application. This property is only valid if **CapStatisticsReporting** is TRUE. When **CapUpdateStatistics** is FALSE, then none of the statistical data can be reset/updated by the application. Otherwise, when **CapUpdateStatistics** is TRUE, then (some of) the statistical data can be reset/updated by the application.
* **ResetStatistics** method. Can only be called if both **CapStatisticsReporting** and **CapUpdateStatistics** are TRUE. This method resets one, some, or all the resettable device statistics to zero.
* **RetrieveStatistics** method. Can only be called if **CapStatisticsReporting** is TRUE. This method retrieves one, some, or all the accumulated statistics for the device.
* **UpdateStatistics** method. Can only be called if both **CapStatisticsReporting** and **CapUpdateStatistics** are TRUE. This method updates one, some, or all the resettable device statistics to the supplied values.

**POS for .NET Component Descriptions**

### POS for .NET Data Types Updated in Release 1.11

The parameter and return types specified in the POS for .NET descriptions are as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **C# Type** | **VB.NET Type** | **.NET Framework Type** | **Description** | **UnifiedPOS Type** |
| bool | Boolean | System.Boolean | A Boolean value (TRUE or FALSE). | *boolean* |
| byte | Byte | System.Byte | Arbitrary binary data. | *byte* |
| byte[] | Byte() | System.Array with array element type System.Byte | Arbitrary binary data array. | *binary* |
| decimal | Decimal | System.Decimal | A currency value. | *currency* |
| int | Integer | System.Int32 | Signed 32-bit integer. | *int32* |
| int[] | Integer() | System.Array with array element type System.Int32 | Signed 32-bit integer array. | *int32 array* |
| CultureInfo | CultureInfo | System.  Globalization.  CultureInfo | Provides information about a specific culture, such as the names of the culture, the writing system, the calendar used, and how to format dates and sort strings. | *nls* |
| object | Object | System.Object | An object reference. This will usually be a subclass to the root of the class hierarchy to provide a Device Service-specific parameter for **directIO** or **DirectIOEvent**. | *object* |
| Point[] | Point() | System.Array with array element type System.Drawing.  Point | An array of ordered pairs of integer x- and y-coordinates that define a point in a two-dimensional plane. | *array of points* |
| string | String | System.String | An immutable, fixed-length string of Unicode characters. | *string* |

POS for .NET Common Properties, Methods, Events, Statistics, and Constants

#### Common Properties Updated in Release 1.11

|  |  |
| --- | --- |
| **Name** | **Type** |
| **AutoDisable** | bool |
| **CapCompareFirmwareVersion** | bool |
| **CapPowerReporting** | PowerReporting |
| **CapStatisticsReporting** | bool |
| **CapUpdateFirmware** | bool |
| **CapUpdateStatistics** | bool |
| **CheckHealthText** | string |
| **Claimed** | bool |
| **Compatibility** | DeviceCompatibilities |
| **DataCount** | int |
| **DataEventEnabled** | bool |
| **DeviceDescription** | string |
| **DeviceEnabled** | bool |
| **DeviceName** | string |
| **DevicePath** | string |
| **FreezeEvents** | bool |
| **OutputId** | int |
| **PowerNotify** | PowerNotification |
| **PowerState** | PowerState |
| **ServiceObjectDescription** | string |
| **ServiceObjectVersion** | System.version |
| **State** | ControlState |
| **SynchronizingObject** | System.ComponentModel.ISynchronizeInvoke |

The common properties are explained in detail further below.

#### Common Methods Updated in Release 1.11

The following are POS for .NET implementation-specific definitions of Common Methods:

CheckHealth ( HealthCheckLevel level );

Claim ( int timeout );

ClearInput ();

ClearInputProperties ();

ClearOutput ();

Close ();

CompareFirmwareVersion ( string filename );

DeleteConfigurationProperty ( string propertyName );

DirectIO ( int command, int data, object obj );

GetConfigurationProperty ( string propertyName );

Invoke ( Delegate method, object[] args );

Open ();

Release ();

ResetStatistics ();

ResetStatistics ( StatisticCategories statistics );

ResetStatistics ( string[] statistics );

RetrieveStatistics ( StatisticCategories statistics );

RetrieveStatistics ( string[] statistics );

RetrieveStatistic ( string statistic );

UpdateFirmware ( string filename );

UpdateStatistic ( string name, object value );

UpdateStatistics ( Statistic[] statistics );

UpdateStatistics ( StatisticCategories statistics, object value );

The common methods are explained in detail further below.

#### Common Events

Events in the .NET Framework are based on the delegate model. For more information about the delegate model, on how to consume events in applications, and how to raise events from a class, see [http://msdn.microsoft.com/library/](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguide/html/cpconevents.asp)default.asp?url=/library/en-us/cpguide/html/cpconevents.asp.

The following are POS for .NET implementation-specific definitions of Common Events:

DataEventHandler DataEvent;

DirectIOEventHandler DirectIOEvent;

DeviceErrorEventHandler ErrorEvent;

OutputCompleteEventHandler OutputCompleteEvent;

StatusUpdateEventHandler StatusUpdateEvent;

The common events are explained in detail further below.

#### Common Statistics

StatisticUnifiedPOSVersion = “UnifiedPOSVersion”;

StatisticDeviceCategory = “DeviceCategory”;

StatisticManufacturerName = “ManufacturerName”;

StatisticModelName = “ModelName”;

StatisticSerialNumber = “SerialNumber”;

StatisticManufactureDate = “ManufactureDate”;

StatisticMechanicalRevision = “MechanicalRevision”;

StatisticFirmwareRevision = “FirmwareRevision”;

**StatisticInterface** = “Interface”;

StatisticInstallationDate = “InstallationDate”;

StatisticHoursPoweredCount = “HoursPoweredCount”;

StatisticCommunicationErrorCount = “CommunicationErrorCount”;

#### Common Constants

int WaitForever = -1;

int StatusPowerOnline = 2001;

int StatusPowerOff = 2002;

int StatusPowerOffline = 2003;

int StatusPowerOffOffline = 2004;

int ExtendedErrorStatistics = 280;

Common Properties

AutoDisable Property

Type bool

Remarks If true, the Service will set **DeviceEnabled** to false after it receives and enqueues data as a **DataEvent**. Before any additional input can be received, the application must set **DeviceEnabled** to true.

If false, the Service does not automatically disable the device when data is received.

This property provides the application with an additional option for controlling the receipt of input data. If an application wants to receive and process only one input, or only one input at a time, then this property should be set to true. This property applies only to event-driven input devices.

This property is initialized to false by the **open** method.

Errors A **PosControlException** may be thrown when this property is accessed. For further information, see “**Exceptions**” on page C-39.

CapCompareFirmwareVersion Property Added in Release 1.11

Type bool

RemarksIf true, then the Service/device supports comparing the version of the firmware in the physical device against that of a firmware file.

Errors A **PosControlException** may be thrown when this property is accessed. For further information, see “**Exceptions**” on page C-39.

CapPowerReporting Property

Type PowerReporting

RemarksIdentifies the reporting capabilities of the device. Possible values are defined by the **PowerReporting** enumeration.

The service object should then set **PowerReporting** based on the capabilities of the device.

The power reporting values are:

Value Meaning

*None* The Service Object cannot determine the state of the device. Therefore, no power reporting is possible.

*Standard* The Service Object can determine and report two of the power states – OffOffLine (that is, off or offline) and Online.

*Advanced* The Service Object can determine and report all three power states – Off, OffLine, and OnLine.

Errors None.

CapStatisticsReporting Property

Type bool

RemarksIf set to TRUE, the device accumulates and can provide various statistics regarding usage. The information accumulated is device-specific and can be retrieved using the **RetrieveStatistic**(s) method.

Errors A **PosControlException** may be thrown when this property is accessed. For further information, see “**Exceptions**” on page C-39.

CapUpdateFirmware Property Added in Release 1.11

Type bool

RemarksIf true, then the device’s firmware can be updated via the **UpdateFirmware** method.

Errors A **PosControlException** may be thrown when this property is accessed. For further information, see “**Exceptions**” on page C-39.

CapUpdateStatistics Property

Type bool

Remarks If set to TRUE, some or all the device statistics can be reset to 0 (zero) using the **ResetStatistic**(s) methods or updated using the **UpdateStatistic**(s) methods.

If the **CapStatisticsReporting** property is set to FALSE, **CapUpdateStatistics** will always be FALSE.

Errors A **PosControlException** may be thrown when this property is accessed. For further information, see “**Exceptions**” on page C-39.

CheckHealthText Property

Type string

RemarksContains text indicating the health of the device. Updated by the service object when the application calls the **CheckHealth** method.

Errors A **PosControlException** may be thrown when this property is accessed. For further information, see “**Exceptions**” on page C-39.

Claimed Property

Type bool

RemarksIf TRUE, the device is claimed for exclusive access. If FALSE, the device is released for sharing with other applications.

Exclusive use devices must be claimed using the **Claim** method before the service object will allow access to many of its methods and properties, and before the service object will fire events to the application.

Errors A **PosControlException** may be thrown when this property is accessed. For further information, see “**Exceptions**” on page C-39.

Compatibility Property

Type DeviceCompatibilities

RemarksIndicates the compatibility level of a device.

This property has one of the following values:

Member Name Description

*CompatibilityLevel1* Indicates compatibility with any .NET service object.

*Opos* Indicates compatibility with any COM service object.

OposAndCompatibilityLevel1

Indicates compatibility with any .NET or COM service object.

Errors A **PosControlException** may be thrown when this property is accessed. For further information, see “**Exceptions**” on page C-39.

DataCount Property

Type int

Remarks Holds the number of enqueued **DataEvent**s.

The application may read this property to determine whether additional input is enqueued from a device but has not yet been delivered because of other application processing, freezing of events, or other causes.

This property is initialized to zero by the **open** method.

Errors A **PosControlException** may be thrown when this property is accessed. For further information, see “**Exceptions**” on page C-39.

DataEventEnabled Property

Type bool

Remarks If true, a **DataEvent** will be delivered as soon as input data is enqueued. If changed to true and some input data is already queued, then a **DataEvent** is delivered immediately. (Note that other conditions may delay “immediate” delivery: if **FreezeEvents** is true or another event is already being processed at the application, the **DataEvent** will remain queued at the Service until the condition is corrected.)

If false, input data is enqueued for later delivery to the application. Also, if an input error occurs, the **ErrorEvent** is not delivered while this property is false.

This property is initialized to false by the **open** method.

Errors A **PosControlException** may be thrown when this property is accessed. For further information, see “**Exceptions**” on page C-39.

DeviceDescription Property

Type string

RemarksContains text identifying the device and any pertinent information about it. A sample of the text might be:

“NCR 7192-0184 Printer, Japanese Version”

This property is initialized when the application calls the **Open** method.

Errors None.

DeviceEnabled Property

Type bool

RemarksWhen TRUE, the device has been placed in an operational state. If changed to TRUE, then the device is brought to an operational state.

When FALSE, the device has been disabled. If changed to FALSE, then the device is physically disabled when possible. Any subsequent input will be discarded, and output operations are disallowed.

Changing **DeviceEnabled** usually does not physically affect output devices. For consistency, however, the application must set **DeviceEnabled** to TRUE before using output devices.

Errors A **PosControlException** may be thrown when this property is accessed. For further information, see “**Exceptions**” on page C-39.

DeviceName Property

Type string

RemarksContains a short string identifying the device and any pertinent information about it.

This is a condensed version of **DeviceDescription** and should be limited to 30 characters.

**DeviceName** will typically be used to identify the device in an application message box, where the full description is too verbose. A sample **DeviceName** string is:

“NCR 7192 Printer, Japanese”

Errors None.

DevicePath Property Updated in Release 1.13

Type string

RemarksContains the hardware path of a device. Note: This is a common property for .NET service objects, but it is only intended for usage between the .NET service object and the POS for .NET system. The Application should not access this property. A .NET service object that attempts to change this non-public **DevicePath** property to public will result in an exception error.

The **PosExplorer** class attempts to initialize **DevicePath** to the hardware path of the physical device using the following algorithm:

* If the physical hardware supports Plug and Play and the service object is mapped to a specific hardware ID via the **HardwareId** custom attribute or a configuration XML file, **PosExplorer** class will set **DevicePath** to the **HardwarePath** of the physical device. Service objects can typically use this **DevicePath** to directly access the device.
* If the device does not support Plug and Play, but has been configured via Posdm.exe or WMI, **DevicePath** will be set to the path specified when the device was configured.
* If the device does not support Plug and Play and has not been configured via Posdm.exe or WMI, **DevicePath** will be set to empty string (“”) and must be set by the service object before the **Open** method in the base/basic class can be called.

Errors A **PosControlException** may be thrown when this property is accessed. For further information, see “**Exceptions**” on page C-39.

FreezeEvents Property Updated in Release 1.12

Type bool

Remarks When set to TRUE, the application has requested that the service object not deliver events. Events will be queued by the service object but not delivered until the application changes FreezeEvents to FALSE.

When set to FALSE, the application allows events to be delivered. If some events have been held while events were frozen, and all other conditions are correct for delivering the events, changing **FreezeEvents** to FALSE will allow these events to be delivered.

An application can choose to freeze events for a specific sequence of code where interruption by an event is not desirable.

Unless specified otherwise, properties that convey device state information (e.g., **JrnEmpty** and **DrawerOpened**) are kept current while the device is enabled, regardless of the setting of the **FreezeEvents** property.

Errors A **PosControlException** may be thrown when this property is accessed. For further information, see “**Exceptions**” on page C-39.

OutputId Property

Type int

Remarks Holds the identifier of the most recently started asynchronous output.

When a method successfully initiates an asynchronous output, the Service assigns an identifier to the request. When the output completes, an **OutputCompleteEvent** will be enqueued with this output ID as a parameter.

The output ID numbers are assigned by the Service and are guaranteed to be unique among the set of outstanding asynchronous outputs. No other facts about the ID should be assumed.

Errors A **PosControlException** may be thrown when this property is accessed. For further information, see “**Exceptions**” on page C-39.

PowerNotify Property

Type PowerNotification

Remarks Contains the type of power notification selection made by the application. Possible values are defined by the **PowerNotification** enumeration.

**PowerNotify** can only be set while the device is disabled, that is, while the **DeviceEnabled** property is set to FALSE.

Errors A **PosControlException** may be thrown when this property is accessed. For further information, see “**Exceptions**” on page C-39.

PowerState Property

Type PowerState

RemarksContains the current power condition. Possible values are defined by the **PowerState** enumeration.

When **PowerNotify** is set to enabled and **DeviceEnabled** is TRUE, **PowerState** is updated as the service object detects power condition changes. When the power state changes, the service object updates **PowerState** and queues a **StatusUpdateEvent** event, notifying the application.

Errors None.

ServiceObjectDescription Property

Type string

RemarksContains a string identifying the service object supporting the device and the company that produced it.

A sample ServiceObjectDescription string is:

“TM-T88IV Printer POS for .Net Service Driver, (C) 2005 Epson”

Errors A **PosControlException** may be thrown when this property is accessed. For further information, see “**Exceptions**” on page C-39.

ServiceObjectVersion Property

Type System.version

Remarks **ServiceObjectVersion** holds the service object version number. Version numbers consist of two to four integers, Major, Minor, Build, and Revision. Build and Revision are optional, but Revision is optional only if Build is not specified.

The Major and Minor version numbers correspond to the UnifiedPOS version implemented by the service object. A service object that implements the UnifiedPOS 1.8 specification would set Major=1 and Minor=8. The Build and Revision version numbers are optional and can be used by the service object to track its internal version.

Errors A **PosControlException** may be thrown when this property is accessed. For further information, see “**Exceptions**” on page C-39.

State Property

Type ControlState

RemarksContains the current state of the device. Possible values are defined by the **ControlState** enumeration.

State is set to **ControlState.Idle** by the Open method and is always readable, regardless of the state of the device.

Errors None.

SynchronizingObject Property

Type System.ComponentModel.ISynchronizeInvoke

RemarksContains an instance of the **ISynchronizeInvoke** class. Applications can use this property to specify the thread events that are to be delivered on. If **SynchronizingObject** is set to null, events are delivered on an internal thread owned by the service object. Applications using Windows Forms should set **SynchronizationObject** to the **this** pointer of the main Form class so that events are delivered on the main application thread ... as required by the Form class.

Errors A **PosControlException** may be thrown when this property is accessed. For further information, see “**Exceptions**” on page C-39.

## **Common Methods**

CheckHealth Method

Syntax string CheckHealth ( HealthCheckLevel *level* );

RemarksThe application calls **CheckHealth** to test the state of a device. **CheckHealth** is always performed synchronously. The service object returns a string indicating the health level and updates the **CheckHealthText** property.

The *level* parameter indicates the type of health check to be performed on the device. Possible values are defined by the **HealthCheckLevel** enumeration.

Value Meaning

*Internal* Perform a health check that does not physically change the device. The device is tested by internal tests to the extent possible.

*External* Perform a more thorough test that may change the device. For example, a pattern may be printed on the printer.

*Interactive* Perform an interactive test of the device. The supporting Service Object will typically display a modal dialog box to present test options and results.

Errors A **PosControlException** may be thrown when this method is invoked. For further information, see “**Exceptions**” on page C-39.

CheckHealth may throw the following PosControlException:

|  |  |
| --- | --- |
| **ErrorCode Value** | **Description** |
| *Illegal* | The specified health check level is not supported by the service object. |

Claim Method

Syntax void Claim ( int *timeout* );

RemarksThe application calls **Claim** to request exclusive access to the device. Many devices require an application to claim them before they can be used.

If the timeout parameter is set to 0 (zero), the method attempts to claim the device, then returns the appropriate status immediately. If *timeout* is set to **WaitForever** (-1), **Claim** waits until exclusive access is satisfied.

An application can claim a device more than once without generating an error. When **Claim** is successful, the **Claimed** property is set to TRUE.

The *timeout* parameter contains the maximum number of milliseconds to wait for exclusive access to be satisfied.

Errors A **PosControlException** may be thrown when this method is invoked. For further information, see “**Exceptions**” on page C-39.

**Claim** may throw the following **PosControlException**s:

|  |  |
| --- | --- |
| **ErrorCode Value** | **Description** |
| *Illegal* | One of the following conditions has occurred:  The device cannot currently be claimed for exclusive access; or  a value of less than -1 has been specified for the timeout parameter. |
| *Timeout* | Another application has exclusive access to the device and did not relinquish control before timeout milliseconds expired. |

ClearInput Method

Syntax void ClearInput ();

Remarks Clears all device input that has been buffered.

Any data events or input error events that are enqueued – usually waiting for **DataEventEnabled** to be set to true and **FreezeEvents** to be set to false – are also cleared.

Errors A **PosControlException** may be thrown when this method is invoked. For further information, see “**Exceptions**” on page C-39.

ClearInputProperties Method Added in Release 1.11

Syntax void ClearInputProperties ();

Remarks Sets all data properties that were populated because of firing a **DataEvent** or **ErrorEvent** back to their default values. This does not reset the **DataCount** or **State** properties.

Errors A **PosControlException** may be thrown when this method is invoked. For further information, see “**Exceptions**” on page C-39.

ClearOutput Method

Syntax void ClearOutput ();

Remarks Clears all buffered output data, including all asynchronous output. Also, when possible, halts outputs that are in progress.

Any output error events that are enqueued – usually waiting for **FreezeEvents** to be set to false – are also cleared.

Errors A **PosControlException** may be thrown when this method is invoked. For further information, see “**Exceptions**” on page C-39.

Close Method

Syntax void Close ( );

RemarksThe application calls **Close** to release the device and its resources.

If the **DeviceEnabled** property is set to TRUE, the device will be disabled. If the **Claimed** property is set to TRUE, the device will be released.

Errors A **PosControlException** may be thrown when this method is invoked. For further information, see “**Exceptions**” on page C-39.

**Close** may throw the following **PosControlException**s:

|  |  |
| --- | --- |
| **ErrorCode Value** | **Description** |
| *Busy* | The **State** property is set to **ControlState.Busy**, indicating that the device is currently in use and cannot be shut down. |
| *Closed* | The device is already closed. |

CompareFirmwareVersion Method Added in Release 1.11

Syntax CompareFirmwareResult CompareFirmwareVersion (string firmwareFileName );

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| *firmwareFileName* | Specifies either the name of the file containing the firmware or a file containing a set of firmware files whose versions are to be compared against those of the device. |

Remarks This method determines whether the version of the firmware contained in the specified file is newer than, older than, or the same as the version of the firmware in the physical device.

The Service should check that the specified firmware file exists and that its contents are valid for this device before attempting to perform the comparison operation.

The result of the comparison is returned in the enumeration **CompareFirmwareResult** and will be one of the following values:

Value Meaning

Older Indicates that the version of one or more of the firmware files is older than the firmware in the device and that none of the firmware files is newer than the firmware in the device.

Same Indicates that the versions of all the firmware files are the same as the firmware in the device.

Newer Indicates that the version of one or more of the firmware files is newer than the firmware in the device and that none of the firmware files is older than the firmware in the device.

Different Indicates that the version of one or more of the firmware files is different than the firmware in the device, but either:

* The chronological relationship cannot be determined, or
* The relationship is inconsistent -- one or more are older while one or more are newer.

Unknown Indicates that a relationship between the two firmware versions could not be determined.

A reason for this enumeration could be an attempt to compare Japanese and US versions of firmware.

If the *firmwareFileName* parameter specifies a file list, all the component firmware files should reside in the same directory as the firmware list file. This will allow for distribution of the updated firmware without requiring a modification to the firmware list file

Errors A **PosControlException** may be thrown when this method is invoked. For further information, see “**Exceptions**” on page C-39.

CompareFirmwareVersion may throw the following PosControlExceptions:

|  |  |
| --- | --- |
| **ErrorCode Value** | **Description** |
| *Illegal* | **CapCompareFirmwareVersion** is false. |
| *NoExist* | The file specified by *firmwareFileName* does not exist or,  if *firmwareFileName* specifies a file list, one or more of the component firmware files are missing. |
| *Extended* | ***ErrorCodeExtended*** = EFIRMWARE\_BAD\_FILE:  The specified firmware file or files exist, but one or more are either not in the correct format or are corrupt. |

DirectIO Method

Syntax DirectIOData DirectIO ( int *command*, int *data*, object *obj* );

RemarksThe application calls **DirectIO** to communicate directly with the service object.

Using **DirectIO** allows a service object to provide functionality to the application that is not otherwise supported by the standard service interface for its device class. Depending on the service object’s definition of the command, **DirectIO** may be asynchronous or synchronous.

Errors A **PosControlException** may be thrown when this method is invoked. For further information, see “**Exceptions**” on page C-39.

**DirectIO** returns an instance of the **DirectIOData** structure.

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| *command* | The command numbers. Specific values are assigned by the service object. |
| *data* | Additional numeric data. Specific values vary by command and the service object. |
| *obj* | Additional data supplied by the service object. Specific values vary by command and by what the service object chooses to transmit. |

Open Method

Syntax void Open ( );

RemarksThe application calls **Open** to open a device for subsequent input/output processing. **Open** initializes the values of numerous properties, including **DataEventEnabled**, **FreezeEvents**, **AutoDisable**, **Claimed**, and so on.

Errors A **PosControlException** may be thrown when this method is invoked. For further information, see “**Exceptions**” on page C-39.

**Open** may throw the following **PosControlException**:

|  |  |
| --- | --- |
| **ErrorCode Value** | **Description** |
| *Illegal* | The device is already opened |

Release Method

Syntax void Release ( );

RemarksThe application calls **Release** to release exclusive access to the device.

If the **DeviceEnabled** property is set to TRUE, and the device is an exclusive-use device, the device is first disabled. (**Release** does not change the device-enabled state of sharable devices.) If **Release** is successful, it sets the **Claimed** property to FALSE.

Errors A **PosControlException** may be thrown when this method is invoked. For further information, see “**Exceptions**” on page C-39.

**Release** may throw the following **PosControlException**s:

|  |  |
| --- | --- |
| **ErrorCode Value** | **Description** |
| *Busy* | The device is in use. |
| *Illegal* | One of the following conditions has occurred:  The application does not have exclusive access to the device; or  the device is not claimed. |

ResetStatistic Method (string)

Syntax void ResetStatistic ( string *statistic* );

Remarks*statistic* specifies the statistic that is to be reset.

The application calls **ResetStatistic** to reset the specified statistic to 0 (zero).

For **ResetStatistic** to be successful, both the **CapStatisticsReporting** and **CapUpdateStatistics** properties must be set to TRUE.

**ResetStatistic** is always executed synchronously.

Errors A **PosControlException** may be thrown when this method is invoked. For further information, see “**Exceptions**” on page C-39.

ResetStatistic may throw the following PosControlExceptions:

|  |  |
| --- | --- |
| **ErrorCode Value** | **Description** |
| *Illegal* | One of the following conditions has occurred:  Either the **CapStatisticsReporting** or **CapUpdateStatistics** property is set to FALSE;  The statistic parameter is null; or  The specified *statistic* does not exist. |
| *Extended* | **ExtendedErrorStatistics**. The specified *statistic* can not be reset. |

ResetStatistics Method ()

Syntax void ResetStatistics ( );

RemarksResets all statistics associated with a device to 0 (zero).

For **ResetStatistics** to be successful, both the **CapStatisticsReporting** and **CapUpdateStatistics** properties must be set to TRUE.

**ResetStatistics** is always executed synchronously.

Errors A **PosControlException** may be thrown when this method is invoked. For further information, see “**Exceptions**” on page C-39.

ResetStatistics may throw the following PosControlExceptions:

|  |  |
| --- | --- |
| **ErrorCode Value** | **Description** |
| *Illegal* | The **CapStatisticsReporting** or  **CapUpdateStatistics** property is set to FALSE. |
| *Extended* | **ExtendedErrorStatistics**. At least one of the specified statistics could not be reset. |

ResetStatistics Method (StatisticsCategories)

Syntax void ResetStatistics ( StatisticCategories *statistics* );

RemarksResets all statistics for a specified category to 0 (zero).

For **ResetStatistics** to be successful, both the **CapStatisticsReporting** and **CapUpdateStatistics** properties must be set to TRUE.

**ResetStatistics** is always executed synchronously.

The *statistics* parameter contains the category of statistics the application wants to reset for the device. Possible categories are defined by the **StatisticsCategories** enumeration.

Errors A **PosControlException** may be thrown when this method is invoked. For further information, see “**Exceptions**” on page C-39.

ResetStatistics may throw the following PosControlExceptions:

|  |  |
| --- | --- |
| **ErrorCode Value** | **Description** |
| *Illegal* | One of the following conditions has occurred:  The **CapStatisticsReporting** or **CapUpdateStatistics** property is set to FALSE; or  the specified statistics category is not valid. |
| *Extended* | **ExtendedErrorStatistics**. At least one of the specified statistics could not be reset. |

ResetStatistics Method (String[])

Syntax void ResetStatistics ( string [] *statistics* );

RemarksResets the specified statistics to 0 (zero).

For **ResetStatistics** to be successful, both the **CapStatisticsReporting** and **CapUpdateStatistics** properties must be set to TRUE.

**ResetStatistics** is always executed synchronously.

The *statistics* parameter contains a comma-separated string of statistics.

Errors A **PosControlException** may be thrown when this method is invoked. For further information, see “**Exceptions**” on page C-39.

ResetStatistics may throw the following PosControlExceptions:

|  |  |
| --- | --- |
| **ErrorCode Value** | **Description** |
| *Illegal* | One of the following conditions has occurred:  The **CapStatisticsReporting** or **CapUpdateStatistics** property is set to FALSE; or  One of the specified *statistics* is not defined. |
| *Extended* | **ExtendedErrorStatistics**. At least one of the specified statistics could not be reset. |

RetrieveStatistic Method (string)

Syntax string RetrieveStatistic ( string *statistic* );

RemarksThe application calls **RetrieveStatistic** to retrieve the specified device statistic.

**RetrieveStatistic** is always executed synchronously.

The *statistic* parameter specifies the statistic that is to be retrieved.

**RetrieveStatistic** returns and XML string of statistics if successful.

Errors A **PosControlException** may be thrown when this method is invoked. For further information, see “**Exceptions**” on page C-39.

RetrieveStatistic may throw the following PosControlException:

|  |  |
| --- | --- |
| **ErrorCode Value** | **Description** |
| *Illegal* | One of the following conditions has occurred:  The **CapStatisticsReporting** property is set to FALSE, indicating that the device does not support statistics reporting;  The *statistic* parameter is null or has a length of 0 (zero); or  the specified statistic does not exist. |

RetrieveStatistics Method ()

Syntax string RetrieveStatistics ( );

RemarksThe application calls **RetrieveStatistics** to retrieve all device statistics.

**RetrieveStatistics** is always executed synchronously.

**RetrieveStatistics** returns an XML string of statistics if successful.

Errors A **PosControlException** may be thrown when this method is invoked. For further information, see “**Exceptions**” on page C-39.

RetrieveStatistics may throw the following PosControlException:

|  |  |
| --- | --- |
| **ErrorCode Value** | **Description** |
| *Illegal* | The **CapStatisticsReporting** property is set to FALSE, indicating that the device does not support statistics reporting. |

RetrieveStatistics Method (StatisticCategories)

Syntax string RetrieveStatistics ( StatisticCategories *statistics* );

RemarksRetrieves the statistics for the specified category.

**RetrieveStatistics** is always executed synchronously.

The *statistics* parameter contains the category of statistics the application wants to retrieve. Possible values are defined by the **StatisticCategories** enumeration.

**RetrieveStatistics** returns an XML string of statistics if successful.

Errors A **PosControlException** may be thrown when this method is invoked. For further information, see “**Exceptions**” on page C-39.

RetrieveStatistics may throw the following PosControlException:

|  |  |
| --- | --- |
| **ErrorCode Value** | **Description** |
| Illegal | One of the following conditions has occurred:  The **CapStatisticsReporting** property is set to FALSE, indicating that the device does not support statistics reporting;  The *statistics* parameter is null or has a length of 0 (zero); or the specified statistics category is invalid. |

RetrieveStatistics Method (String[])

Syntax string RetrieveStatistics ( string [] *statistics* );

RemarksRetrieves the statistics for the specified category.

**RetrieveStatistics** is always executed synchronously.

The *statistics* parameter contains a comma-separated string of statistics. Retrieves the specified string of statistics.

**RetrieveStatistics** returns an XML string of statistics if successful

Errors A **PosControlException** may be thrown when this method is invoked. For further information, see “**Exceptions**” on page C-39.

RetrieveStatistics may throw the following PosControlException:

|  |  |
| --- | --- |
| **ErrorCode Value** | **Description** |
| *Illegal* | One of the following conditions has occurred:  The **CapStatisticsReporting** property is set to FALSE, indicating that the device does not support statistics reporting;  The *statistics* parameter is null or has a length of 0 (zero); or  , one or more of the specified *statistics* do not exist. |

UpdateFirmware Method Added in Release 1.11

Syntax UpdateFirmware ( string *firmwareFileName* );

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| *firmwareFileName* | Specifies either the name of the file containing the firmware or a file containing a set of firmware files that are to be downloaded into the device. |

Remarks This method updates the firmware of a device with the version of the firmware contained or defined in the file specified by the *firmwareFileName* parameter regardless of whether that firmware’s version is newer than, older than, or the same as the version of the firmware already in the device. If the *firmwareFileName* parameter specifies a file list, all the component firmware files should reside in the same directory as the firmware list file. This will allow for distribution of the updated firmware without requiring a modification to the firmware list file.

When this method is invoked, the Service should check that the specified firmware file exists and that its contents are valid for this device. If so, this method should return immediately, and the remainder of the update firmware process should continue asynchronously.

The Service should notify the application of the status of the update firmware process by firing **StatusUpdateEvent**s with values of SUE\_UF\_PROGRESS + an integer between 1 and 100 indicating the completion percentage of the update firmware process. For application convenience, the **StatusUpdateEvent** value SUE\_UF\_COMPLETE is defined to be the same value as SUE\_UF\_PROGRESS + 100.

For consistency, the update firmware process is complete after the new firmware has been downloaded into the physical device, any necessary physical device reset has completed, and the Service and the physical device have been returned to the state they were in before the update firmware process began.

For consistency, a Service must always fire at least one **StatusUpdateEvent** with an incomplete progress completion percentage (i.e. a percentage between 1 and 99), even if the device cannot physically report the progress of the update firmware process. If the update firmware process completes successfully, the Service must fire a **StatusUpdateEvent** with a progress of 100 or use the special constant SUE\_UF\_COMPLETE, which has the same value. These Service requirements allow applications using this method to be designed to always expect some level of progress notification.

If an error is detected during the asynchronous portion of an update firmware process, one of the following **StatusUpdateEvent**s will be fired:

Value Meaning

SUE\_UF\_FAILED\_DEV\_OK The update firmware process failed but the device is still operational.

SUE\_UF\_FAILED\_DEV\_UNRECOVERABLE

The update firmware process failed, and the device is neither usable nor recoverable through software. The device requires service to be returned to an operational state.

SUE\_UF\_FAILED\_DEV\_NEEDS\_FIRMWARE

The update firmware process failed, and the device will not be operational until another attempt to update the firmware is successful.

SUE\_UF\_FAILED\_DEV\_UNKNOWN

The update firmware process failed, and the device is in an indeterminate state.

Errors A **PosControlException** may be thrown when this method is invoked. For further information, see “**Exceptions**” on page C-39.

UpdateFirmware may throw the following PosControlExceptions:

|  |  |
| --- | --- |
| **ErrorCode Value** | **Description** |
| *Illegal* | **CapUpdateFirmware** is false. |
| *NoExist* | The file specified by *firmwareFileName* does not exist or,  if *firmwareFileName* specifies a file list, one or more of the component firmware files are missing. |
| *Extended* | ***ErrorCodeExtended*** = EFIRMWARE\_BAD\_FILE:  The specified firmware file or files exist, but one or more are either not in the correct format or are corrupt. |

UpdateStatistic Method

Syntax void UpdateStatistic ( string *name*, object *value* );

RemarksThe application calls **UpdateStatistic** to update the value of a specified device statistic.

For **UpdateStatistic** to be successful, both the **CapStatisticsReporting** and **CapUpdateStatistics** properties must be set to TRUE.

**UpdateStatistic** is always executed synchronously.

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| *name* | Name of the statistic to be updated. |
| *value* | Value to which the statistic should be set. |

Errors A **PosControlException** may be thrown when this method is invoked. For further information, see “**Exceptions**” on page C-39.

UpdateStatistic may throw the following PosControlExceptions:

|  |  |
| --- | --- |
| **ErrorCode Value** | **Description** |
| *Illegal* | One of the following conditions has occurred:  The **CapStatisticsReporting** or **CapUpdateStatistics** property is set to FALSE; or  The specified statistic does not exist. |
| *Extended* | **ExtendedErrorStatistics**. The specified statistic could not be updated. |

UpdateStatistics Method (Statistic[])

Syntax void UpdateStatistics ( Statistic [] *statistics* );

RemarksUpdates a list of statistics with the corresponding specified values.

For **UpdateStatistics** to be successful, both the **CapStatisticsReporting** and **CapUpdateStatistics** properties must be set to TRUE.

**UpdateStatistics** is always executed synchronously.

The *statistics* parameter contains an array of **Statistic** class instances (name-value pairs).

Errors A **PosControlException** may be thrown when this method is invoked. For further information, see “**Exceptions**” on page C-39.

UpdateStatistics may throw the following PosControlExceptions:

|  |  |
| --- | --- |
| **ErrorCode Value** | **Description** |
| *Illegal* | One of the following conditions has occurred:  The **CapStatisticsReporting** or **CapUpdateStatistics** property is set to FALSE; or  The *statistics* parameter is null; or  One or more of the specified *statistics* does not exist. |
| *Extended* | **ExtendedErrorStatistics**. At least one of the specified *statistics* could not be updated. |

UpdateStatistics Method (StatisticCategories, Object)

Syntax void UpdateStatistics ( StatisticCategories *statistics*, object *value* );

RemarksUpdates the specified category of statistics with the specified value.

For **UpdateStatistics** to be successful, both the **CapStatisticsReporting** and **CapUpdateStatistics** properties must be set to TRUE.

**UpdateStatistics** is always executed synchronously.

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| statistics | Contains the category of statistics the application wants to update. Possible categories are defined by the **StatisticCategories** enumeration. |
| value | Contains the value to be used to update the statistics in the specified category. |

Errors A **PosControlException** may be thrown when this method is invoked. For further information, see “**Exceptions**” on page C-39.

UpdateStatistics may throw the following PosControlExceptions:

|  |  |
| --- | --- |
| **ErrorCode Value** | **Description** |
| Illegal | One of the following conditions has occurred:  The CapStatisticsReporting or CapUpdateStatistics property is set to FALSE; or  The specified statistics category is invalid. |
| Extended | **ExtendedErrorStatistics**. At least one of the specified *statistics* could not be updated. |

## **Common Events**

DataEvent Event

Remarks Fired to present input data from the device to the application. The **DataEventEnabled** property is changed 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-specific properties.

If **DataEventEnabled** is FALSE at the time that data is received, then the data is queued in an internal buffer, the device-specific input data properties are not updated, and the event is not delivered. (When this property is subsequently changed back to TRUE, the event will be delivered immediately if input data is queued and **FreezeEvents** is FALSE.)

DirectIOEvent Event

Remarks Fired by the service object to communicate information directly to the application. **DirectIOEvent** provides a means for a service object to communicate information in the form of an event to the application that would not otherwise be supported by other events or properties defined for the device. Use of this event may restrict the application from being used with other vendor’s devices which may not have any knowledge of the service object’s need for this event.

ErrorEvent Event Updated in Release 1.12

RemarksFired when an error is detected and the service object's State transitions into the error state.

Input error events are not delivered until the **DataEventEnabled** property is TRUE, so that proper application sequencing occurs.

Unlike a **DataEvent**, the Control does not disable further **DataEvent**s or input **ErrorEvent**s; it leaves the **DataEventEnabled** property value at TRUE. Note that the application may set **DataEventEnabled** to FALSE within its event handler if subsequent input events need to be disabled for a period.

OutputCompleteEvent Event

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

StatusUpdateEvent Event

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

Examples are a change in the cash drawer position (open vs. closed), a change in a POS printer sensor (form present vs. absent), or a change in the power state of the device.

When a device is enabled, the service object may fire initial **StatusUpdateEvent**s to inform the application of the device state. This behavior, however, is not required.

**POS for .NET vs. UnifiedPOS Members**

POS for .NET class member names sometimes vary from those in the UnifiedPOS specification. In many cases, the variance is only in case (.NET uses the Pascal naming convention for methods, properties, and events). For example, the common property **OutputID** in the UnifiedPOS specification is **OutputId** in POS for .NET.

For some devices, POS for .NET introduces several properties and methods not found in the UnifiedPOS specification.

The table below has examples of some of the property names that vary from the UnifiedPOS specification:

|  |  |
| --- | --- |
| **UnifiedPOS Property** | **Corresponding POS for .NET Property** |
| **CapMACCalculation** | **CapMacCalculation** |
| **DeviceServiceDescription** | **ServiceObjectDescription** |
| **DeviceServiceVersion** | **ServiceObjectVersion** |
| **OutputID** | **OutputId** |
| **POSKeyData** | **PosKeyData** |
| **POSKeyEventType** | **PosKeyEventType** |
| **PhysicalDeviceDescription** | **DeviceDescription** |
| **PhysicalDeviceName** | **DeviceName** |
| N/A | **Compatibility** |
| N/A | **DevicePath** |
| N/A | **SynchronizingObject** |

The table below includes some of the method names that vary from the UnifiedPOS specification:

|  |  |
| --- | --- |
| **UnifiedPOS Method** | **Corresponding POS for .NET Method** |
| **beginEFTTransaction** | **BeginEftTransaction** |
| **checkHealth** | **CheckHealth** |
| **claim** | **Claim** |
| **computeMAC** | **ComputeMac** |
| **DeviceServiceVersion** | **ServiceObjectVersion** |
| **directIO** | **DirectIO** |
| **enablePINEntry** | **EnablePinEntry** |
| **endEFTTransaction** | **EndEftTransaction** |
| **read** | **Read** |
| **resetStatistics** | **ResetStatistics** |
| **verifyMAC** | **VerifyMac** |
| N/A | **ResetStatistic** |
| N/A | **RetrieveStatistic** |
| N/A | **UpdateStatistic** |

The table below includes event names that vary from the UnifiedPOS specification:

|  |  |
| --- | --- |
| **UnifiedPOS Event Attribute** | **Corresponding POS for .NET EventArg Class Property** |
| **OutputID** | **OutputId** |
| N/A | **public DateTime TimeStamp {get; }** |

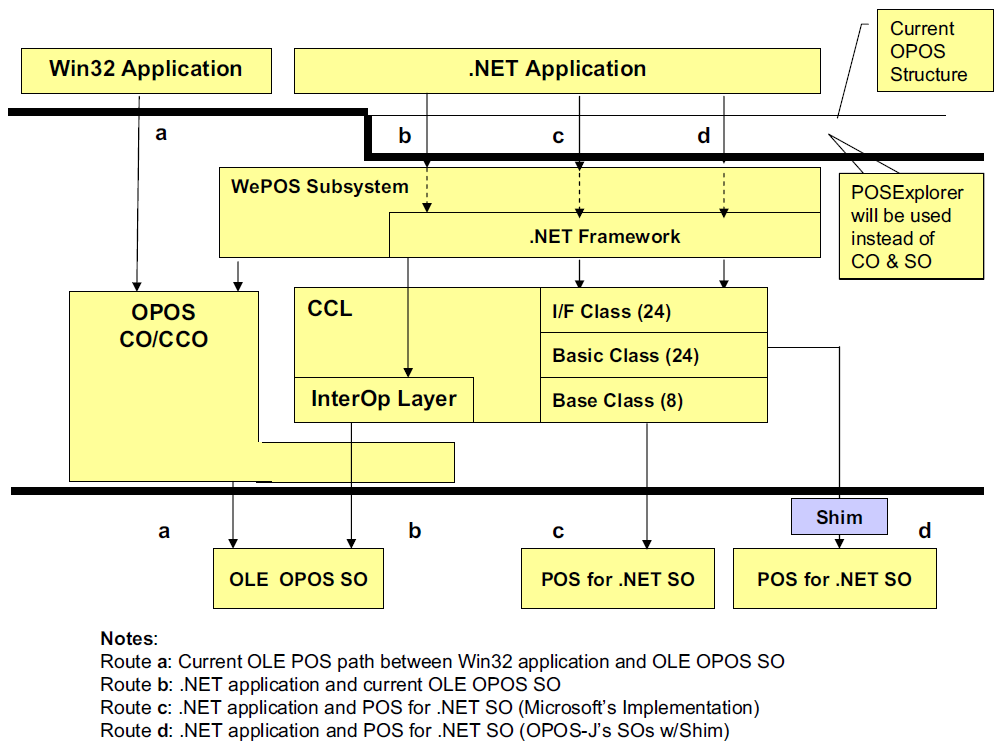
Interim Procedure Available for Legacy OPOS Services...  
Shim Code Usage Updated in Release 1.11

The .NET architecture allows for new features and functions that can be invoked using current and future Windows operating systems. To benefit from all the .NET architecture has to offer, new service objects should be written. However, to more quickly leverage existing OPOS service object source code in the .NET environment, OPOS-Japan (OPOS-J) has created a translation middle layer of software, referred to as the “Shim”. The “Shim” is a module to develop (or implement) a .NET Service Object by utilizing existing OPOS based service object naming methodologies. It is freely available for service object providers to use when porting their existing OPOS service objects to POS for .NET. Some of the reasons behind the strategy in using the Shim are as follows:

* POS for .NET extends the definitions for the UnifiedPOS methods and requires modifications in the OPOS service objects to handle these extensions. The Shim handles these extensions and masks any changes that would otherwise be required to be made to an existing OPOS service object.
* POS for .NET requires enumeration types in its usage, a feature that was not specified in an OPOS service object implementation. The Shim provides a mechanism to map constants of the parameters to an enumeration type without changing the name from the existing OPOS service object source code.
* It is important to note that the usage of the Shim does not require any changes to the .NET application; the Shim hides any OPOS and POS for .NET service object differences from the application. When a POS for .NET service object is available, it should be able to replace the Shim/OPOS service object with no required changes to the application.
* The development of the POS Application should be in accordance with the reference material outlined earlier in this appendix. The only difference is in the development of the service object used to support a UnifiedPOS, POS for .NET environment. Potentially, usage of the Shim allows for faster generation of POS for .NET service objects by allowing for greater re-usability of existing OPOS service object source code.

### Architecture Structures Added in Release 1.11

The following diagram shows the structures of the OPOS, POS for .NET, and Shim-POS for .NET architectures.



Method of Implementation

#### Shim Code Naming rules

The Shim code extends the POS for .NET Basic class as described below:

Microsoft.PointOfService.BasicServiceObjects NameSpace.

The names of the Shim classes comply with the following rule:

<DeviceCategoryName>+ShimBasic

For example:

PosPrinterShimBasic

LineDisplayShimBasic

The file name that defines the Shim class complies with the following rule:

<Class Name>.cs

For example:

PosPrinterShimBasic.cs

LineDisplayShimBasic.cs

The shim class is defined in the following NameSpace:

Opos.PointOfService.BasicShimServiceObjects.

The file that defines the specific enumeration type is specified in a separate file associated with its device category. The file name that defines this takes the same name as the header file of the OPOS Common Control Object (CCO).

For example:

Constants definition for POS Printer,   
OposPtr.cs

Constants definition for LineDisplay  
OposDisp.cs

The enumeration type name is derived from the name associated with the function parameter that uses the constants.

For example, the alignment parameter that is used with the **PrintBarCode** function supported by a POS Printer would map as follows:

OposPtr.cs

Enum BarCodeAlignment

{

Left = -1,

Center = -2,

Right = -3

}

The enumeration type is defined in the following NameSpace:

Opos.PointOfService

#### Shim Method Redefinition Rules

As noted earlier in this appendix, POS for .NET method calls are handled differently than UnifiedPOS OPOS implementations. For instance, under POS for .NET return values are used instead of OPOS requiring a separate method call to obtain the information. The Shim provides the translation code to allow for the mapping of these operational differences.

The functions of the UnifiedPOS specification that are implemented differently between POS for .NET and OPOS are redefined using an abstract attribute at the protected level.

For example, the **DirectIO** method would map as follows:

public override DirectIOData DirectIO (int command, int data, object obj)

{

;

}

protected abstract void **DirectIO** (int command, ref int data, ref object obj);

Note that the abstract function that UnifiedPOS defined, **DirectIO**, is called in a way that is consistent with the POS for .NET Application implementation requirements. However, the Shim code performs the necessary functions to process the OPOS **DirectIO** method and any other method calls to obtain the method functionality and data exchange. The Shim code then responds back to the POS for .NET Application with the functionality and result codes that are consistent with what it is expecting to see. Continuing with the example:

public override DirectIOData **DirectIO** (int command, int data, object obj)

{

this.**DirectIO** (command, ref data, ref obj);

return new DirectIOData (data, obj);

}

**/\*\*** The abstract function implements it with Service Object that

extends the Shim class.\*\*/

It is possible that the implementation of the function regarded as the object of the translation could be implemented by the Shim class. To prevent that from happening, the sealed attribute is added to prevent the override in Service Object.

For example:

public sealed override DirectIOData **DirectIO** (int command, int data, object obj)

#### Shim Code Rules for In/Out Parameters

Any OPOS parameter that is defined with an In/Out attribute in the UnifiedPOS specification is handled differently under a POS for .NET implementation. POS for .NET is expecting the data to be provided as return values. The Shim code facilitates this mapping by using the “ref” attribute to the In/Out parameter. This translation is handled automatically by the Shim code and is transparent to the calling application.

Method of Administration

The source for the Shim components is managed by the OPOS-J Committee. The Shim source code is currently available to the public from the following web site:

<http://www.monroecs.com/posfordotnet/shim.htm>.

### Shim Code File Names

The following is a list of the files that are currently available with the Shim Code. The naming convention has been chosen to provide as much intuitive device usage as possible. As new devices are released, the Shim Code will be updated to reflect the new devices. In addition, bug fixes and other support issues will be handled by OPOS-J.

|  |  |
| --- | --- |
| Shim file list |  |
| Shim class files | Description |
| CashChangerShimBasic.cs | Shim class of CashChanger |
| CashDrawerShimBasic.cs | Shim class of CashDrawer |
| CatShimBasic.cs | Shim class of Cat |
| CheckScannerShimBasic.cs | Shim class of CheckScanner |
| CoinDispenserShimBasic.cs | Shim class of CoinDispenser |
| HardTotalsShimBasic.cs | Shim class of HardTotals |
| KeylockShimBasic.cs | Shim class of Keylock |
| LineDisplayShimBasic.cs | Shim class of LineDisplay |
| MicrShimBasic.cs | Shim class of Micr |
| MsrShimBasic.cs | Shim class of Msr |
| PinPadShimBasic.cs | Shim class of PinPad |
| PointCardRWShimBasic.cs | Shim class of PointCardRW |
| PosKeyboardShimBasic.cs | Shim class of PosKeyboard |
| PosPowerShimBasic.cs | Shim class of PosPower |
| PosPrinterShimBasic.cs | Shim class of PosPrinter |
| ScaleShimBasic.cs | Shim class of Scale |
| ScannerShimBasic.cs | Shim class of Scanner |
| SmartCardRWShimBasic.cs | Shim class of SmartCardRW |
| ToneIndicatorShimBasic.cs | Shim class of ToneIndicator |
| Enumeration type definition files | Description |
| OposCash.cs | Enumeration type for CashDrawer |
| OposCat.cs | Enumeration type for Cat |
| OposChan.cs | Enumeration type for CashChanger |
| OposChk.cs | Enumeration type for CheckScanner |
| OposCoin.cs | Enumeration type for CoinDispenser |
| OposDisp.cs | Enumeration type for LineDisplay |
| OposKbd.cs | Enumeration type for PosKeyBoard |
| OposLock.cs | Enumeration type for Keylock |
| OposMicr.cs | Enumeration type for Micr |
| OposMsr.cs | Enumeration type for Msr |
| OposPcrw.cs | Enumeration type for PointCardRW |
| OposPpad.cs | Enumeration type for PinPad |
| OposPtr.cs | Enumeration type for PosPrinter |
| OposPwr.cs | Enumeration type for PosPower |
| OposScal.cs | Enumeration type for Scale |
| OposScan.cs | Enumeration type for Scanner |
| OposScrw.cs | Enumeration type for SmartCardRW |
| OposTone.cs | Enumeration type for ToneIndicator |
| OposTot.cs | Enumeration type for HardTotals |
| Project files | Description |
| AssemblyInfo.cs | Assembly information file |
| Opos.PointOfService.BasicShimServiceObjects.csproj | Project file |

### Class Diagrams

#### Interface Class

|  |
| --- |
| public abstract DirectIOData DirectIO(int command, int data, object obj) |
| public abstract void ResetStatistic(string statistic) |
| public abstract void ResetStatistics() |
| public abstract void ResetStatistics(StatisticCategories statistics) |
| public abstract void ResetStatistics(string[] statistics) |
| public abstract string RetrieveStatistic(string statistic) |
| public abstract string RetrieveStatistics() |
| public abstract string RetrieveStatistics(StatisticCategories statistics) |
| public abstract string RetrieveStatistics(string[] statistics) |
| public abstract void UpdateStatistic(string name, object value) |
| public abstract void UpdateStatistics(Statistic[] statistics) |
| public abstract void UpdateStatistics(StatisticCategories statistics, object value) |
|  |
|  |

#### Basic Class

|  |
| --- |
| public override void ResetStatistic(string statistic) |
| public override void ResetStatistics() |
| public override void ResetStatistics(StatisticCategories statistics) |
| public override void ResetStatistics(string[] statistics) |
| public override string RetrieveStatistic(string statistic) |
| public override string RetrieveStatistics() |
| public override string RetrieveStatistics(StatisticCategories statistics) |
| public override string RetrieveStatistics(string[] statistics) |
| public override void UpdateStatistic(string name, object value) |
| public override void UpdateStatistics(Statistic[] statistics) |
| public override void UpdateStatistics(StatisticCategories statistics, object value) |
|  |
|  |

#### Shim Class

|  |
| --- |
| public sealed override DirectIOData DirectIO(int command, int data, object obj) |
| public sealed override void ResetStatistic(string statistic) |
| public sealed override void ResetStatistics() |
| public sealed override void ResetStatistics(StatisticCategories statistics) |
| public sealed override void ResetStatistics(string[] statistics) |
| public sealed override string RetrieveStatistic(string statistic) |
| public sealed override string RetrieveStatistics() |
| public sealed override string RetrieveStatistics(StatisticCategories statistics) |
| public sealed override string RetrieveStatistics(string[] statistics) |
| public sealed override void UpdateStatistic(string name, object value) |
| public sealed override void UpdateStatistics(Statistic[] statistics) |
| public sealed override void UpdateStatistics(StatisticCategories statistics, object value) |
|  |
| protected abstract void DirectIO(int command, ref int data, ref object obj) |
| protected abstract void ResetStatistics(string statistics) |
| protected abstract void RetrieveStatistics(ref string statistics) |
| protected abstract void UpdateStatistics(string statistics) |
|  |
|  |

#### Service Class

|  |
| --- |
| protected override void DirectIO(int command, ref int data, ref object obj) |
| protected override void ResetStatistics(string statistics) |
| protected override void RetrieveStatistics(ref string statistics) |
| protected override void UpdateStatistics(string statistics) |
|  |
|  |