# CEN

# WORKSHOP

# AGREEMENT

## CWA 15748-62

July 2008

English version

# Extensions for Financial Services (XFS) interface specification - Release 3.10 - Part 62: Printer and Scanning Device Class Interface - Migration from Version 3.0 (CWA 14050) to Version 3.10 (this CWA) - Programmer's Reference

This CEN Workshop Agreement has been drafted and approved by a Workshop of representatives of interested parties, the constitution of which is indicated in the foreword of this Workshop Agreement.

The formal process followed by the Workshop in the development of this Workshop Agreement has been endorsed by the National Members of CEN but neither the National Members of CEN nor the CEN Management Centre can be held accountable for the technical content of this CEN Workshop Agreement or possible conflicts with standards or legislation.

This CEN Workshop Agreement can in no way be held as being an official standard developed by CEN and its Members.

This CEN Workshop Agreement is publicly available as a reference document from the CEN Members National Standard Bodies.

CEN members are the national standards bodies of Austria, Belgium, Bulgaria, Cyprus, Czech Republic, Denmark, Estonia, Finland, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, Netherlands, Norway, Poland, Portugal, Romania, Slovakia, Slovenia, Spain, Sweden, Switzerland and United Kingdom.

EUROPEAN COMMITTEE FOR STANDARDIZATION
COMITÉ EUROPÉEN DE NORMALISATION
EUROPÄISCHES KOMITEE FÜR NORMUNG

**Management Centre: rue de Stassart, 36    B-1050 Brussels**

# Table of Contents

## Foreword

This CWA is revision 3.10 of the XFS interface specification.

The CEN/ISSS XFS Workshop gathers suppliers as well as banks and other financial service companies. A list of companies participating in this Workshop and in support of this CWA is available from the CEN/ISSS Secretariat.

This CWA was formally approved by the XFS Workshop meeting on 2007-11-29. The specification is continuously reviewed and commented in the CEN/ISSS Workshop on XFS. It is therefore expected that an update of the specification will be published in due time as a CWA, superseding this revision 3.10.

The CWA is published as a multi-part document, consisting of:

Part 1: Application Programming Interface (API) - Service Provider Interface (SPI) - Programmer's Reference

Part 2: Service Classes Definition - Programmer's Reference

Part 3: Printer and Scanning Device Class Interface - Programmer's Reference

Part 4: Identification Card Device Class Interface - Programmer's Reference

Part 5: Cash Dispenser Device Class Interface - Programmer's Reference

Part 6: PIN Keypad Device Class Interface - Programmer's Reference

Part 7: Check Reader/Scanner Device Class Interface - Programmer's Reference

Part 8: Depository Device Class Interface - Programmer's Reference

Part 9: Text Terminal Unit Device Class Interface - Programmer's Reference

Part 10: Sensors and Indicators Unit Device Class Interface - Programmer's Reference

Part 11: Vendor Dependent Mode Device Class Interface - Programmer's Reference

Part 12: Camera Device Class Interface - Programmer's Reference

Part 13: Alarm Device Class Interface - Programmer's Reference

Part 14: Card Embossing Unit Device Class Interface - Programmer's Reference

Part 15: Cash-In Module Device Class Interface - Programmer's Reference

Part 16: Card Dispenser Device Class Interface - Programmer's Reference

Part 17: Barcode Reader Device Class Interface - Programmer's Reference

Part 18: Item Processing Module Device Class Interface- Programmer's Reference

Parts 19 - 28: Reserved for future use.

Parts 29 through 47 constitute an optional addendum to this CWA. They define the integration between the SNMP standard and the set of status and statistical information exported by the Service Providers.

Part 29: XFS MIB Architecture and SNMP Extensions - Programmer's Reference

Part 30: XFS MIB Device Specific Definitions - Printer Device Class

Part 31: XFS MIB Device Specific Definitions - Identification Card Device Class

Part 32: XFS MIB Device Specific Definitions - Cash Dispenser Device Class

Part 33: XFS MIB Device Specific Definitions - PIN Keypad Device Class

Part 34: XFS MIB Device Specific Definitions - Check Reader/Scanner Device Class

Part 35: XFS MIB Device Specific Definitions - Depository Device Class

Part 36: XFS MIB Device Specific Definitions - Text Terminal Unit Device Class

Part 37: XFS MIB Device Specific Definitions - Sensors and Indicators Unit Device Class

Part 38: XFS MIB Device Specific Definitions - Camera Device Class

Part 39: XFS MIB Device Specific Definitions - Alarm Device Class

Part 40: XFS MIB Device Specific Definitions - Card Embossing Unit Class

Part 41: XFS MIB Device Specific Definitions - Cash-In Module Device Class

Part 42: Reserved for future use.

Part 43: XFS MIB Device Specific Definitions - Vendor Dependent Mode Device Class

Part 44: XFS MIB Application Management

Part 45: XFS MIB Device Specific Definitions - Card Dispenser Device Class

Part 46: XFS MIB Device Specific Definitions - Barcode Reader Device Class

Part 47: XFS MIB Device Specific Definitions - Item Processing Module Device Class

Parts 48 - 60 are reserved for future use.

Part 61: Application Programming Interface (API) - Service Provider Interface (SPI) - Migration from Version 3.0 (CWA 14050) to Version 3.10 (this CWA) - Programmer's Reference

Part 62: Printer Device Class Interface - Migration from Version 3.0 (CWA 14050) to Version 3.10 (this CWA) - Programmer's Reference

Part 63: Identification Card Device Class Interface - Migration from Version 3.02 (CWA 14050) to Version 3.10 (this CWA) - Programmer's Reference

Part 64: Cash Dispenser Device Class Interface - Migration from Version 3.0 (CWA 14050) to Version 3.10 (this CWA) - Programmer's Reference

Part 65: PIN Keypad Device Class Interface - Migration from Version 3.03 (CWA 14050) to Version 3.10 (this CWA) - Programmer's Reference

Part 66: Check Reader/Scanner Device Class Interface - Migration from Version 3.0 (CWA 14050) to Version 3.10 (this CWA) - Programmer's Reference

Part 67: Depository Device Class Interface - Migration from Version 3.0 (CWA 14050) to Version 3.10 (this CWA) - Programmer's Reference

Part 68: Text Terminal Unit Device Class Interface - Migration from Version 3.0 (CWA 14050) to Version 3.10 (this CWA) - Programmer's Reference

Part 69: Sensors and Indicators Unit Device Class Interface - Migration from Version 3.01 (CWA 14050) to Version 3.10 (this CWA) - Programmer's Reference

Part 70: Vendor Dependent Mode Device Class Interface - Migration from Version 3.0 (CWA 14050) to Version 3.10 (this CWA) - Programmer's Reference

Part 71: Camera Device Class Interface - Migration from Version 3.0 (CWA 14050) to Version 3.10 (this CWA) - Programmer's Reference

Part 72: Alarm Device Class Interface - Migration from Version 3.0 (CWA 14050) to Version 3.10 (this CWA) - Programmer's Reference

Part 73: Card Embossing Unit Device Class Interface - Migration from Version 3.0 (CWA 14050) to Version 3.10 (this CWA) - Programmer's Reference

Part 74: Cash-In Module Device Class Interface - Migration from Version 3.02 (CWA 14050) to Version 3.10 (this CWA) - Programmer's Reference

In addition to these Programmer's Reference specifications, the reader of this CWA is also referred to a complementary document, called Release Notes. The Release Notes contain clarifications and explanations on the CWA specifications, which are not requiring functional changes. The current version of the Release Notes is available online from http://www.cen.eu/isss/Workshop/XFS.

The information in this document represents the Workshop's current views on the issues discussed as of the date of publication. It is furnished for informational purposes only and is subject to change without notice. CEN/ISSS makes no warranty, express or implied, with respect to this document.

This CEN Workshop Agreement is publicly available as a reference document from the National Members of CEN : AENOR, AFNOR, ASRO, BDS, BSI, CSNI, CYS, DIN, DS, ELOT, EVS, IBN, IPQ, IST, LVS, LST, MSA, MSZT, NEN, NSAI, ON, PKN, SEE, SIS, SIST, SFS, SN, SNV, SUTN and UNI.

Comments or suggestions from the users of the CEN Workshop Agreement are welcome and should be addressed to the CEN Management Centre.

# 1. Migration Information

XFS 3.10 has been designed to minimize backwards compatibility issues. This document highlights the changes made to the PTR device class between version 3.0 and 3.10, by highlighting the additions and deletions to the text.

## 2. Banking Printers

This specification describes the functionality of the services provided by banking printers and scanning devices under XFS, focusing on the following areas:

- application programming for printing

- print document definition

- integration with the Windows architecture

- scanning images for devices such as check scanners

These descriptions include definitions of the service-specific commands that can be issued, using the **WFSAsyncExecute**, **WFSExecute, WFSGetInfo** and **WFSAsyncGetInfo** functions.

The requirements for printing in banking applications are significantly different from those of the conventional PC environment, and the XFS support delivers the foundation for financial application printing, including:

- **Controlled access to shared printers**

  The banking printers can be shared between workstations and the XFS layer provides the ability for the application to manage ownership of a print device. This allows an application to identify the operator granted control of the printer, and to ensure that a teller printing multiple documents is not interrupted by work for other applications.

- **Application controlled printing**

  In the banking environment, it is necessary for the application to receive positive feedback on the availability of print devices, and the success or failure of individual print operations. The XFS printer support provides a standard mechanism for application retrieval of this status information.

- **Management of printing peripherals**

  Distributed banking networks require the ability to track the availability and failure of printing peripherals on a branch and system-wide basis. Through the XFS **WFSRegister** function monitoring programs can collect error alerts from the banking printers.

- **Vendor independent API and document definition**

  All of the XFS peripheral implementations are designed around a standardized family of APIs to allow application code portability across vendor hardware platforms. With printers, it is also recognized that banks invest a significant amount of resource in the authoring of print documents. The XFS printer service class is implemented around a forms model which also standardizes the basic document definition. This extends the investment protection provided by XFS compliant systems to include this additional part of the application development.

- **Windows printing integration**

  It is possible for a banking printer to offer printing capabilities that can be accessed by non-banking specific applications, such as general office productivity packages. This would not, for example, be true for a receipt printer, but it could be the case for a device with document printing capabilities. A vendor may choose an XFS implementation that allows both types of applications (XFS and Windows applications using the Windows printing subsystem) to share the printing devices. The vendor should specify any impact this approach has on XFS subsystem operation, such as error reporting.

Full implementation of the above features depends on the individual vendor-supplied Service Providers. This specification outlines the functionality and requirements for applications using the XFS printer and scanning services, and for the development of those services.

# 3. Banking Printer Types

The XFS printer service defines and supports five types of banking printers through a common interface:

- **Receipt Printer**

  The receipt printer is used to print cut sheet documents. It may or may not require insert or eject operations, and often includes an operator identification device, e.g. Teller A and Teller B lights, for shared operation.

- **Journal Printer**

  The journal is a continuous form device used to record a hardcopy audit trail of transactions, and for certain report printing requirements.

- **Passbook Printer**

  The passbook device is physically and functionally the most complex printer. The XFS definition supports automatic positioning of the book, as well as read/write capability for an optional integrated magnetic stripe. The implementation also manages the book geometry - i.e. the margins and centerfolds - presenting the simplest possible application interface while delivering the full range of functionality.

  Some passbook devices also support the dispensing of new passbooks from up to four passbook paper sources (upper, aux, aux2, lower). Some passbook devices may also be able to place a full passbook in a parking station, print the new passbook and return both to the customer. Passbooks can only be dispensed or moved from the parking station if there is no other media in the print position or in the entry/exit slot.

- **Document Printer**

  Document printing is similar to receipt printing - a set of fields are positioned on one or more inserted sheets of paper - but the focus is on full-size forms. It should be noted that the XFS environment supports the printing of text and graphic fields from the application. The electronic printing of the form image (the template portion of the form which is usually pre-printed with dot-matrix style printers) may also be printed by the application.

- **Scanner Printer**

  The scanner printer is a device incorporating both the capabilities to scan inserted documents and optionally to print on them. These devices may have more than one area where documents may be retained.

Additional hardware components, like scanners, stripe readers, OCR readers, and stamps, normally attached directly to the printer are also controlled through this interface. Additionally the Printer and Scanning class interface can also be used for devices that are capable of scanning without necessarily printing. This includes devices such as Check Scanners.

The specification refers to the terms paper and media. When the term paper is used this refers to paper that is situated in a paper supply attached to the device. The term media is used for media that is inserted by the customer (e.g. check and other material that is scanned) or that is issued to the customer (e.g. a receipt or statement). Receipt, document printers and also passbook printers with white passbook dispensing capability have both. As soon as the paper gets printed it becomes media. Scanners only have media. The term media does not apply to journal printers. When paper is in the print position it is classified as media, on some printers that maintain paper under the print head there will always be both media and paper.

# 4. Forms Model

The XFS printing class functionality is based on a "forms" model for printing. Banking documents are represented as a series of text and/or graphic fields output from the application, and positioned on the document by the XFS printing system.

The form is an object which includes the positioning and presentation information for each of the fields in the document. The application selects a form, and supplies only the field data and the control parameters to fully define the print document.

The form objects are owned and managed by the XFS printing service. To optimize maintainability of the system, the application can query the service for the list of fields required to print a given form. Through this mechanism, it is not necessary to duplicate the field contents of forms in application authoring data. The figure below outlines the printing process from the application's view.
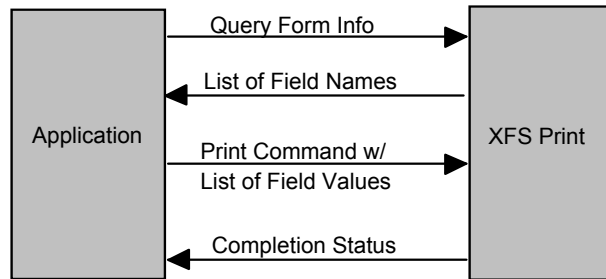


The XFS implementation recognizes that the form object must be supported by job-specific data to fully address printing requirements. As an example, a form defining a passbook print line will need to have its origin defined externally in order to be reused for different passbook lines. These job specific parameters are supplied on the call to the **WFSExecute**: WFS_CMD_PTR_PRINT_FORM command.

In some cases, the application wants to print a block of data without considering it as a series of separate fields. One example is a line of journal data, fully formatted by the application. This can be handled by defining a one field form, or by use of the **WFSExecute**: WFS_CMD_PTR_RAW_DATA command.

The document definition under XFS printing is standardized to provide portability across vendor implementations. The standard has been defined at the source language level for the document definition, allowing vendor differences at the runtime level to manage implementation specific dependencies, providing several areas where vendors can provide value-added extensions. As an example, a vendor providing a graphical form definition tool can produce the field definition object format directly. The XFS requirements for portability are:

- A vendor must be able to export print format in the standardized field definition source format for portability to other systems.

- A vendor must be able to import document formats produced on other systems in the standardized field definition source format.

- A vendor can extend the field definition source language, but any verbs included in the standard must be implemented strictly as defined by the standard. Import and export facilities must be tolerant of source language extensions, reporting but ignoring the exceptions.

The document definition also recognizes that unique hardware restrictions may require tuning of field positioning from one vendor's platform to another. To enhance portability, the XFS document format has specifically been defined to allow a single reference adjustment for all fields to avoid forcing the customer to reposition each field.

# 5.  References

1. XFS Application Programming Interface (API)/Service Provider Interface ( SPI), Programmer's Reference
Revision 3.10

# 6. Command Overview

The basic operation of the print devices is managed using the **WFSGetInfo/WFSAsyncGetInfo** and **WFSExecute/WFSAsyncExecute** functions, with two primary commands:

### WFS_INF_PTR_QUERY_FORM

This command retrieves the form header information, and the list of fields. It is performed using **WFSGetInfo**, which means that it can be performed even when the service is locked by another user.

### WFS_CMD_PTR_PRINT_FORM

This command is performed using **WFSExecute**, and includes as parameter data the name of the form to select and the required field data values.

This approach combines in the most efficient manner the four logical steps required to print a form:

- Selecting a document form object.

- Querying the service for the list of fields.

- Supplying the data for each field.

- Issuing the print command.

By using a **WFSGetInfo** command for retrieval of the list of field names, rather than **WFSExecute** (which is blocked when the service is locked by another application), it is possible for an application to assemble the required set of fields for a form before locking the service. This minimizes the time that each application request ties up the service. Using **WFSGetInfo**, it is also possible to query the attributes of a particular field. This command is generally not required for most applications.

The combination of form selection, field value presentation, and the print action into an atomic command - the **WFSExecute**: WFS_CMD_PTR_PRINT_FORM command - makes it possible to express a complete print operation with one API call. This implementation allows an application to perform a print operation without locking and subsequently unlocking the service (although locking may still be desirable for other reasons). To do multiple print operations without allowing other applications to intersperse their print requests, it is still necessary to use the lock functions. Where these multiple print functions represent a series of passbook lines (using the INDEX capability in the field definition), the **WFSExecute**: WFS_CMD_PTR_PRINT_FORM command provides support for management of the print line number. Note that if a form contains a tabular field (i.e. one with a non-zero INDEX value), and data is not supplied for some of the lines in the "table", then those lines are left blank.

For printers with the capability to read from a passbook (OCR, MICR and/or magnetic stripe), the data is read with the **WFSExecute**: WFS_CMD_PTR_READ_FORM command. The data is written using the **WFSExecute**: WFS_CMD_PTR_PRINT_FORM command. Since these devices are usable only for passbook operations, they are not defined as separate logical devices.

Finally, the **WFSExecute**: WFS_CMD_PTR_PRINT_RAW_FILE command can be used to print a file that contains a complete print job in the native printer language. This file will have been created through the Windows GDI.

# 7. Info Commands

## 7.1 WFS_INF_PTR_STATUS

**Description**    This command is used to request status information for the device.

**Input Param**    None.

**Output Param**  LPWFSPTRSTATUS lpStatus;

```
typedef struct _wfs_ptr_status
    {
    WORD                fwDevice;
    WORD                fwMedia;
    WORD                fwPaper[WFS_PTR_SUPPLYSIZE];
    WORD                fwToner;
    WORD                fwInk;
    WORD                fwLamp;
    LPWFSPTRRETRACTBINS *lppRetractBins;
    USHORT              usMediaOnStacker;
    LPSTR               lpszExtra;
    DWORD               dwGuidLights[WFS_PTR_GUIDLIGHTS_SIZE];
    WORD                wDevicePosition;
    USHORT              usPowerSaveRecoveryTime;
    } WFSPTRSTATUS, *LPWFSPTRSTATUS;
```

*fwDevice*
Specifies the state of the print device as one of the following flags:

| Value | Meaning |
|---|---|
| WFS_PTR_DEVONLINE | The device is online (i.e. powered on and operable). |
| WFS_PTR_DEVOFFLINE | The device is offline (e.g. the operator has taken the device offline by turning a switch or pulling out the device). |
| WFS_PTR_DEVPOWEROFF | The device is powered off or physically not connected. |
| WFS_PTR_DEVNODEVICE | There is no device intended to be there; e.g. this type of self service machine does not contain such a device or it is internally not configured. |
| WFS_PTR_DEVHWERROR | The device is inoperable due to a hardware error. |
| WFS_PTR_DEVUSERERROR | The device is present but a person is preventing proper device operation. |
| WFS_PTR_DEVBUSY | The device is busy and unable to process an execute command at this time. |
| WFS_PTR_DEVFRAUDATTEMPT | The device is present but has detected a fraud attempt. |

*fwMedia*
Specifies the state of the print media (i.e. receipt, statement, passbook, etc.) as one of the following values. This field does not apply to journal printers:

| Value | Meaning |
|---|---|
| WFS_PTR_MEDIAPRESENT | Media is in the print position, on the stacker or on the transport (i.e. a passbook in the parking station is not considered to be present). On devices with continuous paper supplies, this value is set when paper is under the print head. On devices with individual sheet supplies, this value is set when paper is successfully inserted/loaded. |
| WFS_PTR_MEDIANOTPRESENT | Media is not in the print position or on the stacker. |

| | |
|---|---|
| WFS_PTR_MEDIAJAMMED | Media is jammed in the device. |
| WFS_PTR_MEDIANOTSUPP | The capability to report the state of the print media is not supported by the device. |
| WFS_PTR_MEDIAUNKNOWN | The state of the print media cannot be determined with the device in its current state. |
| WFS_PTR_MEDIAENTERING | Media is at the entry/exit slot of the device. |
| WFS_PTR_MEDIARETRACTED | Media was retracted during the reset operation. |

*fwPaper [...]*
Specifies the state of the paper supplies. A number of paper supplies are defined below. Vendor specific paper supplies are defined starting from the end of the array. The maximum paper index is WFS_PTR_SUPPLYMAX.

*fwPaper [WFS_PTR_SUPPLYUPPER]*
Specifies the state of the only paper supply or the upper paper supply, if more than one, as one of the following values:

| Value | Meaning |
|---|---|
| WFS_PTR_PAPERFULL | The paper supply is full. |
| WFS_PTR_PAPERLOW | The paper supply is low. |
| WFS_PTR_PAPEROUT | The paper supply is empty. |
| WFS_PTR_PAPERNOTSUPP | Capability not supported by device. |
| WFS_PTR_PAPERUNKNOWN | Status cannot be determined with device in its current state. |
| WFS_PTR_PAPERJAMMED | The paper supply is jammed. |

*fwPaper [WFS_PTR_SUPPLYLOWER]*
Specifies the state of the lower paper supply as one of the following values:

| Value | Meaning |
|---|---|
| WFS_PTR_PAPERFULL | The paper supply is full. |
| WFS_PTR_PAPERLOW | The paper supply is low. |
| WFS_PTR_PAPEROUT | The paper supply is empty. |
| WFS_PTR_PAPERNOTSUPP | Capability not supported by device. |
| WFS_PTR_PAPERUNKNOWN | Status cannot be determined with device in its current state. |
| WFS_PTR_PAPERJAMMED | The paper supply is jammed. |

*fwPaper [WFS_PTR_SUPPLYEXTERNAL]*
Specifies the state of the external paper supply as one of the following values:

| Value | Meaning |
|---|---|
| WFS_PTR_PAPERFULL | The paper supply is full. |
| WFS_PTR_PAPERLOW | The paper supply is low. |
| WFS_PTR_PAPEROUT | The paper supply is empty. |
| WFS_PTR_PAPERNOTSUPP | Capability not supported by device. |
| WFS_PTR_PAPERUNKNOWN | Status cannot be determined with device in its current state. |
| WFS_PTR_PAPERJAMMED | The paper supply is jammed. |

*fwPaper [WFS_PTR_SUPPLYAUX]*
Specifies the state of the auxiliary paper supply as one of the following values:

| Value | Meaning |
|---|---|
| WFS_PTR_PAPERFULL | The paper supply is full. |
| WFS_PTR_PAPERLOW | The paper supply is low. |
| WFS_PTR_PAPEROUT | The paper supply is empty. |
| WFS_PTR_PAPERNOTSUPP | Capability not supported by device. |
| WFS_PTR_PAPERUNKNOWN | Status cannot be determined with device in its current state. |
| WFS_PTR_PAPERJAMMED | The paper supply is jammed. |

*fwPaper [WFS_PTR_SUPPLYAUX2]*
Specifies the state of the second auxiliary paper supply as one of the following values:

| Value | Meaning |
|---|---|
| WFS_PTR_PAPERFULL | The paper supply is full. |
| WFS_PTR_PAPERLOW | The paper supply is low. |
| WFS_PTR_PAPEROUT | The paper supply is empty. |
| WFS_PTR_PAPERNOTSUPP | Capability not supported by device. |
| WFS_PTR_PAPERUNKNOWN | Status cannot be determined with device in its current state. |
| WFS_PTR_PAPERJAMMED | The paper supply is jammed. |

*fwPaper [WFS_PTR_SUPPLYPARK]*
Specifies the state of the parking station as one of the following values:

| Value | Meaning |
|---|---|
| WFS_PTR_PAPERFULL | The parking station is busy. |
| WFS_PTR_PAPEROUT | The parking station is free. |
| WFS_PTR_PAPERNOTSUPP | Capability not supported by device. |
| WFS_PTR_PAPERUNKNOWN | Status cannot be determined with device in its current state. |
| WFS_PTR_PAPERJAMMED | The parking station is jammed. |

*fwToner*
Specifies the state of the toner or ink supply or the state of the ribbon as one of the following values:

| Value | Meaning |
|---|---|
| WFS_PTR_TONERFULL | The toner or ink supply is full or the ribbon is OK. |
| WFS_PTR_TONERLOW | The toner or ink supply is low or the print contrast with a ribbon is weak. |
| WFS_PTR_TONEROUT | The toner or ink supply is empty or the print contrast with a ribbon is not sufficient any more. |
| WFS_PTR_TONERNOTSUPP | Capability not supported by device. |
| WFS_PTR_TONERUNKNOWN | Status of toner or ink supply or the ribbon cannot be determined with device in its current state. |

*fwInk*
Specifies the status of the stamping ink in the printer as one of the following values:

| Value | Meaning |
|---|---|
| WFS_PTR_INKFULL | Ink supply in device is full. |
| WFS_PTR_INKLOW | Ink supply in device is low. |
| WFS_PTR_INKOUT | Ink supply in device is empty. |
| WFS_PTR_INKNOTSUPP | Capability not supported by device. |
| WFS_PTR_INKUNKNOWN | Status of the stamping ink supply cannot be determined with device in its current state. |

*fwLamp*
Specifies the status of the printer imaging lamp as one of the following values:

| Value | Meaning |
|---|---|
| WFS_PTR_LAMPOK | The lamp is OK. |
| WFS_PTR_LAMPFADING | The lamp should be changed. |
| WFS_PTR_LAMPINOP | The lamp is inoperative. |
| WFS_PTR_LAMPNOTSUPP | Capability not supported by device. |
| WFS_PTR_LAMPUNKNOWN | Status of the imaging lamp cannot be determined with device in its current state. |

*lppRetractBins*
Pointer to a NULL terminated array of pointers to WFSPTRRETRACTBINS structures (one for each supported bin). The first pointer holds the structure for bin one, the second for bin two and so on. A NULL pointer is returned if no retract bin is supported.

```
typedef struct _wfs_ptr_retract_bins
    {
    WORD            wRetractBin;
    USHORT          usRetractCount;
    } WFSPTRRETRACTBINS, *LPWFSPTRRETRACTBINS;
```

*wRetractBin*
Specifies the state of the printer retract bin as one of the following values:

| Value | Meaning |
|---|---|
| WFS_PTR_RETRACTBINOK | The retract bin of the printer is in a healthy state. |
| WFS_PTR_RETRACTBINFULL | The retract bin of the printer is full. |
| WFS_PTR_RETRACTUNKNOWN | Status cannot be determined with device in its current state. |
| WFS_PTR_RETRACTBINHIGH | The retract bin of the printer is nearly full. |
| WFS_PTR_RETRACTBINMISSING | The retract bin is missing. |

*usRetractCount*
The number of media retracted to this bin. This value is persistent; it may be reset to zero by the WFS_CMD_PTR_RESET_COUNT command.

*usMediaOnStacker*
The number of media on stacker; applicable only to printers with stacking capability.

*lpszExtra*
Pointer to a list of vendor-specific, or any other extended, information. The information is returned as a series of *"key=value"* strings so that it is easily extensible by Service Providers. Each string is null-terminated, with the final string terminating with two null characters. An empty list may be indicated by either a NULL pointer or a pointer to two consecutive null characters.

*dwGuidLights [...]*
Specifies the state of the guidance light indicators. A number of guidance light types are defined below. Vendor specific guidance lights are defined starting from the end of the array. The maximum guidance light index is WFS_PTR_GUIDLIGHTS_MAX.

Specifies the state of the guidance light indicator as WFS_PTR_GUIDANCE_NOT_AVAILABLE, WFS_PTR_GUIDANCE_OFF or a combination of the following flags consisting of one type B, and optionally one type C.

| Value | Meaning | Type |
|---|---|---|
| WFS_PTR_GUIDANCE_NOT_AVAILABLE | The status is not available. | A |
| WFS_PTR_GUIDANCE_OFF | The light is turned off. | A |
| WFS_PTR_GUIDANCE_SLOW_FLASH | The light is blinking slowly. | B |
| WFS_PTR_GUIDANCE_MEDIUM_FLASH | The light is blinking medium frequency. | B |
| WFS_PTR_GUIDANCE_QUICK_FLASH | The light is blinking quickly. | B |
| WFS_PTR_GUIDANCE_CONTINUOUS | The light is turned on continuous (steady). | B |
| WFS_PTR_GUIDANCE_RED | The light is red. | C |
| WFS_PTR_GUIDANCE_GREEN | The light is green. | C |
| WFS_PTR_GUIDANCE_YELLOW | The light is yellow. | C |
| WFS_PTR_GUIDANCE_BLUE | The light is blue. | C |
| WFS_PTR_GUIDANCE_CYAN | The light is cyan. | C |
| WFS_PTR_GUIDANCE_MAGENTA | The light is magenta. | C |
| WFS_PTR_GUIDANCE_WHITE | The light is white. | C |

*dwGuidLights [WFS_PTR_ GUIDANCE_PRINTER]*
Specifies the state of the guidance light indicator on the printer unit.

*wDevicePosition*
Specifies the device position. The device position value is independent of the *fwDevice* value, e.g. when the device position is reported as WFS_PTR_DEVICENOTINPOSITION, *fwDevice* can have any of the values defined above (including WFS_PTR_DEVONLINE or WFS_PTR_DEVOFFLINE). If the device is not in its normal operating position (i.e. WFS_PTR_DEVICEINPOSITION) then media may not be presented through the normal customer interface. This value is one of the following values:

| Value | Meaning |
| --- | --- |
| WFS_PTR_DEVICEINPOSITION | The device is in its normal operating position, or is fixed in place and cannot be moved. |
| WFS_PTR_DEVICENOTINPOSITION | The device has been removed from its normal operating position. |
| WFS_PTR_DEVICEPOSUNKNOWN | Due to a hardware error or other condition, the position of the device cannot be determined. |
| WFS_PTR_DEVICEPOSNOTSUPP | The physical device does not have the capability of detecting the position. |

*usPowerSaveRecoveryTime*
Specifies the actual number of seconds required by the device to resume its normal operational state from the current power saving mode. This value is zero if either the power saving mode has not been activated or no power save control is supported.

**Error Codes**  Only the generic error codes defined in [Ref. 1] can be generated by this command.

**Comments**  Applications which require or expect specific information to be present in the *lpszExtra* parameter may not be device or vendor-independent.

In the case where communications with the device has been lost, the *fwDevice* field will report WFS_PTR_DEVPOWEROFF when the device has been removed or WFS_PTR_DEVHWERROR if the communications are unexpectedly lost. All other fields should contain a value based on the following rules and priority:

1. Report the value as unknown.

2. Report the value as a general h/w error.

3. Report the value as the last known value.

## 7.2 WFS_INF_PTR_CAPABILITIES

**Description**    This command is used to request device capability information.

**Input Param**    None.

**Output Param**   LPWFSPTRCAPS lpCaps;

```
typedef struct _wfs_ptr_caps
    {
    WORD                wClass;
    WORD                fwType;
    BOOL                bCompound;
    WORD                wResolution;
    WORD                fwReadForm;
    WORD                fwWriteForm;
    WORD                fwExtents;
    WORD                fwControl;
    USHORT              usMaxMediaOnStacker;
    BOOL                bAcceptMedia;
    BOOL                bMultiPage;
    WORD                fwPaperSources;
    BOOL                bMediaTaken;
    USHORT              usRetractBins;
    LPUSHORT            lpusMaxRetract;
    WORD                fwImageType;
    WORD                fwFrontImageColorFormat;
    WORD                fwBackImageColorFormat;
    WORD                fwCodelineFormat;
    WORD                fwImageSource;
    WORD                fwCharSupport;
    BOOL                bDispensePaper;
    LPSTR               lpszExtra;
    DWORD               dwGuidLights[WFS_PTR_GUIDLIGHTS_SIZE];
    LPSTR               lpszWindowsPrinter;
    BOOL                bMediaPresented;
    USHORT              usAutoRetractPeriod;
    BOOL                bRetractToTransport;
    BOOL                bPowerSaveControl;
    } WFSPTRCAPS, *LPWFSPTRCAPS;
```

*wClass*
Specifies the logical service class as WFS_SERVICE_CLASS_PTR.

*fwType*
Specifies the type(s) of the physical device driven by the logical service, as a combination of the
following flags:

| Value | Meaning |
|---|---|
| WFS_PTR_TYPERECEIPT | Device is a receipt printer. |
| WFS_PTR_TYPEPASSBOOK | Device is a passbook printer. |
| WFS_PTR_TYPEJOURNAL | Device is a journal printer. |
| WFS_PTR_TYPEDOCUMENT | Device is a document printer. |
| WFS_PTR_TYPESCANNER | Device is a scanner with printing capabilities. |

*bCompound*
Specifies whether the logical device is part of a compound physical device.

*wResolution*
Specifies at which resolution(s) the physical device can print. Used by the application to select the
level of print quality desired (e.g. as in Word for Windows); does not imply any absolute level of
resolution, only relative. Specified as a combination of the following flags:

| Value | Meaning |
|---|---|
| WFS_PTR_RESLOW | Can print with low resolution. |
| WFS_PTR_RESMED | Can print with medium resolution. |
| WFS_PTR_RESHIGH | Can print with high resolution. |
| WFS_PTR_RESVERYHIGH | Can print with very high resolution. |

*fwReadForm*

Specifies whether the device can read data from media, as a combination of the following flags (zero if none of the choices is supported):

| Value | Meaning |
|---|---|
| WFS_PTR_READOCR | Device has OCR capability. |
| WFS_PTR_READMICR | Device has MICR capability. |
| WFS_PTR_READMSF | Device has MSF capability. |
| WFS_PTR_READBARCODE | Device has Barcode capability. |
| WFS_PTR_READPAGEMARK | Device has Page Mark capability. |
| WFS_PTR_READIMAGE | Device has imaging capability. |
| WFS_PTR_READEMPTYLINE | Device has capability to detect empty print lines for passbook printing. |

*fwWriteForm*

Specifies whether the device can write data to the media, as a combination of the following flags (zero if none of the choices is supported):

| Value | Meaning |
|---|---|
| WFS_PTR_WRITETEXT | Device has Text capability. |
| WFS_PTR_WRITEGRAPHICS | Device has Graphics capability. |
| WFS_PTR_WRITEOCR | Device has OCR capability. |
| WFS_PTR_WRITEMICR | Device has MICR capability. |
| WFS_PTR_WRITEMSF | Device has MSF capability. |
| WFS_PTR_WRITEBARCODE | Device has Barcode capability. |
| WFS_PTR_WRITESTAMP | Device has stamping capability. |

*fwExtents*

Specifies whether the device is able to measure the inserted media, as a combination of the following flags (zero if none of the choices is supported):

| Value | Meaning |
|---|---|
| WFS_PTR_EXTHORIZONTAL | Device has horizontal size detection capability. |
| WFS_PTR_EXTVERTICAL | Device has vertical size detection capability. |

*fwControl*

Specifies the manner in which media can be controlled, as a combination of the following flags (zero if none of the choices is supported):

| Value | Meaning |
|---|---|
| WFS_PTR_CTRLEJECT | Device can eject media. |
| WFS_PTR_CTRLPERFORATE | Device can perforate media. |
| WFS_PTR_CTRLCUT | Device can cut media. |
| WFS_PTR_CTRLSKIP | Device can skip to mark. |
| WFS_PTR_CTRLFLUSH | Device can be sent data that is buffered internally, and flushed to the printer on request. |
| WFS_PTR_CTRLRETRACT | Device can retract media under application control. |
| WFS_PTR_CTRLSTACK | Device can stack media items before ejecting as a bundle. |
| WFS_PTR_CTRLPARTIALCUT | Device can partially cut the media. |
| WFS_PTR_CTRLALARM | Device can ring a bell, beep or otherwise sound an audible alarm. |
| WFS_PTR_CTRLATPFORWARD | Capability to turn one page forward. |
| WFS_PTR_CTRLATPBACKWARD | Capability to turn one page backward. |
| WFS_PTR_CTRLTURNMEDIA | Device can turn inserted media. |
| WFS_PTR_CTRLSTAMP | Device can stamp on media. |
| WFS_PTR_CTRLPARK | Device can park a document into the parking station. |
| WFS_PTR_CTRLEXPEL | Device can expel media out of the exit slot. |
| WFS_PTR_CTRLEJECTTOTRANSPORT | Device can move media to a position on the transport just behind the exit slot. |

*usMaxMediaOnStacker*
Specifies the maximum number of media items that the stacker can hold (zero if not available).

*bAcceptMedia*
Specifies whether the device is able to accept media while no execute command is running that is waiting explicitly for media to be inserted. Its value is either TRUE or FALSE.

*bMultiPage*
Specifies whether the device is able to support multiple page print jobs. Its value is either TRUE or FALSE.

*fwPaperSources*
Specifies the Paper sources available for this printer as a combination of the following flags:

| Value | Meaning |
| --- | --- |
| WFS_PTR_PAPERUPPER | Indicates an upper paper source is available; devices with only one paper supply must indicate WFS_PTR_PAPERUPPER as being available. |
| WFS_PTR_PAPERLOWER | Indicates a lower paper source is available. |
| WFS_PTR_PAPEREXTERNAL | Indicates an external paper source (such as envelope tray or single sheet feed) is available. |
| WFS_PTR_PAPERAUX | An auxiliary paper source is available. |
| WFS_PTR_PAPERAUX2 | A second auxiliary paper source is available. |
| WFS_PTR_PAPERPARK | A parking station is available. |

*bMediaTaken*
Specifies whether the device is able to detect when the media is taken from the exit slot. If FALSE, the WFS_SRVE_PTR_MEDIATAKEN event is not fired. Its value is either TRUE or FALSE.

*usRetractBins*
Specifies the number of retract bins (zero if not supported).

*lpusMaxRetract*
Pointer to an array of the length *usRetractBins* with the maximum number of media items that each retract bin can hold (one count for each supported bin, starting from 0 for bin number one to *usRetractBins*-1 for bin number *usRetractBins*). NULL pointer if the device has no retract bin.

*fwImageType*
Specifies the image format supported by this device, as a combination of following flags (zero if not supported):

| Value | Meaning |
| --- | --- |
| WFS_PTR_IMAGETIF | The device can return scanned images in TIFF 6.0 format. |
| WFS_PTR_IMAGEWMF | The device can return scanned images in WMF (Windows Metafile) format. |
| WFS_PTR_IMAGEBMP | The device can return scanned images in Windows BMP format. |
| WFS_PTR_IMAGEJPG | The device can return scanned images in JPG format. |

*fwFrontImageColorFormat*
Specifies the front image color formats supported by this device, as a combination of following flags (zero if not supported):

| Value | Meaning |
| --- | --- |
| WFS_PTR_IMAGECOLORBINARY | The device can return scanned images in binary (image contains two colors, usually the colors back and white). |
| WFS_PTR_IMAGECOLORGRAYSCALE | The device can return scanned images in gray scale (image contains multiple gray colors). |

| | |
|---|---|
| WFS_PTR_IMAGECOLORFULL | The device can return scanned images in full color (image contains colors like red, green, blue etc.). |

*fwBackImageColorFormat*
Specifies the back image color formats supported by this device, as a combination of following flags (zero if not supported):

| Value | Meaning |
|---|---|
| WFS_PTR_IMAGECOLORBINARY | The device can return scanned images in binary (image contains two colors, usually the colors back and white). |
| WFS_PTR_IMAGECOLORGRAYSCALE | The device can return scanned images in gray scale (image contains multiple gray colors). |
| WFS_PTR_IMAGECOLORFULL | The device can return scanned images in full color (image contains colors like red, green, blue etc.). |

*fwCodelineFormat*
Specifies the code line (MICR data) formats supported by this device, as a combination of following flags (zero if not supported):

| Value | Meaning |
|---|---|
| WFS_PTR_CODELINECMC7 | The device can read CMC7 code lines. |
| WFS_PTR_CODELINEE13B | The device can read E13B code lines. |
| WFS_PTR_CODELINEOCR | The device can read code lines using Optical Character Recognition. |

*fwImageSource*
Specifies the source for the read image command supported by this device, as a combination of the following flags (zero if not supported):

| Value | Meaning |
|---|---|
| WFS_PTR_IMAGEFRONT | The device can scan the front image of the document. |
| WFS_PTR_IMAGEBACK | The device can scan the back image of the document. |
| WFS_PTR_CODELINE | The device can recognize the code line. |

*fwCharSupport*
One or more flags specifying the character sets, in addition to single byte ASCII, that is supported by the Service Provider:

| Value | Meaning |
|---|---|
| WFS_PTR_ASCII | ASCII is supported for XFS forms. |
| WFS_PTR_UNICODE | UNICODE is supported for XFS forms. |

For *fwCharSupport,* a Service Provider can support ONLY ASCII forms or can support BOTH ASCII and UNICODE forms. A Service Provider can not support UNICODE forms without also supporting ASCII forms.

*bDispensePaper*
Specifies whether the device is able to dispense paper. Its value is either TRUE or FALSE.

*lpszExtra*
Pointer to a list of vendor-specific, or any other extended, information. The information is returned as a series of *"key=value"* strings so that it is easily extensible by Service Providers. Each string is null-terminated, with the final string terminating with two null characters. An empty list may be indicated by either a NULL pointer or a pointer to two consecutive null characters.

*dwGuidLights [...]*
Specifies which guidance lights are available. A number of guidance light types are defined below. Vendor specific guidance lights are defined starting from the end of the array. The maximum guidance light index is WFS_PTR_GUIDLIGHTS_MAX.

The elements of this array are specified as a combination of the following flags and indicate all of the possible flash rates (type B) and colors (type C) that the guidance light indicator is capable of handling. If the guidance light indicator only supports one color then no value of type C is returned. A value of WFS_PTR_GUIDANCE_NOT_AVAILABLE indicates that the device has no guidance light indicator or the device controls the light directly with no application control possible.

| Value | Meaning | Type |
|---|---|---|
| WFS_PTR_GUIDANCE_NOT_AVAILABLE | There is no guidance light control available at this position. | A |
| WFS_PTR_GUIDANCE_OFF | The light can be off. | B |
| WFS_PTR_GUIDANCE_SLOW_FLASH | The light can blink slowly. | B |
| WFS_PTR_GUIDANCE_MEDIUM_FLASH | The light can blink medium frequency. | B |
| WFS_PTR_GUIDANCE_QUICK_FLASH | The light can blink quickly. | B |
| WFS_PTR_GUIDANCE_CONTINUOUS | The light can be continuous (steady). | B |
| WFS_PTR_GUIDANCE_RED | The light can be red. | C |
| WFS_PTR_GUIDANCE_GREEN | The light can be green. | C |
| WFS_PTR_GUIDANCE_YELLOW | The light can be yellow. | C |
| WFS_PTR_GUIDANCE_BLUE | The light can be blue. | C |
| WFS_PTR_GUIDANCE_CYAN | The light can be cyan. | C |
| WFS_PTR_GUIDANCE_MAGENTA | The light can be magenta. | C |
| WFS_PTR_GUIDANCE_WHITE | The light can be white. | C |

*dwGuidLights [WFS_PTR_GUIDANCE_PRINTER]*
Specifies whether the guidance light indicator on the printer unit is available.

*lpszWindowsPrinter*
Specifies the name of the default logical Windows printer that is associated with this Service Provider. Applications should use this printer name to generate native printer files (i.e. .PRN) to be printed through the WFS_CMD_PTR_PRINT_RAW_FILE command. This value will be NULL if the Service Provider does not support the WFS_CMD_PTR_PRINT_RAW_FILE command.

*bMediaPresented*
Specifies whether the device is able to detect when the media is presented to the user for removal. If TRUE, the WFS_EXEE_PTR_MEDIAPRESENTED event is fired. If FALSE, the WFS_EXEE_PTR_MEDIAPRESENTED event is not fired.

*usAutoRetractPeriod*
Specifies the number of seconds before the device will automatically retract the presented media. If the command that generated the media is still active when the media is automatically retracted, the command will complete with a WFS_ERR_PTR_RETRACTED error. If the device does not retract media automatically this value will be zero.

*bRetractToTransport*
Specifies whether the device is able to retract the previously ejected media to the transport. Its value is either TRUE or FALSE.

*bPowerSaveControl*
Specifies whether power saving control is available. This can either be TRUE if available or FALSE if not available.

**Error Codes**  Only the generic error codes defined in [Ref. 1] can be generated by this command.

**Comments**  Applications which require or expect specific information to be present in the *lpszExtra* parameter may not be device or vendor-independent.

## 7.3  WFS_INF_PTR_FORM_LIST

**Description**    This command is used to retrieve the list of forms available on the device.

**Input Param**    None.

**Output Param**    LPSTR lpszFormList;

*lpszFormList*
Pointer to a list of null-terminated form names, with the final name terminating with two null characters.

**Error Codes**    Only the generic error codes defined in [Ref. 1] can be generated by this command.

**Comments**    None.

## 7.4   WFS_INF_PTR_MEDIA_LIST

**Description**   This command is used to retrieve the list of media definitions available on the device.

**Input Param**   None.

**Output Param**   LPSTR lpszMediaList;

*lpszMediaList*
Pointer to a list of null-terminated media names, with the final name terminating with two null characters.

**Error Codes**   Only the generic error codes defined in [Ref. 1] can be generated by this command.

**Comments**   None.

## 7.5  WFS_INF_PTR_QUERY_FORM

**Description**  This command is used to retrieve details of the definition of a specified form.

**Input Param**  LPSTR lpszFormName;

*lpszFormName*
Points to the null-terminated form name on which to retrieve details.

**Output Param**  LPWFSFRMHEADER lpHeader;

```
typedef struct _wfs_frm_header
     {
     LPSTR                lpszFormName;
     WORD                 wBase;
     WORD                 wUnitX;
     WORD                 wUnitY;
     WORD                 wWidth;
     WORD                 wHeight;
     WORD                 wAlignment;
     WORD                 wOrientation;
     WORD                 wOffsetX;
     WORD                 wOffsetY;
     WORD                 wVersionMajor;
     WORD                 wVersionMinor;
     LPSTR                lpszUserPrompt;
     WORD                 fwCharSupport;
     LPSTR                lpszFields;
     WORD                 wLanguageID;
     } WFSFRMHEADER, *LPWFSFRMHEADER;
```

*lpszFormName*
Specifies the null-terminated name of the form.

*wBase*
Specifies the base unit of measurement of the form and can be one of the following:

| Value | Meaning |
| --- | --- |
| WFS_FRM_INCH | The base unit is inches. |
| WFS_FRM_MM | The base unit is millimeters. |
| WFS_FRM_ROWCOLUMN | The base unit is rows and columns. |

*wUnitX*
Specifies the horizontal resolution of the base units as a fraction of the *wBase* value. For example, a value of 16 applied to the base unit WFS_FRM_INCH means that the base horizontal resolution is 1/16".

*wUnitY*
Specifies the vertical resolution of the base units as a fraction of the *wBase* value. For example, a value of 10 applied to the base unit WFS_FRM_MM means that the base vertical resolution is 0.1 mm.

*wWidth*
Specifies the width of the form in terms of the base horizontal resolution.

*wHeight*
Specifies the height of the form in terms of the base vertical resolution.

*wAlignment*
Specifies the relative alignment of the form on the media and can be one of the following values:

| Value | Meaning |
| --- | --- |
| WFS_FRM_TOPLEFT | The form is aligned relative to the top and left edges of the media. |
| WFS_FRM_TOPRIGHT | The form is aligned relative to the top and right edges of the media. |
| WFS_FRM_BOTTOMLEFT | The form is aligned relative to the bottom and left edges of the media. |

| | |
|---|---|
| WFS_FRM_BOTTOMRIGHT | The form is aligned relative to the bottom and right edges of the media. |

*wOrientation*
Specifies the orientation of the form and can be one of the following values:

| Value | Meaning |
|---|---|
| WFS_FRM_PORTRAIT | The orientation of the form is portrait. |
| WFS_FRM_LANDSCAPE | The orientation of the form is landscape. |

*wOffsetX*
Specifies the horizontal offset of the position of the top-left corner of the form, relative to the left or right edge specified by *wAlignment*. This value is specified in terms of the base horizontal resolution and is always positive.

*wOffsetY*
Specifies the vertical offset of the position of the top-left corner of the form, relative to the top or bottom edge specified by *wAlignment*. This value is specified in terms of the base vertical resolution and is always positive.

*wVersionMajor*
Specifies the major version of the form. If the version is not specified in the form, then zero is returned.

*wVersionMinor*
Specifies the minor version of the form. If the version is not specified in the form, then zero is returned.

*lpszUserPrompt*
Pointer to a null-terminated user prompt string. NULL will be returned if the form does not define a value for the user prompt.

*fwCharSupport*
A single flag specifying the Character Set in which the form is encoded:

| Value | Meaning |
|---|---|
| WFS_PTR_ASCII | ASCII is supported for XFS forms initial data values and FORMAT strings. |
| WFS_PTR_UNICODE | UNICODE is supported for XFS forms initial data values and FORMAT strings. |

*lpszFields*
Pointer to a list of null-terminated field names, with the final name terminating with two null characters.

*wLanguageID*
Specifies the language identifier for the form.

**Error Codes**     In addition to the generic error codes defined in [Ref. 1], the following error codes can be generated by this command:

| Value | Meaning |
|---|---|
| WFS_ERR_PTR_FORMNOTFOUND | The specified form cannot be found. |
| WFS_ERR_PTR_FORMINVALID | The specified form is invalid. |

**Comments**     None.

## 7.6 WFS_INF_PTR_QUERY_MEDIA

**Description**   This command is used to retrieve details of the definition of a specified media.

**Input Param**   LPSTR lpszMediaName;

*lpszMediaName*
Pointer to the null-terminated media name about which to retrieve details.

**Output Param**   LPWFSFRMMEDIA lpMedia;

```
typedef struct _wfs_frm_media
      {
      WORD                fwMediaType;
      WORD                wBase;
      WORD                wUnitX;
      WORD                wUnitY;
      WORD                wSizeWidth;
      WORD                wSizeHeight;
      WORD                wPageCount;
      WORD                wLineCount;
      WORD                wPrintAreaX;
      WORD                wPrintAreaY;
      WORD                wPrintAreaWidth;
      WORD                wPrintAreaHeight;
      WORD                wRestrictedAreaX;
      WORD                wRestrictedAreaY;
      WORD                wRestrictedAreaWidth;
      WORD                wRestrictedAreaHeight;
      WORD                wStagger;
      WORD                wFoldType;
      WORD                wPaperSources;
      } WFSFRMMEDIA, *LPWFSFRMMEDIA;
```

*fwMediaType*
Specifies the type of media as one of the following values:

| Value | Meaning |
|---|---|
| WFS_FRM_MEDIAGENERIC | The media is a generic media, i.e. a single sheet. |
| WFS_FRM_MEDIAPASSBOOK | The media is a passbook media. |
| WFS_FRM_MEDIAMULTIPART | The media is a multi part media. |

*wBase*
Specifies the base unit of measurement of the form and can be one of the following values:

| Value | Meaning |
|---|---|
| WFS_FRM_INCH | The base unit is inches. |
| WFS_FRM_MM | The base unit is millimeters. |
| WFS_FRM_ROWCOLUMN | The base unit is rows and columns. |

*wUnitX*
Specifies the horizontal resolution of the base units as a fraction of the *wBase* value. For example, a value of 16 applied to the base unit WFS_FRM_INCH means that the base horizontal resolution is 1/16".

*wUnitY*
Specifies the vertical resolution of the base units as a fraction of the *wBase* value. For example, a value of 10 applied to the base unit WFS_FRM_MM means that the base vertical resolution is 0.1 mm.

*wSizeWidth*
Specifies the width of the media in terms of the base horizontal resolution.

*wSizeHeight*
Specifies the height of the media in terms of the base vertical resolution.

*wPageCount*
Specifies the number of pages in a media of type WFS_FRM_MEDIAPASSBOOK.

*wLineCount*
Specifies the number of lines on a page for a media of type WFS_FRM_MEDIAPASSBOOK.

*wPrintAreaX*
Specifies the horizontal offset of the printable area relative to the top left corner of the media in terms of the base horizontal resolution.

*wPrintAreaY*
Specifies the vertical offset of the printable area relative to the top left corner of the media in terms of the base vertical resolution.

*wPrintAreaWidth*
Specifies the printable area width of the media in terms of the base horizontal resolution.

*wPrintAreaHeight*
Specifies the printable area height of the media in terms of the base vertical resolution.

*wRestrictedAreaX*
Specifies the horizontal offset of the restricted area relative to the top left corner of the media in terms of the base horizontal resolution.

*wRestrictedAreaY*
Specifies the vertical offset of the restricted area relative to the top left corner of the media in terms of the base vertical resolution.

*wRestrictedAreaWidth*
Specifies the restricted area width of the media in terms of the base horizontal resolution.

*wRestrictedAreaHeight*
Specifies the restricted area height of the media in terms of the base vertical resolution.

*wStagger*
Specifies the staggering from the top in terms of the base vertical resolution for a media of type WFS_FRM_MEDIAPASSBOOK.

*wFoldType*
Specified the type of fold (vertical, horizontal or none) for a media of type WFS_FRM_MEDIAPASSBOOK as one of the following values:

| Value | Meaning |
| --- | --- |
| WFS_FRM_FOLDNONE | Passbook has no fold. |
| WFS_FRM_FOLDHORIZONTAL | Passbook has a horizontal fold. |
| WFS_FRM_FOLDVERTICAL | Passbook has a vertical fold. |

*wPaperSources*
Specifies the Paper sources to use when printing forms using this media as a combination of the following flags:

| Value | Meaning |
| --- | --- |
| WFS_PTR_PAPERANY | Use any paper source. |
| WFS_PTR_PAPERUPPER | Use the only or the upper paper source. |
| WFS_PTR_PAPERLOWER | Use the lower paper source. |
| WFS_PTR_PAPEREXTERNAL | Use the external paper source. |
| WFS_PTR_PAPERAUX | Use the auxiliary paper source. |
| WFS_PTR_PAPERAUX2 | Use the second auxiliary paper source. |
| WFS_PTR_PAPERPARK | Use the parking station. |

**Error Codes**  In addition to the generic error codes defined in [Ref. 1], the following error codes can be generated by this command:

| Value | Meaning |
| --- | --- |
| WFS_ERR_PTR_MEDIANOTFOUND | The specified media definition cannot be found. |
| WFS_ERR_PTR_MEDIAINVALID | The specified media definition is invalid. |

**Comments**  None.

## 7.7 WFS_INF_PTR_QUERY_FIELD

**Description** This command is used to retrieve details of the definition of a single or all fields on a specified form.

**Input Param** LPWFSPTRQUERYFIELD lpQueryField;

```
typedef struct _wfs_ptr_query_field
    {
    LPSTR               lpszFormName;
    LPSTR               lpszFieldName;
    } WFSPTRQUERYFIELD, *LPWFSPTRQUERYFIELD;
```

*lpszFormName*
Pointer to the null-terminated form name.

*lpszFieldName*
Pointer to the null-terminated name of the field about which to retrieve details. If the value of *lpszFieldName* is NULL, then details are retrieved for all fields on the form. Depending upon whether the form is encoded in UNICODE representation either the *lpszInitialValue* or *lpszUNICODEInitialValue* output fields are used to retrieve the field Initial Value.

**Output Param** LPWFSFRMFIELD *lppFields;

*lppFields*
Pointer to a null-terminated array of pointers to WFSFRMFIELD structures:

```
typedef struct _wfs_frm_field
    {
    LPSTR               lpszFieldName;
    WORD                wIndexCount;
    WORD                fwType;
    WORD                fwClass;
    WORD                fwAccess;
    WORD                fwOverflow;
    LPSTR               lpszInitialValue;
    LPWSTR              lpszUNICODEInitialValue;
    LPSTR               lpszFormat;
    LPWSTR              lpszUNICODEFormat;
    WORD                wLanguageID;
    } WFSFRMFIELD, *LPWFSFRMFIELD;
```

*lpszFieldName*
Pointer to the null-terminated field name.

*wIndexCount*
Specifies the number of entries for an index field. A value of zero indicates that this field is not an index field. Index fields are typically used to present information in a tabular fashion.

*fwType*
Specifies the type of field and can be one of the following values:

| Value | Meaning |
|---|---|
| WFS_FRM_FIELDTEXT | The field is a text field. |
| WFS_FRM_FIELDMICR | The field is a Magnetic Ink Character Recognition field. |
| WFS_FRM_FIELDOCR | The field is an Optical Character Recognition field. |
| WFS_FRM_FIELDMSF | The field is a Magnetic Stripe Facility field. |
| WFS_FRM_FIELDBARCODE | The field is a Barcode field. |
| WFS_FRM_FIELDGRAPHIC | The field is a Graphic field. |
| WFS_FRM_FIELDPAGEMARK | The field is a Page Mark field. |

*fwClass*
Specifies the class of the field and can be one of the following values:

| Value | Meaning |
|---|---|
| WFS_FRM_CLASSSTATIC | The field data cannot be set by the application. |

| WFS_FRM_CLASSOPTIONAL | The field data can be set by the application. |
| WFS_FRM_CLASSREQUIRED | The field data must be set by the application. |

*fwAccess*
Specifies whether the field is to be used for input, output, or both and can be a combination of the following flags:

| Value | Meaning |
| --- | --- |
| WFS_FRM_ACCESSREAD | The field is used for input. |
| WFS_FRM_ACCESSWRITE | The field is used for output. |

*fwOverflow*
Specifies how an overflow of field data should be handled and can be one of the following values:

| Value | Meaning |
| --- | --- |
| WFS_FRM_OVFTERMINATE | Return an error and terminate printing of the form. |
| WFS_FRM_OVFTRUNCATE | Truncate the field data to fit in the field. |
| WFS_FRM_OVFBESTFIT | Fit the text in the field. |
| WFS_FRM_OVFOVERWRITE | Print the field data beyond the extents of the field boundary. |
| WFS_FRM_OVFWORDWRAP | If the field can hold more than one line the text is wrapped around. |

*lpszInitialValue*
The initial value of the field. When the form is printed (using WFS_CMD_PTR_PRINT_FORM), this value will be used if another value is not provided. This value can be NULL if the parameter is not specified in the field definition or the form is encoded in UNICODE.

*lpszUNICODEInitialValue*
The initial value of the field when form is encoded in UNICODE. When the form is printed (using WFS_CMD_PTR_PRINT_FORM), this value will be used if another value is not provided. This value can be NULL if the parameter is not specified in the field definition or the form is not encoded in UNICODE.

*lpszFormat*
Format string as defined in the form for this field. This value can be NULL if the parameter is not specified in the field definition or the form is encoded in UNICODE.

*lpszUNICODEFormat*
Format string as defined in the form for this field when form is encoded in UNICODE. This value can be NULL if the parameter is not specified in the field definition or the form is not encoded in UNICODE.

*wLanguageID*
Specifies the language identifier for the field.

**Error Codes**    In addition to the generic error codes defined in [Ref. 1], the following error codes can be generated by this command:

| Value | Meaning |
| --- | --- |
| WFS_ERR_PTR_FORMNOTFOUND | The specified form cannot be found. |
| WFS_ERR_PTR_FIELDNOTFOUND | The specified field cannot be found. |
| WFS_ERR_PTR_FORMINVALID | The specified form is invalid. |
| WFS_ERR_PTR_FIELDINVALID | The specified field is invalid. |

**Comments**    None.

## 7.8  WFS_INF_PTR_CODELINE_MAPPING

**Description**    This command is used to retrieve the byte code mapping for the special banking symbols defined for image processing (e.g. check processing). This mapping must be reported as there is no standard for the fonts defined below.

**Input Param**    LPWFSCODELINEMAPPING lpCodelineMapping;

```
typedef struct  wfs ptr codeline mapping
       {
       WORD                wCodelineFormat;
       } WFSPTRCODELINEMAPPING, *LPWFSPTRCODELINEMAPPING;
```

*wCodeLineFormat*
Specifies the code-line format that the mapping for the special characters is required for. This field can be one of the following values:

| Value | Meaning |
|---|---|
| WFS_PTR_CODELINECMC7 | Report the CMC7 mapping. |
| WFS_PTR_CODELINEE13B | Report the E13B mapping. |

**Output Param**   LPWFSCODELINEMAPPINGOUT lpCodelineMapping;

```
typedef struct  wfs ptr codeline mapping out
       {
       WORD                wCodelineFormat;
       LPWFSPTRXDATA       lpxCharMapping;
       } WFSPTRCODELINEMAPPINGOUT, *LPWFSPTRCODELINEMAPPINGOUT;
```

*wCodeLineFormat*
Specifies the code-line format that is being reported.

*lpxCharMapping*
Defines the mapping of the font specific symbols to byte values. These byte values are used to represent the font specific characters when the code line is read through the WFS_CMD_PTR_READ_IMAGE command. The font specific meaning of each index is defined in the following tables:

E13B

| Index | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| Symbol that byte value represents | | | | | N/A |
| Meaning | Transit | Amount | On Us | Dash | Reject / Unreadable |

CMC7

| Index | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| Symbol | | | | | | N/A |
| Meaning | S1 - Start of Bank Account | S2 - Start of the Amount field | S3 - Terminate Routing | S4 - Unused | S5 - Transit / Routing | Reject / Unreadable |

**Error Codes**    Only the generic error codes defined in [Ref. 1] can be generated by this command.

**Comments**    None.

# 8. Execute Commands

## 8.1 WFS_CMD_PTR_CONTROL_MEDIA

**Description**   This command is used to control a form drawn in by the device (e.g. after reading or in case of termination of an application request).

If an eject operation is specified, it completes when the media is moved to the exit slot. A service event is generated when the media has been taken by the user (only if field *bMediaTaken* defined in structure WFSPTRCAPS is equal to TRUE).

**Input Param**   LPDWORD lpdwMediaControl;

*lpdwMediaControl*
Pointer to a value which specifies the manner in which the media should be handled, as a combination of the following bit-flags:

| Value | Meaning |
|---|---|
| WFS_PTR_CTRLEJECT | Flush any data to the printer that has not yet been printed from previous WFS_CMD_PTR_PRINT_FORM or WFS_CMD_PTR_PRINT_RAW_FILE commands, then eject the media. |
| WFS_PTR_CTRLPERFORATE | Flush data as above, then perforate the media. |
| WFS_PTR_CTRLCUT | Flush data as above, then cut the media. For printers which have the ability to stack multiple cut sheets and deliver them as a single bundle to the customer, cut causes the media to be stacked and eject causes the bundle to be moved to the exit slot. |
| WFS_PTR_CTRLSKIP | Flush data as above, then skip the media to mark. |
| WFS_PTR_CTRLFLUSH | Flush any data to the printer that has not yet been printed from previous WFS_CMD_PTR_PRINT_FORM or WFS_CMD_PTR_PRINT_RAW_FILE commands. |
| WFS_PTR_CTRLRETRACT | Flush data as above, then retract the media to retract bin number one, for devices with more than one bin the command WFS_CMD_PTR_RETRACT_MEDIA should be used if the media should be retracted to another bin than bin number one. |
| WFS_PTR_CTRLSTACK | Flush data as above, then move the media item on the internal stacker. |
| WFS_PTR_CTRLPARTIALCUT | Flush the data as above, then partially cut the media. |
| WFS_PTR_CTRLALARM | Cause the printer to ring a bell, beep, or otherwise sound an audible alarm. |
| WFS_PTR_CTRLATPFORWARD | Flush the data as above, then turn one page forward. |
| WFS_PTR_CTRLATPBACKWARD | Flush the data as above, then turn one page backward. |
| WFS_PTR_CTRLTURNMEDIA | Flush the data as above, then turn inserted media. |
| WFS_PTR_CTRLSTAMP | Flush the data as above, then stamp on inserted media. |
| WFS_PTR_CTRLPARK | Park the media in the parking station. |
| WFS_PTR_CTRLEXPEL | Flush the data as above, then throw the media out of the exit slot. |

Page 32
CWA 15748-62:2008

WFS_PTR_CTRLEJECTTOTRANSPORT    Flush the data as above, then move the media to a position on the transport just behind the exit slot.

It is not possible to combine the flags WFS_PTR_CTRLEJECT, WFS_PTR_CTRLRETRACT, WFS_PTR_CTRLPARK and WFS_PTR_CTRLEXPEL with each other. In this case the command completes with WFS_ERR_INVALID_DATA.

An application should be aware that the sequence of the actions is not guaranteed if more than one flag is specified in this parameter.

**Output Param**  None.

**Error Codes**  In addition to the generic error codes defined in [Ref. 1], the following error codes can be generated by this command:

| Value | Meaning |
|---|---|
| WFS_ERR_PTR_NOMEDIAPRESENT | The control action could not be completed because there is no media in the device, the media is not in a position where it can be controlled, or (in the case of WFS_PTR_CTRLRETRACT) has been removed. |
| WFS_ERR_PTR_FLUSHFAIL | The device was not able to flush data. |
| WFS_ERR_PTR_RETRACTBINFULL | The retract bin is full. No more media can be retracted. The current media is still in the device. |
| WFS_ERR_PTR_STACKERFULL | The internal stacker is full. No more media can be moved to the stacker. |
| WFS_ERR_PTR_PAGETURNFAIL | The device was not able to turn the page. |
| WFS_ERR_PTR_MEDIATURNFAIL | The device was not able to turn the inserted media. |
| WFS_ERR_PTR_SHUTTERFAIL | Open or close of the shutter failed due to manipulation or hardware error. |
| WFS_ERR_PTR_MEDIAJAMMED | The media is jammed; operator intervention is required. |
| WFS_ERR_PTR_PAPERJAMMED | The paper is jammed. |
| WFS_ERR_PTR_PAPEROUT | The paper supply is empty. |
| WFS_ERR_PTR_INKOUT | No stamping possible, stamping ink supply empty. |
| WFS_ERR_PTR_TONEROUT | Toner or ink supply is empty or printing contrast with ribbon is not sufficient. |
| WFS_ERR_PTR_SEQUENCEINVALID | Programming error. Invalid command sequence (e.g. WFS_PTR_CTRLPARK and the parking station is busy). |
| WFS_ERR_PTR_MEDIARETAINED | Media has been retracted in attempts to eject it. The device is clear and can be used. |
| WFS_ERR_PTR_BLACKMARK | Black mark detection has failed, nothing has been printed. |
| WFS_ERR_PTR_MEDIARETRACTED | Presented media was automatically retracted before all wads could be presented and before the command could complete successfully. |

**Events**  In addition to the generic events defined in [Ref. 1], the following events can be generated by this command:

| Value | Meaning |
|---|---|
| WFS_USRE_PTR_RETRACTBINTHRESHOLD | The retract bin is high or full; operator intervention is required. Note that this event is sent only once, at the point at which the bin becomes high or full. It is sent with WFS_PTR_RETRACTBINHIGH or WFS_PTR_RETRACTBINFULL status. |

**Deleted:** No

**Deleted:** is present

| | |
|---|---|
| WFS_SRVE_PTR_MEDIATAKEN | The media has been taken by the user. |
| WFS_USRE_PTR_PAPERTHRESHOLD | The paper supply is low or empty; operator intervention is required. Note that this event is sent only once, at the point at which the supply becomes low or empty. It is sent with WFS_PTR_PAPERLOW or WFS_PTR_PAPEROUT status. |
| WFS_USRE_PTR_TONERTHRESHOLD | The toner or ink supply is low or empty or the printing contrast with ribbon is weak or not sufficient; operator intervention is required. Note that this event is sent only once, at the point at which the supply becomes low or empty. It is sent with WFS_PTR_TONERLOW or WFS_PTR_TONEROUT status. |
| WFS_USRE_PTR_INKTHRESHOLD | The stamping ink supply is low or empty; operator intervention is required. Note that this event is sent only once, at the point at which the supply becomes low or empty. It is sent with WFS_PTR_INKLOW or WFS_PTR_INKOUT status. |
| WFS_EXEE_PTR_MEDIAPRESENTED | Media has been presented for removal. See section 11 for further details. |
| WFS_SRVE_PTR_MEDIAAUTORETRACTED | The presented media has been automatically retracted. |

**Comments**    None.

## 8.2 WFS_CMD_PTR_PRINT_FORM

**Description** This command is used to print a form by merging the supplied variable field data with the defined form and field data specified in the form. If no media is present, the device waits for the period of time specified by the *dwTimeOut* parameter in the **WFSExecute** call for media to be inserted from the external paper source.

**Input Param** LPWFSPTRPRINTFORM lpPrintForm;

```
typedef struct _wfs_ptr_print_form
    {
    LPSTR               lpszFormName;
    LPSTR               lpszMediaName;
    WORD                wAlignment;
    WORD                wOffsetX;
    WORD                wOffsetY;
    WORD                wResolution;
    DWORD               dwMediaControl;
    LPSTR               lpszFields;
    LPWSTR              lpszUNICODEFields;
    WORD                wPaperSource;
    } WFSPTRPRINTFORM, *LPWFSPTRPRINTFORM;
```

*lpszFormName*
Pointer to the null-terminated form name.

*lpszMediaName*
Pointer to the null-terminated media name. *lpszMediaName* is NULL if no media definition applies.

*wAlignment*
Specifies the alignment of the form on the physical media, as one of the following values:

| Value | Meaning |
|---|---|
| WFS_PTR_ALNUSEFORMDEFN | Use the alignment specified in the form definition. |
| WFS_PTR_ALNTOPLEFT | Align form to top left of physical media. |
| WFS_PTR_ALNTOPRIGHT | Align form to top right of physical media. |
| WFS_PTR_ALNBOTTOMLEFT | Align form to bottom left of physical media. |
| WFS_PTR_ALNBOTTOMRIGHT | Align form to bottom right of physical media. |

*wOffsetX*
Specifies the horizontal offset of the form, relative to the horizontal alignment specified in *wAlignment*, in horizontal resolution units (from form definition); always a positive number (i.e. if aligned to the right side of the media, means offset the form to the left). A value of WFS_PTR_OFFSETUSEFORMDEFN indicates that the *xoffset* value from the form definition should be used.

*wOffsetY*
Specifies the vertical offset of the form, relative to the vertical alignment specified in *wAlignment*, in vertical resolution units (from form definition); always a positive number (i.e. if aligned to the bottom of the media, means offset the form upward). A value of WFS_PTR_OFFSETUSEFORMDEFN indicates that the *yoffset* value from the form definition should be used.

*wResolution*
Specifies the resolution in which to print the form. Possible values are:

| Value | Meaning |
|---|---|
| WFS_PTR_RESLOW | Print form with low resolution. |
| WFS_PTR_RESMED | Print form with medium resolution. |
| WFS_PTR_RESHIGH | Print form with high resolution. |
| WFS_PTR_RESVERYHIGH | Print form with very high resolution. |

*dwMediaControl*
Specifies the manner in which the media should be handled after the printing is done, as a combination of the flags described under WFS_CMD_PTR_CONTROL_MEDIA. A zero value of this parameter means to do none of these actions, as when printing multiple forms on a single page.

*lpszFields*
Pointer to a series of "<FieldName>=<FieldValue>" strings, where each string is null-terminated with the entire field string terminating with two null characters. If the field is an index field, then the syntax of the string is instead "<FieldName>[<index>]=<FieldValue>", where <index> specifies the zero-based element of the index field.

*lpszUNICODEFields*
Pointer to a series of "<FieldName>=<FieldValue>" UNICODE strings, where each string is null-terminated with the entire field string terminating with two null characters. If the field is an index field, then the syntax of the string is instead "<FieldName>[<index>]=<FieldValue>", where <index> specifies the zero-based element of the index field.

The *lpszUNICODEFields* field should only be used if the form is encoded in UNICODE representation. This can be determined with the WFS_PTR_INF_QUERY_FORM command.

*wPaperSource*
Specifies the Paper source to use when printing this form. When the value is zero, then the paper source is determined from the media definition. This parameter is ignored if there is already paper in the print position. Possible values are:

| Value | Meaning |
|---|---|
| WFS_PTR_PAPERANY | Any paper source can be used; it is determined by the service. |
| WFS_PTR_PAPERUPPER | Use the only paper source or the upper paper source, if there is more than one paper supply. |
| WFS_PTR_PAPERLOWER | Use the lower paper source. |
| WFS_PTR_PAPEREXTERNAL | Use the external paper source (such as envelope tray or single sheet feed). |
| WFS_PTR_PAPERAUX | Use the auxiliary paper source. |
| WFS_PTR_PAPERAUX2 | Use the second auxiliary paper source. |
| WFS_PTR_PAPERPARK | Use the parking station. |

**Output Param** None.

**Error Codes** In addition to the generic error codes defined in [Ref. 1], the following error codes can be generated by this command:

| Value | Meaning |
|---|---|
| WFS_ERR_PTR_FORMNOTFOUND | The specified form definition cannot be found. |
| WFS_ERR_PTR_FLUSHFAIL | The device was not able to flush data. |
| WFS_ERR_PTR_MEDIAOVERFLOW | The form overflowed the media. |
| WFS_ERR_PTR_FIELDSPECFAILURE | The syntax of the *lpszFields* member is invalid. |
| WFS_ERR_PTR_FIELDERROR | An error occurred while processing a field, causing termination of the print request. An execute event WFS_EXEE_PTR_FIELDERROR is posted with the details. |
| WFS_ERR_PTR_MEDIANOTFOUND | The specified media definition cannot be found. |
| WFS_ERR_PTR_MEDIAINVALID | The specified media definition is invalid. |
| WFS_ERR_PTR_FORMINVALID | The specified form definition is invalid. |
| WFS_ERR_PTR_MEDIASKEWED | The media skew exceeded the limit in the form definition. |
| WFS_ERR_PTR_RETRACTBINFULL | The retract bin is full. No more media can be retracted. The current media is still in the device. |

Page 36
CWA 15748-62:2008

| | |
|---|---|
| WFS_ERR_PTR_STACKERFULL | The internal stacker is full. No more media can be moved to the stacker. |
| WFS_ERR_PTR_PAGETURNFAIL | The device was not able to turn the page. |
| WFS_ERR_PTR_MEDIATURNFAIL | The device was not able to turn the inserted media. |
| WFS_ERR_PTR_SHUTTERFAIL | Open or close of the shutter failed due to manipulation or hardware error. |
| WFS_ERR_PTR_MEDIAJAMMED | The media is jammed; operator intervention is required. |
| WFS_ERR_PTR_CHARSETDATA | Character set(s) supported by Service Provider is inconsistent with use of *lpszFields* or *lpszUNICODEFields* fields. |
| WFS_ERR_PTR_PAPERJAMMED | The paper is jammed. |
| WFS_ERR_PTR_PAPEROUT | The paper supply is empty. |
| WFS_ERR_PTR_INKOUT | No stamping possible, stamping ink supply empty. |
| WFS_ERR_PTR_TONEROUT | Toner or ink supply is empty or printing contrast with ribbon is not sufficient. |
| WFS_ERR_PTR_SEQUENCEINVALID | Programming error. Invalid command sequence (e.g. *dwMediaControl* = WFS_PTR_CTRLPARK and park position is busy). |
| WFS_ERR_PTR_SOURCEINVALID | The selected paper source is not supported by the hardware. |
| WFS_ERR_PTR_MEDIARETAINED | Media has been retracted in attempts to eject it. The device is clear and can be used. |
| WFS_ERR_PTR_BLACKMARK | Black mark detection has failed, nothing has been printed. |
| WFS_ERR_PTR_MEDIASIZE | The media entered has an incorrect size and the media remains inside the device. |
| WFS_ERR_PTR_MEDIAREJECTED | The media was rejected during the insertion phase and no data has been printed. The WFS_EXEE_PTR_MEDIAREJECTED execute event is posted with the details. The device is still operational. |
| WFS_ERR_PTR_MEDIARETRACTED | Presented media was automatically retracted before all wads could be presented and before the command could complete successfully. |
| WFS_ERR_PTR_MSFERROR | An error occurred while writing the magnetic stripe data. |
| WFS_ERR_PTR_NOMSF | No magnetic stripe found; media may have been inserted or pulled through the wrong way. |

**Events**     In addition to the generic events defined in [Ref. 1], the following events can be generated by this command:

| Value | Meaning |
|---|---|
| WFS_EXEE_PTR_NOMEDIA | No media is present in the device. |
| WFS_EXEE_PTR_MEDIAINSERTED | Media has been inserted into the device. |
| WFS_EXEE_PTR_FIELDERROR | A fatal error occurred while processing a field. |
| WFS_EXEE_PTR_FIELDWARNING | A non-fatal error occurred while processing a field. |
| WFS_USRE_PTR_RETRACTBINTHRESHOLD | The retract bin is full; operator intervention is required. Note that this event is sent only once, at the point at which the bin becomes full. It is sent with WFS_PTR_RETRACTBINFULL or WFS_PTR_RETRACTBINHIGH status. |
| WFS_SRVE_PTR_MEDIATAKEN | The media has been taken by the user. |

| | |
|---|---|
| WFS_USRE_PTR_PAPERTHRESHOLD | The paper supply is low or empty; operator intervention is required. Note that this event is sent only once, at the point at which the supply becomes low or empty. It is sent with WFS_PTR_PAPERLOW or WFS_PTR_PAPEROUT status. |
| WFS_USRE_PTR_TONERTHRESHOLD | The toner or ink supply is low or empty or the printing contrast with ribbon is weak or not sufficient; operator intervention is required. Note that this event is sent only once, at the point at which the supply becomes low or empty. It is sent with WFS_PTR_TONERLOW or WFS_PTR_TONEROUT status. |
| WFS_USRE_PTR_INKTHRESHOLD | The stamping ink supply is low or empty; operator intervention is required. Note that this event is sent only once, at the point at which the supply becomes low or empty. It is sent with WFS_PTR_INKLOW or WFS_PTR_INKOUT status. |
| WFS_EXEE_PTR_MEDIAPRESENTED | Media has been presented for removal. See section 11 for further details. |
| WFS_EXEE_PTR_MEDIAREJECTED | The media has been rejected and presented back to the user. It is available at the entry/exit slot. When the media is removed, a WFS_SRVE_PTR_MEDIATAKEN event will be sent. |
| WFS_SRVE_PTR_MEDIAAUTORETRACTED | The presented media has been automatically retracted. |

**Comments**       All error codes (except WFS_ERR_PTR_NOMEDIAPRESENT) and events listed under the WFS_CMD_PTR_CONTROL_MEDIA command description can also occur on this command.

An invalid field name is treated as a WFS_EXEE_PTR_FIELDWARNING event with WFS_PTR_FIELDNOTFOUND status. A WFS_EXEE_PTR_FIELDWARNING event is returned with WFS_PTR_FIELDOVERFLOW status if the data overflows the field, and the field definition OVERFLOW value is TRUNCATE, BESTFIT, OVERWRITE or WORDWRAP. Other field-related problems generate a field error return and event.

The application will use *lpszFields* or *lpszUNICODEFields* as an input parameter, depending upon the Service Provider capabilities. Legacy (non-UNICODE aware) applications will only use the *lpszFields* field. UNICODE applications can use either the *lpszFields* or *lpszUNICODEFields* fields, provided the Service Provider is UNICODE compliant.

## 8.3 WFS_CMD_PTR_READ_FORM

**Description**   This command is used to read data from input fields on the specified form. These input fields may consist of MICR, OCR, MSF, BARCODE, or PAGEMARK input fields. These input fields may also consist of TEXT fields for purposes of detecting available passbook print lines with passbook printers supporting such capability. If no media is present, the device waits for the period of time specified by the *dwTimeOut* parameter in the **WFSExecute** call for media to be inserted.

**Input Param**   LPWFSPTRREADFORM lpReadForm;

```
typedef struct _wfs_ptr_read_form
    {
    LPSTR               lpszFormName;
    LPSTR               lpszFieldNames;
    LPSTR               lpszMediaName;
    DWORD               dwMediaControl;
    } WFSPTRREADFORM, *LPWFSPTRREADFORM;
```

*lpszFormName*
Pointer to the null-terminated name of the form.

*lpszFieldNames*
Pointer to a list of null-terminated field names from which to read input data, with the final name terminating with two null characters. If this value is NULL, then read data from all input fields on the form.

*lpszMediaName*
Pointer to the null-terminated media name. *lpszMediaName* is NULL if no media definition applies.

*dwMediaControl*
Specifies the manner in which the media should be handled after the reading was done and can be a combination of the flags described under WFS_CMD_PTR_CONTROL_MEDIA.

**Output Param**   LPWFSPTRREADFORMOUT lpReadFormOut;

```
typedef struct _wfs_ptr_read_form_out
    {
    LPSTR               lpszFields;
    LPWSTR              lpszUNICODEFields;
    } WFSPTRREADFORMOUT, *LPWFSPTRREADFORMOUT;
```

*lpszFields*
Pointer to a series of "<FieldName>=<FieldValue>" strings, where each string is null-terminated with the entire field string terminating with two null characters. If the field is an index field, then the syntax of the string is instead "<FieldName>[<index>]=<FieldValue>", where <index> specifies the zero-based element of the index field.

*lpszUNICODEFields*
Pointer to a series of "<FieldName>=<FieldValue>" UNICODE strings, where each string is null-terminated with the entire field string terminating with two null characters. If the field is an index field, then the syntax of the string is instead "<FieldName>[<index>]=<FieldValue>", where <index> specifies the zero-based element of the index field.

**Error Codes**   In addition to the generic error codes defined in [Ref. 1], the following error codes can be generated by this command:

| Value | Meaning |
|---|---|
| WFS_ERR_PTR_FORMNOTFOUND | The specified form cannot be found. |
| WFS_ERR_PTR_READNOTSUPPORTED | The device has no read capability. |
| WFS_ERR_PTR_FIELDSPECFAILURE | The syntax of the *lpszFieldNames* member is invalid. |
| WFS_ERR_PTR_FIELDERROR | An error occurred while processing a field, causing termination of the print request. An execute event WFS_EXEE_PTR_FIELDERROR is posted with the details. |

| | |
|---|---|
| WFS_ERR_PTR_MEDIANOTFOUND | The specified media definition cannot be found. |
| WFS_ERR_PTR_MEDIAINVALID | The specified media definition is invalid. |
| WFS_ERR_PTR_FORMINVALID | The specified form definition is invalid. |
| WFS_ERR_PTR_MEDIASKEWED | The media skew exceeded the limit in the form definition. |
| WFS_ERR_PTR_RETRACTBINFULL | The retract bin is full. No more media can be retracted. The current media is still in the device. |
| WFS_ERR_PTR_SHUTTERFAIL | Open or close of the shutter failed due to manipulation or hardware error. |
| WFS_ERR_PTR_MEDIAJAMMED | The media is jammed. |
| WFS_ERR_PTR_INKOUT | No stamping possible, stamping ink supply empty. |
| WFS_ERR_PTR_LAMPINOP | Imaging lamp is inoperative. |
| WFS_ERR_PTR_SEQUENCEINVALID | Programming error. Invalid command sequence (e.g. *dwMediaControl* = WFS_PTR_CTRLPARK and park position is busy). |
| WFS_ERR_PTR_MEDIASIZE | The media entered has an incorrect size. |
| WFS_ERR_PTR_MEDIAREJECTED | The media was rejected during the insertion phase. The WFS_EXEE_PTR_MEDIAREJECTED execute event is posted with the details. The device is still operational. |
| WFS_ERR_PTR_MSFERROR | The MSF read operation specified by the forms definition could not be completed successfully due to invalid magnetic stripe data. |
| WFS_ERR_PTR_NOMSF | No magnetic stripe found; media may have been inserted or pulled through the wrong way. |

**Events**  In addition to the generic events defined in [Ref. 1], the following events can be generated by this command:

| Value | Meaning |
|---|---|
| WFS_EXEE_PTR_NOMEDIA | No media is present in the device. |
| WFS_EXEE_PTR_MEDIAINSERTED | Media has been inserted into the device. |
| WFS_EXEE_PTR_FIELDERROR | A fatal error occurred while processing a field. |
| WFS_EXEE_PTR_FIELDWARNING | A non-fatal error occurred while processing a field. |
| WFS_USRE_PTR_RETRACTBINTHRESHOLD | The retract bin is full; operator intervention is required. Note that this event is sent only once, at the point at which the bin becomes full. It is sent with WFS_PTR_RETRACTBINFULL or WFS_PTR_RETRACTBINHIGH status. |
| WFS_SRVE_PTR_MEDIATAKEN | The media has been taken by the user. |
| WFS_USRE_PTR_INKTHRESHOLD | The stamping ink supply is low or empty; operator intervention is required. Note that this event is sent only once, at the point at which the supply becomes low or empty. It is sent with WFS_PTR_INKLOW or WFS_PTR_INKOUT status. |
| WFS_USRE_PTR_LAMPTHRESHOLD | The imaging lamp is fading or inoperative; operator intervention is required. Note that this event is sent only once, at the point at which the threshold is reached. It is sent with WFS_PTR_LAMPFADING or WFS_PTR_LAMPINOP status. |

**Deleted:** supply becomes low or empty

|  |  |
|---|---|
| WFS_EXEE_PTR_MEDIAREJECTED | The media has been rejected and presented back to the user. It is available at the entry/exit slot. When the media is removed, a WFS_SRVE_PTR_MEDIATAKEN event will be sent. |

**Comments**     All error codes (except WFS_ERR_PTR_NOMEDIAPRESENT) and events listed under the WFS_CMD_PTR_CONTROL_MEDIA command description can also occur on this command.

The application will use *lpszFieldNames* as an input parameter. The Service Provider will return the data in *lpszUNICODEFields* or *lpszFields* depending on the capabilities of the Service Provider and form definition.

For passbook usage of the *lpszFields* and *lpszUNICODEFields* fields the following applies:

If the media type is PASSBOOK, and the field(s) type is TEXT, and the Service Provider and the underlying passbook printer are capable of detecting available passbook print lines, then the field(s) will be returned without a value, in the format "<FieldName>" or "<FieldName>[<index>]", if the field is available for passbook printing. Field(s) unavailable for passbook printing will not be returned. The Service Provider will examine the passbook text field(s) supplied in the *lpszFieldNames* string, and with the form/fields definition and the underlying passbook printer capability determine which fields should be available for passbook printing.

To illustrate when media type is PASSBOOK, if a form named PSBKTST1 contains 24 fields, one field per line, and the field names are LINE1 through LINE24 (same order as printing), and after execution of this command *lpszFields* contains fields LINE13 through LINE24, then the first print line available for passbook printing is line 13.

To illustrate another example when media type is PASSBOOK, if a form named PSBKTST2 contains 24 fields, one field per line, and the field names are LINE1 through LINE24 (same order as printing), and after execution of this command *lpszFields* contains fields LINE13, and LINE20 through LINE24 then the first print line available for passbook printing is line 13, however lines 14-19 are not also available, so if the application is attempting to determine the first available print line after which all subsequent print lines are also available then line 20 is a better choice.

**Deleted:** or lpszUNICODEFieldNames

**Deleted:** ,

**Deleted:** up

**Deleted:** service provider

**Deleted:** . Legacy (non-UNICODE aware) applications will only use the lpszFieldNames field. UNICODE applications can use either

**Deleted:** lpszFieldNames or lpszUNICODEFieldNames fields, provided the service provider is UNICODE compliant

## 8.4 WFS_CMD_PTR_RAW_DATA

**Description**　This command is used to send raw data (a byte string of device dependent data) to the physical device.

**Input Param**　LPWFSPTRRAWDATA lpRawData;

```
typedef struct _wfs_ptr_raw_data
    {
    WORD                wInputData;
    ULONG               ulSize;
    LPBYTE              lpbData;
    } WFSPTRRAWDATA, *LPWFSPTRRAWDATA;
```

*wInputData*
Specifies that input data from the device is expected in response to sending the raw data (i.e. the data contains a command requesting data). Possible values are:

| Value | Meaning |
|---|---|
| WFS_PTR_NOINPUTDATA | No input data is expected. |
| WFS_PTR_INPUTDATA | Input data is expected. |

*ulSize*
Specifies the size of the byte string passed to the device.

*lpbData*
Points to the byte string holding the device dependent data.

**Output Param**　LPWFSPTRRAWDATAIN lpRawDataIn;

[used only if *wInputData* is set to WFS_PTR_INPUTDATA]

```
typedef struct _wfs_ptr_raw_data_in
    {
    ULONG               ulSize;
    LPBYTE              lpbData;
    } WFSPTRRAWDATAIN, *LPWFSPTRRAWDATAIN;
```

*ulSize*
Specifies the size of the byte string received from the device.

*lpbData*
Points to the byte string received from the device.

**Error Codes**　In addition to the generic error codes defined in [Ref. 1], the following error codes can be generated by this command:

| Value | Meaning |
|---|---|
| WFS_ERR_PTR_SHUTTERFAIL | Open or close of the shutter failed due to manipulation or hardware error. |
| WFS_ERR_PTR_MEDIAJAMMED | The media is jammed. |
| WFS_ERR_PTR_PAPERJAMMED | The paper is jammed. |
| WFS_ERR_PTR_PAPEROUT | The paper supply is empty. |
| WFS_ERR_PTR_TONEROUT | Toner or ink supply is empty or printing contrast with ribbon is not sufficient. |
| WFS_ERR_PTR_MEDIARETAINED | Media has been retracted in attempts to eject it. The device is clear and can be used. |
| WFS_ERR_PTR_BLACKMARK | Black mark detection has failed, nothing has been printed. |
| WFS_ERR_PTR_MEDIARETRACTED | Presented media was automatically retracted before all wads could be presented and before the command could complete successfully. |

**Events**　In addition to the generic events defined in [Ref. 1], the following events can be generated by this command:

| Value | Meaning |
|---|---|
| WFS_USRE_PTR_RETRACTBINTHRESHOLD | The retract bin is full or high; operator intervention is required. Note that this event is sent only once, at the point at which the bin becomes full or high. It is sent with WFS_PTR_RETRACTBINFULL or WFS_PTR_RETRACTBINHIGH status. |
| WFS_SRVE_PTR_MEDIATAKEN | The media has been taken by the user. |
| WFS_USRE_PTR_PAPERTHRESHOLD | The paper supply is low or empty; operator intervention is required. Note that this event is sent only once, at the point at which the supply becomes low or empty. It is sent with WFS_PTR_PAPERLOW or WFS_PTR_PAPEROUT status. |
| WFS_USRE_PTR_TONERTHRESHOLD | The toner or ink supply is low or empty or the printing contrast with ribbon is weak or not sufficient; operator intervention is required. Note that this event is sent only once, at the point at which the supply becomes low or empty. It is sent with WFS_PTR_TONERLOW or WFS_PTR_TONEROUT status. |
| WFS_EXEE_PTR_MEDIAPRESENTED | Media has been presented for removal. See section 11 for further details. |
| WFS_SRVE_PTR_MEDIAAUTORETRACTED | The presented media has been automatically retracted. |

**Comments**   Applications which send raw data to a device will typically not be device or vendor independent. Problems with the use of this command include:

1. The data sent to the device can include commands that change the state of the device in unpredictable ways (in particular, in ways that the Service Provider may not be aware of).

2. Usage of this command will not be portable.

3. This command violates the XFS forms model that is the basis of XFS printer access.

Thus usage of this command should be avoided whenever possible. If it is used, the usage should be carefully isolated from all other XFS access to the service by at least the **WFSLock** and **WFSUnlock** commands.

## 8.5 WFS_CMD_PTR_MEDIA_EXTENTS

**Description**     This command is used to get the extents of the media inserted in the physical device. The input parameter specifies the base unit and fractions in which the media extent values will be returned. If no media is present, the device waits for the period of time specified by the *dwTimeOut* parameter in the **WFSExecute** call for media to be inserted.

**Input Param**     LPWFSPTRMEDIAUNIT lpMediaUnit;

```
typedef struct _wfs_ptr_media_unit
    {
    WORD                wBase;
    WORD                wUnitX;
    WORD                wUnitY;
    } WFSPTRMEDIAUNIT, *LPWFSPTRMEDIAUNIT;
```

*wBase*
Specifies the base unit of measurement of the media and can be one of the following values:

| Value | Meaning |
|---|---|
| WFS_FRM_INCH | The base unit is inches. |
| WFS_FRM_MM | The base unit is millimeters. |
| WFS_FRM_ROWCOLUMN | The base unit is rows and columns. |

*wUnitX*
Specifies the horizontal resolution of the base units as a fraction of the *wBase* value. For example, a value of 16 applied to the base unit WFS_FRM_INCH means that the base horizontal resolution is 1/16".

*wUnitY*
Specifies the vertical resolution of the base units as a fraction of the *wBase* value. For example, a value of 10 applied to the base unit WFS_FRM_MM means that the base vertical resolution is 0.1 mm.

**Output Param**     LPWFSPTRMEDIAEXT lpMediaExt;

```
typedef struct _wfs_ptr_media_ext
    {
    ULONG               ulSizeX;
    ULONG               ulSizeY;
    } WFSPTRMEDIAEXT, *LPWFSPTRMEDIAEXT;
```

*ulSizeX*
Specifies the width of the media in terms of the base horizontal resolution.

*ulSizeY*
Specifies the height of the media in terms of the base vertical resolution.

**Error Codes**     In addition to the generic error codes defined in [Ref. 1], the following error codes can be generated by this command:

| Value | Meaning |
|---|---|
| WFS_ERR_PTR_EXTENTNOTSUPPORTED | The device cannot report extent(s). |
| WFS_ERR_PTR_MEDIAJAMMED | The media is jammed. |
| WFS_ERR_PTR_LAMPINOP | Imaging lamp is inoperative. |
| WFS_ERR_PTR_MEDIASIZE | The media entered has an incorrect size and the media remains inside the device. |
| WFS_ERR_PTR_MEDIAREJECTED | The media was rejected during the insertion phase. The WFS_EXEE_PTR_MEDIAREJECTED execute event is posted with the details. The device is still operational. |

**Events**     In addition to the generic events defined in [Ref. 1], the following events can be generated by this command:

| Value | Meaning |
|---|---|
| WFS_EXEE_PTR_NOMEDIA | No media is present in the device. |
| WFS_EXEE_PTR_MEDIAINSERTED | Media has been inserted into the device. |

| | |
|---|---|
| WFS_EXEE_PTR_MEDIAREJECTED | The media has been rejected and presented back to the user. It is available at the entry/exit slot. When the media is removed, a WFS_SRVE_PTR_MEDIATAKEN event will be sent. |
| WFS_SRVE_PTR_MEDIATAKEN | The media has been taken by the user. |

**Comments**     None.

## 8.6  WFS_CMD_PTR_RESET_COUNT

**Description**     This function resets the present value for number of media items retracted to zero. The function is possible only for printers with retract capability.

The number of media items retracted is controlled by the service and can be requested before resetting via the info command WFS_INF_PTR_STATUS.

**Input Param**     LPUSHORT lpusBinNumber;

*lpusBinNumber*
Pointer to the number of the retract bin for which the retract count should be reset to zero. This number has to be between one and the number of bins on the device. If this pointer is NULL all bins will be set to zero.

**Output Param**    None.

**Error Codes**     Only the generic error codes defined in [Ref. 1] can be generated by this command.

**Events**          In addition to the generic events defined in [Ref. 1], the following events can be generated by this command:

| Value | Meaning |
|---|---|
| WFS_USRE_PTR_RETRACTBINTHRESHOLD | |
| | The status of the retract bin has changed from high or full to a good state. The event is sent with WFS_PTR_RETRACTBINOK status. |

**Comments**        None.

## 8.7 WFS_CMD_PTR_READ_IMAGE

**Description**     This function returns image data from the current media. If no media is present, the device waits
for the period of time specified by the *dwTimeOut* parameter in the **WFSExecute** call for media
to be inserted.

**Input Param**     LPWFSPTRIMAGEREQUEST lpImageRequest;

```
typedef struct _wfs_ptr_image_request
        {
        WORD                wFrontImageType;
        WORD                wBackImageType;
        WORD                wFrontImageColorFormat;
        WORD                wBackImageColorFormat;
        WORD                wCodelineFormat;
        WORD                fwImageSource;
        LPSTR               lpszFrontImageFile;
        LPSTR               lpszBackImageFile;
        } WFSPTRIMAGEREQUEST, *LPWFSPTRIMAGEREQUEST;
```

*wFrontImageType*
Specifies the format of the front image returned by this command as one of the following flags
(zero if source not selected):

| Value | Meaning |
| --- | --- |
| WFS_PTR_IMAGETIF | The returned image is in TIF 6.0 format. |
| WFS_PTR_IMAGEWMF | The returned image is in WMF (Windows Metafile) format. |
| WFS_PTR_IMAGEBMP | The returned image is in BMP format. |
| WFS_PTR_IMAGEJPG | The returned image is in JPG format. |

*wBackImageType*
Specifies the format of the back image returned by this command as one of the following flags
(zero if source not selected):

| Value | Meaning |
| --- | --- |
| WFS_PTR_IMAGETIF | The returned image is in TIF 6.0 format. |
| WFS_PTR_IMAGEWMF | The returned image is in WMF (Windows Metafile) format. |
| WFS_PTR_IMAGEBMP | The returned image is in BMP format. |
| WFS_PTR_IMAGEJPG | The returned image is in JPG format. |

*wFrontImageColorFormat*
Specifies the color format of the requested front image as one of the following flags (zero if
source not selected):

| Value | Meaning |
| --- | --- |
| WFS_PTR_IMAGECOLORBINARY | The scanned images has to be returned in binary (image contains two colors, usually the colors back and white). |
| WFS_PTR_IMAGECOLORGRAYSCALE | The scanned images has to be returned in gray scale (image contains multiple gray colors). |
| WFS_PTR_IMAGECOLORFULL | The scanned images has to be returned in full color (image contains colors like red, green, blue etc.). |

*wBackImageColorFormat*
Specifies the color format of the requested back image as one of the following flags (zero if
source not selected):

| Value | Meaning |
| --- | --- |
| WFS_PTR_IMAGECOLORBINARY | The scanned images has to be returned in binary (image contains two colors, usually the colors back and white). |

| | |
|---|---|
| WFS_PTR_IMAGECOLORGRAYSCALE | The scanned images has to be returned in gray scale (image contains multiple gray colors). |
| WFS_PTR_IMAGECOLORFULL | The scanned images has to be returned in full color (image contains colors like red, green, blue etc.). |

*wCodelineFormat*
Specifies the code line (MICR data) format, as one of the following flags (Zero if source not selected):

| Value | Meaning |
|---|---|
| WFS_PTR_CODELINECMC7 | Read CMC7 code line. |
| WFS_PTR_CODELINEE13B | Read E13B code line. |
| WFS_PTR_CODELINEOCR | Read code line using OCR. |

*fwImageSource*
Specifies the source as a combination of the following flags:

| Value | Meaning |
|---|---|
| WFS_PTR_IMAGEFRONT | The front image of the document is requested. |
| WFS_PTR_IMAGEBACK | The back image of the document is requested. |
| WFS_PTR_CODELINE | The code line of the document is requested. |

*lpszFrontImageFile*
File specifying where to store the front image, e.g. "C:\Temp\FrontImage.bmp". If a NULL pointer is supplied then the front image data will be returned in the output parameter.

To reduce the size of data sent between the Application and the Service Provider it is recommended to make use of this parameter.

*lpszBackImageFile*
File specifying where to store the back image, e.g. "C:\Temp\BackImage.bmp". If a NULL pointer is supplied then the back image data will be returned in the output structure.

To reduce the size of data sent between the application and the Service Provider it is recommended to make use of this parameter.

**Output Param**  LPWFSPTRIMAGE *lppImage;

Pointer to a NULL-terminated array of pointers to WFSPTRIMAGE structures, one array element for each image source requested.

```
typedef struct _wfs_ptr_image
    {
    WORD                wImageSource;
    WORD                wStatus;
    ULONG               ulDataLength;
    LPBYTE              lpbData;
    } WFSPTRIMAGE, *LPWFSPTRIMAGE;
```

*wImageSource*
Specifies the source of the data returned by this command as one of the following flags:

| Value | Meaning |
|---|---|
| WFS_PTR_IMAGEFRONT | The front image of the document is requested. |
| WFS_PTR_IMAGEBACK | The back image of the document is requested. |
| WFS_PTR_CODELINE | The code line of the document is requested. |

*wStatus*
Status of reading the image data. Possible values are:

| Value | Meaning |
|---|---|
| WFS_PTR_DATAOK | The data is OK. |
| WFS_PTR_DATASRCNOTSUPP | The data source to read from is not supported by the Service Provider. |

| | |
|---|---|
| WFS_PTR_DATASRCMISSING | The data source to read from is missing, e.g. the Service Provider is unable to get the code line. |

*ulDataLength*
Count of bytes of the following *lpbData*. Zero if the image source is WFS_PTR_IMAGEFRONT or WFS_PTR_IMAGEBACK and the image data has been stored to the hard disk (file name provided).

*lpbData*
Points to the image or codeline data. NULL pointer if the image source is WFS_PTR_IMAGEFRONT or WFS_PTR_IMAGEBACK and the image data has been stored to the hard disk (file name provided).

If the image source is WFS_PTR_CODELINE, *lpbData* contains characters in the ASCII range. If the code line was read using the OCR-A font then the ASCII codes will conform to Figure E1 in ANSI X3.17-1981. If the code line was read using the OCR-B font then the ASCII codes will conform to Figure C2 in ANSI X3.49-1975. In both these cases unrecognized characters will be reported as the REJECT code, 0x1A. The E13B and CMC7 fonts use the ASCII equivalents for the standard characters and use the byte values as reported by the WFS_INF_PTR_CODELINE_MAPPING command for the symbols that are unique to MICR fonts.

**Error Codes** In addition to the generic error codes defined in [Ref. 1], the following error codes can be generated by this command:

| Value | Meaning |
|---|---|
| WFS_ERR_PTR_SHUTTERFAIL | Open or close of the shutter failed due to manipulation or hardware error. |
| WFS_ERR_PTR_MEDIAJAMMED | The media is jammed; operator intervention is required. |
| WFS_ERR_PTR_FILE_IO_ERROR | Directory does not exist or a File IO error occurred while storing the image to the hard disk. |
| WFS_ERR_PTR_LAMPINOP | Imaging lamp is inoperative. |
| WFS_ERR_PTR_MEDIASIZE | The media entered has an incorrect size and the media remains inside the device. |
| WFS_ERR_PTR_MEDIAREJECTED | The media was rejected during the insertion phase. The WFS_EXEE_PTR_MEDIAREJECTED execute event is posted with the details. The device is still operational. |

**Events** In addition to the generic events defined in [Ref. 1], the following events can be generated by this command:

| Value | Meaning |
|---|---|
| WFS_EXEE_PTR_NOMEDIA | No media is present in the device. |
| WFS_EXEE_PTR_MEDIAINSERTED | Media has been inserted into the device. |
| WFS_SRVE_PTR_MEDIATAKEN | The media has been taken by the user. |
| WFS_USRE_PTR_LAMPTHRESHOLD | The imaging lamp is fading or inoperative; operator intervention is required. Note that this event is sent only once, at the point at which the threshold is reached. It is sent with WFS_PTR_LAMPFADING or WFS_PTR_LAMPINOP status. |
| WFS_EXEE_PTR_MEDIAREJECTED | The media has been rejected and presented back to the user. It is available at the entry/exit slot. When the media is removed, a WFS_SRVE_PTR_MEDIATAKEN event will be sent. |
| WFS_SRVE_PTR_MEDIAAUTORETRACTED | The presented media has been automatically retracted. |

**Comments** If the returned image data is in Windows bitmap format (BMP) and a file path for storing the

image is not supplied, then the first byte of data will be the start of the Bitmap Info Header (this bitmap format is known as DIB, Device Independent Bitmap). The Bitmap File Info Header, which is only present in file versions of bitmaps, will NOT be returned. If the returned image data is in bitmap format (BMP) and a file path for storing the image is supplied, then the first byte of data in the stored file will be the Bitmap File Info Header.

## 8.8 WFS_CMD_PTR_RESET

**Description**    This command is used by the application to perform a hardware reset which will attempt to return the PTR device to a known good state. This command does not over-ride a lock obtained on another application or service handle.

    The device will attempt to retract or eject any items found anywhere within the device. This may not always be possible because of hardware problems. The WFS_SRVE_PTR_MEDIADETECTED event will inform the application where items were actually moved to.

**Input Param**    LPWFSPTRRESET lpReset;

    Specifies where media should be moved to that is found in the device. If the application does not wish to specify a position it can set *lpReset* to NULL. In this case the Service Provider will determine where to move any items found.

```
typedef struct _wfs_ptr_reset
    {
    DWORD              dwMediaControl;
    USHORT             usRetractBinNumber;
    } WFSPTRRESET, *LPWFSPTRRESET;
```

*dwMediaControl*
Specifies the manner in which the media should be handled, as one of the following bit-flags:

| Value | Meaning |
|---|---|
| WFS_PTR_CTRLEJECT | Eject the media. |
| WFS_PTR_CTRLRETRACT | Retract the media to retract bin as specified in *usRetractBinNumber*. |
| WFS_PTR_CTRLEXPEL | Throw the media out of the exit slot. |

> **Deleted:** number one

*usRetractBinNumber*
Number of the retract bin the media is retracted to. This number has to be between one and the number of bins supported by this device. It is only relevant if *dwMediaControl* equals WFS_PTR_CTRLRETRACT.

**Output Param**    None.

**Error Codes**    In addition to the generic error codes defined in [Ref. 1], the following error codes can be generated by this command:

| Value | Meaning |
|---|---|
| WFS_ERR_PTR_SHUTTERFAIL | Open or close of the shutter failed due to manipulation or hardware error. |
| WFS_ERR_PTR_RETRACTBINFULL | The retract bin is full; no more media can be retracted. The current media is still in the device. |
| WFS_ERR_PTR_MEDIAJAMMED | The media is jammed; operator intervention is required. |
| WFS_ERR_PTR_PAPERJAMMED | The paper is jammed. |

**Events**    In addition to the generic events defined in [Ref. 1], the following events can be generated by this command:

| Value | Meaning |
|---|---|
| WFS_SRVE_PTR_MEDIADETECTED | A media is detected in the device during a reset operation. |
| WFS_USRE_PTR_RETRACTBINTHRESHOLD | The retract bin is full or high; operator intervention is required. Note that this event is sent only once, at the point at which the bin becomes full or high. It is sent with WFS_PTR_RETRACTBINFULL or WFS_PTR_RETRACTBINHIGH status. |

WFS_SRVE_PTR_MEDIAAUTORETRACTED

The presented media has been automatically retracted.

WFS_EXEE_PTR_MEDIAPRESENTED   Media has been presented for removal. See section 11 for further details.

**Comments**   This command is used by an application control program to cause a device to reset itself to a known good condition.

## 8.9  WFS_CMD_PTR_RETRACT_MEDIA

**Description**  The media is removed from its present position (media inserted into device, media entering, unknown position) and stored in one of the retract bins. An event is sent if the storage capacity of the specified retract bin is reached. If the bin is already full and the command cannot be executed, an error is returned and the media remains in its present position.

**Input Param**  LPUSHORT lpusBinNumber;

*lpusBinNumber*
Pointer to the number of one of the retract bins. This number has to be between one and the number of bins supported by this device. If *lpusBinNumber* points to a zero value, the media will be retracted to the transport. After it has been retracted to the transport, in a subsequent operation the media can be ejected again, or retracted to one of the retract bins.

**Output Param**  LPUSHORT lpusBinNumber;

*lpusBinNumber*
Pointer to the number of the retract bin where the media has actually been deposited.

**Error Codes**  In addition to the generic error codes defined in [Ref. 1], the following error codes can be generated by this command:

| Value | Meaning |
|---|---|
| WFS_ERR_PTR_NOMEDIAPRESENT | No media present on retract. Either there was no media present (in a position to be retracted from) when the command was called or the media was removed during the retract. |
| WFS_ERR_PTR_RETRACTBINFULL | The retract bin is full; no more media can be retracted. The current media is still in the device. |
| WFS_ERR_PTR_MEDIAJAMMED | The media is jammed; operator intervention is required. |

**Events**  In addition to the generic events defined in [Ref. 1], the following events can be generated by this command:

| Value | Meaning |
|---|---|
| WFS_USRE_PTR_RETRACTBINTHRESHOLD | |
| | The retract bin is full; operator intervention is required. Note that this event is sent only once, at the point at which the bin becomes full. It is sent with WFS_PTR_RETRACTBINFULL or WFS_PTR_RETRACTBINHIGH status. |

**Comments**  If a retract request is received for a device with no retract capability, the WFS_ERR_UNSUPP_COMMAND error is returned.

## 8.10 WFS_CMD_PTR_DISPENSE_PAPER

**Description**   This command is used to move paper (which can also be a new passbook) from a paper source into the print position.

**Input Param**   LPWORD lpwPaperSource;

*lpwPaperSource*
Pointer to the paper source to dispense from. Possible values are:

| Value | Meaning |
|---|---|
| WFS_PTR_PAPERANY | Any paper source can be used; it is determined by the service. |
| WFS_PTR_PAPERUPPER | Use the only paper source or the upper paper source, if there is more than one paper supply. |
| WFS_PTR_PAPERLOWER | Use the lower paper source. |
| WFS_PTR_PAPEREXTERNAL | Use the external paper. |
| WFS_PTR_PAPERAUX | Use the auxiliary paper source. |
| WFS_PTR_PAPERAUX2 | Use the second auxiliary paper source. |
| WFS_PTR_PAPERPARK | Use the parking station paper source. |

**Output Param**   None.

**Error Codes**   In addition to the generic error codes defined in [Ref. 1], the following error codes can be generated by this command:

| Value | Meaning |
|---|---|
| WFS_ERR_PTR_PAPERJAMMED | The paper is jammed. |
| WFS_ERR_PTR_PAPEROUT | The paper supply is empty. |
| WFS_ERR_PTR_SEQUENCEINVALID | Programming error. Invalid command sequence (e.g. there is already media in the print position). |
| WFS_ERR_PTR_SOURCEINVALID | The selected paper source is not supported by the hardware. |
| WFS_ERR_PTR_MEDIARETRACTED | Presented media was automatically retracted before all wads could be presented and before the command could complete successfully. |

**Events**   In addition to the generic events defined in [Ref. 1], the following events can be generated by this command:

| Value | Meaning |
|---|---|
| WFS_USRE_PTR_PAPERTHRESHOLD | The paper supply is low or empty; operator intervention is required. Note that this event is sent only once, at the point at which the supply becomes low or empty. It is sent with WFS_PTR_PAPERLOW or WFS_PTR_PAPEROUT status. |
| WFS_EXEE_PTR_MEDIAPRESENTED | Media has been presented for removal. See section 11 for further details. |
| WFS_SRVE_PTR_MEDIAAUTORETRACTED | The presented media has been automatically retracted. |

**Comments**   None.

## 8.11 WFS_CMD_PTR_SET_GUIDANCE_LIGHT

**Description** This command is used to set the status of the PTR guidance lights. This includes defining the flash rate and the color. When an application tries to use a color that is not supported then the Service Provider will return the generic error WFS_ERR_UNSUPP_DATA.

**Input Param** LPWFSPTRSETGUIDLIGHT lpSetGuidLight;

```
typedef struct  wfs_ptr_set_guidlight
      {
      WORD                  wGuidLight;
      DWORD                 dwCommand;
      } WFSPTRSETGUIDLIGHT, *LPWFSPTRSETGUIDLIGHT;
```

*wGuidLight*
Specifies the index of the guidance light to set as one of the values defined within the capabilities section.

*dwCommand*
Specifies the state of the guidance light indicator as WFS_PTR_GUIDANCE_OFF or a combination of the following flags consisting of one type B, and optionally one type C. If no value of type C is specified then the default color is used. The Service Provider determines which color is used as the default color.

| Value | Meaning | Type |
|---|---|---|
| WFS_PTR_GUIDANCE_OFF | The light indicator is turned off. | A |
| WFS_PTR_GUIDANCE_SLOW_FLASH | The light indicator is set to flash slowly. | B |
| WFS_PTR_GUIDANCE_MEDIUM_FLASH | The light indicator is set to flash medium frequency. | B |
| WFS_PTR_GUIDANCE_QUICK_FLASH | The light indicator is set to flash quickly. | B |
| WFS_PTR_GUIDANCE_CONTINUOUS | The light indicator is turned on continuously (steady). | B |
| WFS_PTR_GUIDANCE_RED | The light indicator color is set to red. | C |
| WFS_PTR_GUIDANCE_GREEN | The light indicator color is set to green. | C |
| WFS_PTR_GUIDANCE_YELLOW | The light indicator color is set to yellow. | C |
| WFS_PTR_GUIDANCE_BLUE | The light indicator color is set to blue. | C |
| WFS_PTR_GUIDANCE_CYAN | The light indicator color is set to cyan. | C |
| WFS_PTR_GUIDANCE_MAGENTA | The light indicator color is set to magenta. | C |
| WFS_PTR_GUIDANCE_WHITE | The light indicator color is set to white. | C |

**Output Param** None.

**Error Codes** In addition to the generic error codes defined in [Ref. 1], the following error codes can be generated by this command:

| Value | Meaning |
|---|---|
| WFS_ERR_PTR_INVALID_PORT | An attempt to set a guidance light to a new value was invalid because the guidance light does not exist. |

**Events** Only the generic events defined in [Ref. 1] can be generated by this command.

**Comments** Guidance light support was added into the PTR primarily to support guidance lights for workstations where more than one instance of a PTR is present. The original SIU guidance light mechanism was not able to manage guidance lights for workstations with multiple PTRs. This command can also be used to set the status of the PTR guidance lights when only one instance of a PTR is present.

## 8.12 WFS_CMD_PTR_PRINT_RAW_FILE

**Description**
This command is used to print a file that contains a complete print job in the native printer language. This file will have been created through the Windows GDI print sub-system. The contents of this file are printer specific.

If no media is present, the device waits for the period of time specified by the *dwTimeOut* parameter in the **WFSExecute** call for media to be inserted from the external paper source.

This command must not complete until all pages have been presented to the customer.

**Input Param**
LPWFSPTRPRINTRAWFILE lpPrintRawFile;

```
typedef struct _wfs_ptr_print_raw_file
    {
    LPSTR              lpszFileName;
    DWORD              dwMediaControl;
    DWORD              dwPaperSource;
    } WFSPTRPRINTRAWFILE, *LPWFSPTRPRINTRAWFILE;
```

*lpszFileName*
Pointer to the null-terminated file name. This is the full path and file name of the file to be printed.

*dwMediaControl*
Specifies the manner in which the media should be handled after each page is printed, as a combination of the flags described under WFS_CMD_PTR_CONTROL_MEDIA. A zero value of this parameter means to do none of these actions, as when printing multiple pages on a single media item.

*dwPaperSource*
Specifies the paper source to use when printing. When the value is zero, the Service Provider will determine the paper source that will be used. This parameter is ignored if there is already paper in the print position. Possible values are:

| Value | Meaning |
|---|---|
| WFS_PTR_PAPERANY | Any paper source can be used; it is determined by the service. |
| WFS_PTR_PAPERUPPER | Use the only paper source or the upper paper source, if there is more than one paper supply. |
| WFS_PTR_PAPERLOWER | Use the lower paper source. |
| WFS_PTR_PAPEREXTERNAL | Use the external paper source (such as envelope tray or single sheet feed). |
| WFS_PTR_PAPERAUX | Use the auxiliary paper source. |
| WFS_PTR_PAPERAUX2 | Use the second auxiliary paper source. |
| WFS_PTR_PAPERPARK | Use the parking station. |

**Output Param** None.

**Error Codes**
In addition to the generic error codes defined in [Ref. 1], the following error codes can be generated by this command:

| Value | Meaning |
|---|---|
| WFS_ERR_PTR_FILENOTFOUND | The specified file cannot be found. |
| WFS_ERR_PTR_SHUTTERFAIL | Open or close of the shutter failed due to manipulation or hardware error. |
| WFS_ERR_PTR_MEDIAJAMMED | The media is jammed; operator intervention is required. |
| WFS_ERR_PTR_PAPERJAMMED | The paper is jammed. |
| WFS_ERR_PTR_PAPEROUT | The paper supply is empty. |
| WFS_ERR_PTR_TONEROUT | Toner or ink supply is empty or printing contrast with ribbon is not sufficient. |
| WFS_ERR_PTR_FILE_IO_ERROR | Directory does not exist or a File IO error occurred while processing the file. |
| WFS_ERR_PTR_NOMEDIAPRESENT | No media is present in the device. |
| WFS_ERR_PTR_FLUSHFAIL | The device was not able to flush data. |

| WFS_ERR_PTR_RETRACTBINFULL | The retract bin is full. No more media can be retracted. The current media is still in the device. |
|---|---|
| WFS_ERR_PTR_STACKERFULL | The internal stacker is full. No more media can be moved to the stacker. |
| WFS_ERR_PTR_PAGETURNFAIL | The device was not able to turn the page. |
| WFS_ERR_PTR_MEDIATURNFAIL | The device was not able to turn the inserted media. |
| WFS_ERR_PTR_INKOUT | No stamping possible, stamping ink supply empty. |
| WFS_ERR_PTR_SEQUENCEINVALID | Programming error. Invalid command sequence (e.g. WFS_PTR_CTRLPARK and the parking station is busy). |
| WFS_ERR_PTR_MEDIAOVERFLOW | The print request has overflowed the print media (e.g. print on a single sheet printer exceeded one page). |
| WFS_ERR_PTR_MEDIARETAINED | Media has been retracted in attempts to eject it. The device is clear and can be used. |
| WFS_ERR_PTR_BLACKMARK | Black mark detection has failed, nothing has been printed. |
| WFS_ERR_PTR_SOURCEINVALID | The selected paper source is not supported by the hardware. |
| WFS_ERR_PTR_MEDIAREJECTED | The media was rejected during the insertion phase and no data has been printed. The WFS_EXEE_PTR_MEDIAREJECTED execute event is posted with the details. The device is still operational. |
| WFS_ERR_PTR_MEDIARETRACTED | Presented media was automatically retracted before all wads could be presented and before the command could complete successfully. |

**Events**     In addition to the generic events defined in [Ref. 1], the following events can be generated by this command:

| Value | Meaning |
|---|---|
| WFS_EXEE_PTR_NOMEDIA | No media is present in the device. |
| WFS_EXEE_PTR_MEDIAINSERTED | Media has been inserted into the device. |
| WFS_EXEE_PTR_MEDIAPRESENTED | Media has been presented for removal. See section 11 for further details. |
| WFS_SRVE_PTR_MEDIATAKEN | The media has been taken by the user. |
| WFS_USRE_PTR_PAPERTHRESHOLD | The paper supply is low or empty; operator intervention is required. Note that this event is sent only once, at the point at which the supply becomes low or empty. It is sent with WFS_PTR_PAPERLOW or WFS_PTR_PAPEROUT status. |
| WFS_USRE_PTR_TONERTHRESHOLD | The toner or ink supply is low or empty or the printing contrast with ribbon is weak or not sufficient; operator intervention is required. Note that this event is sent only once, at the point at which the supply becomes low or empty. It is sent with WFS_PTR_TONERLOW or WFS_PTR_TONEROUT status. |
| WFS_USRE_PTR_RETRACTBINTHRESHOLD | The retract bin is high or full; operator intervention is required. Note that this event is sent only once, at the point at which the bin becomes high or full. It is sent with WFS_PTR_RETRACTBINHIGH or WFS_PTR_RETRACTBINFULL status. |

| | |
|---|---|
| WFS_USRE_PTR_INKTHRESHOLD | The stamping ink supply is low or empty; operator intervention is required. Note that this event is sent only once, at the point at which the supply becomes low or empty. It is sent with WFS_PTR_INKLOW or WFS_PTR_INKOUT status. |
| WFS_EXEE_PTR_MEDIAREJECTED | The media has been rejected and presented back to the user. It is available at the entry/exit slot. When the media is removed, a WFS_SRVE_PTR_MEDIATAKEN event will be sent. |
| WFS_SRVE_PTR_MEDIAAUTORETRACTED | The presented media has been automatically retracted. |

**Comments**     Printing of multiple pages is handled as described in section 11.

## 8.13 WFS_CMD_PTR_LOAD_DEFINITION

**Description** This command is used to load a form (including sub-forms and frames) or media definition into the list of available forms. Once a form or media definition has been loaded through this command it can be used by any of the other form/media definition processing commands. Forms and media definitions loaded through this command are persistently available across re-boots. When a form or media definition is loaded a WFS_PTR_SRVE_FORM_LOADED event is generated to inform applications that a form or media definition has been added or replaced.

**Input Param** LPWFSPTRLOADDEFINITION lpLoadDefinition;

```
typedef struct  wfs ptr load definition
     {
     LPSTR                lpszFileName;
     BOOL                 bOverwrite;
     } WFSPTRLOADDEFINITION, *LPWFSPTRLOADDEFINITION;
```

*lpszFileName*
Pointer to the null-terminated file name. This is the full path and file name of the file to be loaded. The file contains the form (including sub-forms and frames) or media definition in text format as described in the section 10 (ASCII or UNICODE). Only one form or media definition can be defined in the file.

*bOverwrite*
Specifies if an existing form or media definition with the same name is to be replaced. If this flag is TRUE then an existing form or media definition with the same name will be replaced, unless the command fails with an error, where the definition will remain unchanged. If this flag is FALSE this command will fail with an error if the form or media definition already exists.

**Output Param** None.

**Error Codes** In addition to the generic error codes defined in [Ref. 1], the following error codes can be generated by this command:

| Value | Meaning |
|---|---|
| WFS_ERR_PTR_FILENOTFOUND | The specified file cannot be found. |
| WFS_ERR_PTR_FORMINVALID | The form is invalid. |
| WFS_ERR_PTR_MEDIAINVALID | The media definition is invalid. |
| WFS_ERR_PTR_DEFINITIONEXISTS | The specified form or media definition already exists and the *bOverwrite* flag was FALSE. |

**Events** In addition to the generic events defined in [Ref. 1], the following events can be generated by this command:

| Value | Meaning |
|---|---|
| WFS_SRVE_PTR_DEFINITIONLOADED | A form or media definition has been loaded; an existing definition may have been modified by replacement. |

**Comments** None.

## 8.14 WFS_CMD_PTR_SUPPLY_REPLENISH

**Description**   After the supplies have been replenished, this command is used to indicate that one or more supplies have been replenished and are expected to be in a healthy state.

Hardware that cannot detect the level of a supply and reports on the supply's status using metrics (or some other means), must assume the supply has been fully replenished after this command is issued. The appropriate threshold event must be broadcast.

Hardware that can detect the level of a supply must update its status based on its sensors, generate a threshold event if appropriate, and succeed the command even if the supply has not been replenished. If it has already detected the level and reported the threshold before this command was issued, the command must succeed and no threshold event is required.

**Input Param**   LPWFSPTRSUPPLYREPLEN lpSupplyReplen;

```
typedef struct _wfs_ptr_supply_replen
{
    WORD              fwSupplyReplen;
} WFSPTRSUPPLYREPLEN, *LPWFSPTRSUPPLYREPLEN;
```

*fwSupplyReplen*
Specifies the supply that was replenished as a combination of the following flags:

| Value | Meaning |
|---|---|
| WFS_PTR_REPLEN_PAPERUPPER | The only paper supply or the upper paper supply was replenished. |
| WFS_PTR_REPLEN_PAPERLOWER | The lower paper supply was replenished. |
| WFS_PTR_REPLEN_PAPERAUX | The auxiliary paper supply was replenished. |
| WFS_PTR_REPLEN_PAPERAUX2 | The second auxiliary paper supply was replenished. |
| WFS_PTR_REPLEN_TONER | The toner supply was replenished. |
| WFS_PTR_REPLEN_INK | The ink supply was replenished. |
| WFS_PTR_REPLEN_LAMP | The imaging lamp was replaced. |

**Output Param**   None.

**Error Codes**   Only the generic error codes defined in [Ref. 1] can be generated by this command.

**Events**   In addition to the generic events defined in [Ref. 1], the following events can be generated by this command:

| Value | Meaning |
|---|---|
| WFS_USRE_PTR_PAPERTHRESHOLD | This user event is used to specify that the state of the paper supply threshold has been cleared. |
| WFS_USRE_PTR_TONERTHRESHOLD | This user event is used to specify that the state of the toner (or ink) supply threshold has been cleared. |
| WFS_USRE_PTR_INKTHRESHOLD | This user event is used to specify that the state of the stamping ink supply threshold has been cleared. |
| WFS_USRE_PTR_LAMPTHRESHOLD | This user event is used to specify that the state of the imaging lamp threshold has been cleared. |

**Comments**   If any one of the specified supplies is not supported by a Service Provider, WFS_ERR_UNSUPP_DATA should be returned, and no replenishment actions will be taken by the Service Provider.

## 8.15 WFS_CMD_PTR_POWER_SAVE_CONTROL

**Description** This command activates or deactivates the power-saving mode.

If the Service Provider receives another execute command while in power saving mode, the Service Provider automatically exits the power saving mode, and executes the requested command. If the Service Provider receives an information command while in power saving mode, the Service Provider will not exit the power saving mode.

**Input Param** LPWFSPTRPOWERSAVECONTROL lpPowerSaveControl;

```
typedef struct _wfs_ptr_power_save_control
    {
    USHORT              usMaxPowerSaveRecoveryTime;
    } WFSPTRPOWERSAVECONTROL, *LPWFSPTRPOWERSAVECONTROL;
```

*usMaxPowerSaveRecoveryTime*
Specifies the maximum number of seconds in which the device must be able to return to its normal operating state when exiting power save mode. The device will be set to the highest possible power save mode within this constraint. If *usMaxPowerSaveRecoveryTime* is set to zero then the device will exit the power saving mode.

**Output Param** None.

**Error Codes** In addition to the generic error codes defined in [Ref. 1], the following error codes can be generated by this command:

| Value | Meaning |
| --- | --- |
| WFS_ERR_PTR_POWERSAVETOOSHORT | The power saving mode has not been activated because the device is not able to resume from the power saving mode within the specified *usMaxPowerSaveRecoveryTime* value. |
| WFS_ERR_PTR_POWERSAVEMEDIAPRESENT | The power saving mode has not been activated because media is present inside the device. |

**Events** In addition to the generic events defined in [Ref. 1], the following events can be generated by this command:

| Value | Meaning |
| --- | --- |
| WFS_SRVE_PTR_POWER_SAVE_CHANGE | The power save recovery time has changed. |

**Comments** None.

# 9. Events

## 9.1 WFS_EXEE_PTR_NOMEDIA

**Description**     This event specifies that the physical media must be inserted into the device in order for the execute command to proceed.

**Event Param**     LPSTR lpszUserPrompt;

*lpszUserPrompt*
Pointer to a null-terminated user prompt string from the form definition. NULL will be returned if either a form does not define a value for the user prompt or the event is being generated as the result of a command that does not use forms.

**Comments**     The application may use the *lpszUserPrompt* in any manner it sees fit, for example it might display the string to the operator, along with a message that the media should be inserted.

## 9.2 WFS_EXEE_PTR_MEDIAINSERTED

**Description**    This event specifies that the physical media has been inserted into the device.

**Event Param**    None.

**Comments**    The application may use this event to, for example, remove a message box from the screen telling the user to insert a form.

## 9.3 WFS_EXEE_PTR_FIELDERROR

**Description**  This event specifies that a fatal error has occurred while processing a field.

**Event Param**  LPWFSPTRFIELDFAIL lpFieldFail;

```
typedef struct _wfs_ptr_field_failure
     {
     LPSTR                  lpszFormName;
     LPSTR                  lpszFieldName;
     WORD                   wFailure;
     } WFSPTRFIELDFAIL, *LPWFSPTRFIELDFAIL;
```

*lpszFormName*
Points to the null-terminated form name.

*lpszFieldName*
Points to the null-terminated field name.

*wFailure*
Specifies the type of failure and can be one of the following values:

| Value | Meaning |
| --- | --- |
| WFS_PTR_FIELDREQUIRED | The specified field must be supplied by the application. |
| WFS_PTR_FIELDSTATICOVWR | The specified field is static and thus cannot be overwritten by the application. |
| WFS_PTR_FIELDOVERFLOW | The value supplied for the specified fields is too long. |
| WFS_PTR_FIELDNOTFOUND | The specified field does not exist. |
| WFS_PTR_FIELDNOTREAD | The specified field is not an input field. |
| WFS_PTR_FIELDNOTWRITE | An attempt was made to write to an input field. |
| WFS_PTR_FIELDHWERROR | The specified field uses special hardware (e.g. OCR) and an error occurred. |
| WFS_PTR_FIELDTYPENOTSUPPORTED | The form field type is not supported with device. |
| WFS_PTR_FIELDGRAPHIC | The specified graphic image could not be printed. |
| WFS_PTR_CHARSETFORM | Service Provider does not support character set specified in form. |

**Comments**  None.

## 9.4 WFS_EXEE_PTR_FIELDWARNING

**Description**     This event is used to specify that a non-fatal error has occurred while processing a field.

**Event Param**     LPWFSPTRFIELDFAIL lpFieldFail;

As defined in the section describing WFS_EXEE_PTR_FIELDERROR.

**Comments**     None.

## 9.5 WFS_USRE_PTR_RETRACTBINTHRESHOLD

**Description**    This event specifies that the status of the retract bin holding the retracted media has changed.

**Event Param**    LPWFSPTRBINTHRESHOLD lpBinThreshold;

```
typedef struct _wfs_ptr_bin_threshold
    {
    USHORT              usBinNumber;
    WORD                wRetractBin;
    } WFSPTRBINTHRESHOLD, *LPWFSPTRBINTHRESHOLD;
```

*usBinNumber*
Number of the retract bin for which the status has changed.

*wRetractBin*
Specifies the current state of the retract bin as one of the following values:

| Value | Meaning |
|---|---|
| WFS_PTR_RETRACTBINOK | The retract bin of the printer is in a good state. |
| WFS_PTR_RETRACTBINFULL | The retract bin of the printer is full. |
| WFS_PTR_RETRACTBINHIGH | The retract bin of the printer is high. |

**Comments**    None.

## 9.6 WFS_SRVE_PTR_MEDIATAKEN

**Description**  This event is sent when the media is taken from the exit slot following the completion of a successful eject operation or following a WFS_EXEE_PTR_MEDIAREJECTED event.

**Event Param**  None.

**Comments**  Note that since this event can occur after the completion of a function that includes a media eject, it is not an execute event.

## 9.7  WFS_USRE_PTR_PAPERTHRESHOLD

**Description**  This user event is used to specify that the state of the paper reached a threshold. There is no threshold defined for the parking station as this can contain only one paper item.

**Event Param**  LPWFSPTRPAPERTHRESHOLD lpPaperThreshold;

```
typedef struct _wfs_ptr_paper_threshold
    {
    WORD                wPaperSource;
    WORD                wPaperThreshold;
    } WFSPTRPAPERTHRESHOLD, *LPWFSPTRPAPERTHRESHOLD;
```

*wPaperSource*
Specifies the paper sources as one of the following values:

| Value | Meaning |
|---|---|
| WFS_PTR_PAPERUPPER | The only paper source or the upper paper source, if there is more than one paper supply. |
| WFS_PTR_PAPERLOWER | The lower paper source. |
| WFS_PTR_PAPEREXTERNAL | The external paper source (such as envelope tray or single sheet feed). |
| WFS_PTR_PAPERAUX | The auxiliary paper source. |
| WFS_PTR_PAPERAUX2 | The second auxiliary paper source. |

*wPaperThreshold*
Specifies the current state of the paper source as one of the following values:

| Value | Meaning |
|---|---|
| WFS_PTR_PAPERFULL | The paper in the paper source is in a good state. |
| WFS_PTR_PAPERLOW | The paper in the paper source is low. |
| WFS_PTR_PAPEROUT | The paper in the paper source is out. |

**Comments**  None.

| **Deleted:** An |
| **Deleted:** available, devices with only |
| **Deleted:** must indicate WFS_PTR_PAPERUPPER as being available. ¶ |

## 9.8  WFS_USRE_PTR_TONERTHRESHOLD

**Description**    This user event is used to specify that the state of the toner (or ink) reached a threshold.

**Event Param**    LPWORD lpwTonerThreshold;

*lpwTonerThreshold*
Specifies the current state of the toner (or ink) as one of the following values:

| Value | Meaning |
| --- | --- |
| WFS_PTR_TONERFULL | The toner (or ink) in the printer is in a good state. |
| WFS_PTR_TONERLOW | The toner (or ink) in the printer is low. |
| WFS_PTR_TONEROUT | The toner (or ink) in the printer is out. |

**Comments**    None.

## 9.9  WFS_SRVE_PTR_MEDIAINSERTED

**Description**     This event specifies that the physical media has been inserted into the device without any read or print execute commands being executed. This event is only generated when media is entered in an unsolicited manner.

**Event Param**     None.

**Comments**        None.

## 9.10 WFS_USRE_PTR_LAMPTHRESHOLD

**Description**    This user event is used to specify that the state of the imaging lamp reached a threshold.

**Event Param**    LPWORD lpwLampThreshold;

*lpwLampThreshold*
Specifies the current state of the imaging lamp as one of the following values:

| Value | Meaning |
|---|---|
| WFS_PTR_LAMPOK | The imaging lamp is in a good state. |
| WFS_PTR_LAMPFADING | The imaging lamp is fading and should be changed. |
| WFS_PTR_LAMPINOP | The imaging lamp is inoperative. |

**Comments**    None.

## 9.11 WFS_USRE_PTR_INKTHRESHOLD

**Description**     This user event is used to specify that the state of the stamping ink reached a threshold.

**Event Param**     LPWORD lpwInkThreshold;

*lpwInkThreshold*
Specifies the current state of the stamping ink as one of the following values:

| Value | Meaning |
| --- | --- |
| WFS_PTR_INKFULL | The stamping ink in the printer is in a good state. |
| WFS_PTR_INKLOW | The stamping ink in the printer is low. |
| WFS_PTR_INKOUT | The stamping ink in the printer is out. |

**Comments**     None.

## 9.12 WFS_SRVE_PTR_MEDIADETECTED

**Description**     This event is generated when a media is detected in the device during a reset operation.

**Event Param**     LPWFSPTRMEDIADETECTED lpMediaDetected;

```
typedef struct _wfs_ptr_media_detected
    {
    WORD                wPosition;
    USHORT              usRetractBinNumber;
    } WFSPTRMEDIADETECTED, *LPWFSPTRMEDIADETECTED;
```

*wPosition*
Specifies the media position after the reset operation, as one of the following values:

| Value | Meaning |
|-------|---------|
| WFS_PTR_MEDIARETRACTED | The media was retracted during the reset operation. |
| WFS_PTR_MEDIAPRESENT | The media is in the print position or on the stacker. |
| WFS_PTR_MEDIAENTERING | The media is in the exit slot. |
| WFS_PTR_MEDIAJAMMED | The media is jammed in the device. |
| WFS_PTR_MEDIAUNKNOWN | The media is in an unknown position. |
| WFS_PTR_MEDIAEXPELLED | The media was expelled during the reset operation. |

*usRetractBinNumber*
Number of the retract bin the media was retracted to. This number has to be between one and the number of bins supported by this device. It is only relevant if *wPosition* equals WFS_PTR_MEDIARETRACTED.

**Comments**     None.

## 9.13 WFS_SRVE_PTR_RETRACTBINSTATUS

**Description**      This event specifies that the status of the retract bin has changed.

**Event Param**      LPWFSPTRBINSTATUS lpBinStatus;

```
typedef struct _wfs_ptr_bin_status
    {
    USHORT              usBinNumber;
    WORD                wRetractBin;
    } WFSPTRBINSTATUS, *LPWFSPTRBINSTATUS;
```

*usBinNumber*
Number of the retract bin for which the status has changed.

*wRetractBin*
Specifies the current state of the retract bin as one of the following values:

| Value | Meaning |
|---|---|
| WFS_PTR_RETRACTBININSERTED | The retract bin has been inserted. |
| WFS_PTR_RETRACTBINREMOVED | The retract bin has been removed. |

**Comments**      None.

## 9.14 WFS_EXEE_PTR_MEDIAPRESENTED

**Description**     This event is used to indicate when media has been presented to the customer for removal.

**Event Param**     LPWFSPTRMEDIAPRESENTED lpMediaPresented;

```
typedef struct _wfs_ptr_media_presented
    {
    USHORT              usWadIndex;
    USHORT              usTotalWads;
    } WFSPTRMEDIAPRESENTED, *LPWFSPTRMEDIAPRESENTED;
```

*usWadIndex*
Specifies the index (starting from one) of the presented wad, where a Wad is a bunch of one or more pages presented as a bunch.

*usTotalWads*
Specifies the total number of wads in the print job, zero if the total number of wads is not known.

**Comments**     None.

## 9.15 WFS_SRVE_PTR_DEFINITIONLOADED

**Description**    This event is used to indicate when a form or media definition has successfully been loaded via the WFS_CMD_PTR_LOAD_DEFINITION command.

**Event Param**    LPWFSPTRDEFINITIONLOADED lpDefinitionLoaded;

```
typedef struct _wfs_ptr_definition_loaded
     {
     LPSTR               lpszDefinitionName;
     DWORD               dwDefinitionType;
     } WFSPTRDEFINITIONLOADED, *LPWFSPTRDEFINITIONLOADED;
```

*lpszDefinitionName*
Specifies the name of the form or media just loaded.

*dwDefinitionType*
Specifies the type of definition loaded. This field can be one of the following values:

| Value | Meaning |
|---|---|
| WFS_PTR_FORMLOADED | The form identified by *lpszDefinitionName* has been loaded. |
| WFS_PTR_MEDIALOADED | The media identified by *lpszDefinitionName* has been loaded. |

**Comments**    None.

## 9.16 WFS_EXEE_PTR_MEDIAREJECTED

**Description**   This event is generated as a result of physical media that is rejected whenever an attempt is made to insert media into the physical device. Rejection of the media will cause the command currently executing to complete with a WFS_ERR_PTR_MEDIAREJECTED error, at which point the media should be removed.

**Event Param**   LPWFSPTRMEDIAREJECTED lpMediaRejected;

```
typedef struct  wfs ptr media rejected
        {
     WORD                wMediaRejected;
        } WFSPTRMEDIAREJECTED, *LP WFSPTRMEDIAREJECTED;
```

*wMediaRejected*
Specifies the reason for rejecting the media as one of the following values:

| Value | Meaning |
|---|---|
| WFS_PTR_REJECT_SHORT | The rejected media was too short. |
| WFS_PTR_REJECT_LONG | The rejected media was too long. |
| WFS_PTR_REJECT_MULTIPLE | The media was rejected due to insertion of multiple documents. |
| WFS_PTR_REJECT_ALIGN | The media could not be aligned and was rejected. |
| WFS_PTR_REJECT_MOVETOALIGN | The media could not be transported to the align area and was rejected. |
| WFS_PTR_REJECT_SHUTTER | The media was rejected due to the shutter failing to close. |
| WFS_PTR_REJECT_ESCROW | The media was rejected due to problems transporting media to the escrow position. |
| WFS_PTR_REJECT_THICK | The rejected media was too thick. |
| WFS_PTR_REJECT_OTHER | The media was rejected due to a reason other than those listed above. |

**Comments**   The application may use this event to (for example) display a message box on the screen indicating why the media was rejected, and telling the user to remove and reinsert the media.

## 9.17 WFS_SRVE_PTR_MEDIAPRESENTED

**Description**    This event is used to indicate when media has been presented to the customer for removal as a result of a print operation through some non XFS interface.

**Event Param**    LPWFSPTRMEDIAPRESENTED lpMediaPresented;

```
typedef struct _wfs_ptr_media_presented
    {
    USHORT              usWadIndex;
    USHORT              usTotalWads;
    } WFSPTRMEDIAPRESENTED, *LPWFSPTRMEDIAPRESENTED;
```

*usWadIndex*
Specifies the index (starting from one) of the presented wad, where a Wad is a bunch of one or more pages presented as a bunch.

*usTotalWads*
Specifies the total number of wads in the print job, zero if the total number of wads is not known.

**Comments**    None.

## 9.18 WFS_SRVE_PTR_MEDIAAUTORETRACTED

**Description**    This event indicates when media has been automatically retracted by the device. Support for this event is indicated when the *usAutoRetractPeriod* field of the WFS_INF_PTR_CAPABILITIES output structure is non-zero. The event can be generated as the result of any command that presents media to the customer.

**Event Param**    LPWFSPTRMEDIARETRACTED lpMediaRetracted

```
typedef struct  wfs ptr media retracted
    {
    WORD                wRetractResult;
    USHORT              usBinNumber;
    } WFSPTRMEDIARETRACTED, *LPWFSPTRMEDIARETRACTED;
```

*wRetractResult*
Specifies the result of the automatic retraction, as one of the following values:

| Value | Meaning |
|---|---|
| WFS_PTR_AUTO_RETRACT_OK | The media was retracted successfully. |
| WFS_PTR_AUTO_RETRACT_MEDIAJAMMED | |
| | The media is jammed. |

*usBinNumber*
Number of the retract bin the media was retracted to. This number has to be between one and the number of bins supported by this device. This value is zero if *wRetractResult* is WFS_PTR_AUTO_RETRACT_MEDIAJAMMED.

**Comments**    None.

## 9.19 WFS_SRVE_PTR_DEVICEPOSITION

**Description**    This service event reports that the device has changed its position status.

**Event Param**    LPWFSPTRDEVICEPOSITION lpDevicePosition;

```
typedef struct _wfs_ptr_device_position
     {
     WORD                    wPosition;
     } WFSPTRDEVICEPOSITION, *LPWFSPTRDEVICEPOSITION;
```

*wPosition*
Position of the device as one of the following values:

| Value | Meaning |
|---|---|
| WFS_PTR_DEVICEINPOSITION | The device is in its normal operating position. |
| WFS_PTR_DEVICENOTINPOSITION | The device has been removed from its normal operating position. |
| WFS_PTR_DEVICEPOSUNKNOWN | The position of the device cannot be determined. |

**Comments**    None.

## 9.20 WFS_SRVE_PTR_POWER_SAVE_CHANGE

**Description**    This service event specifies that the power save recovery time has changed.

**Event Param**    LPWFSPTRPOWERSAVECHANGE lpPowerSaveChange;

```
typedef struct _wfs_ptr_power_save_change
    {
    USHORT                usPowerSaveRecoveryTime;
    } WFSPTRPOWERSAVECHANGE, *LPWFSPTRPOWERSAVECHANGE;
```

*usPowerSaveRecoveryTime*
Specifies the actual number of seconds required by the device to resume its normal operational state. This value is zero if the device exited the power saving mode.

**Comments**    None.

# 10. Form, Sub-Form, Field, Frame, Table and Media Definitions

This section outlines the format of the definitions of forms, the fields within them, optional tables and fields within the form, and the media on which they are printed.

## 10.1 Definition Syntax

The syntactic rules for form, field and media definitions are as follows:

- White space        space, tab

- Line continuation      backslash (\)

- Line termination      CR, LF, CR/LF; line termination ends a "keyword section"
  (a keyword and its value[s])

- Keywords         must be all upper case

- Names          (field/media/font names) any case; case is preserved;
  Service Providers are case sensitive

- Strings          all strings must be enclosed in double quote characters (");
  standard C escape sequences are allowed.

- Comments         start with two forward slashes (//), end at line termination

Other notes:

- The values of a keyword are separated by commas.

- If a keyword is present, all its values must be specified; default values are used only if the keyword is absent.

- Values that are character strings are marked with asterisks in the definitions below, and must be quoted as specified above.

- The order of attributes within the forms is not mandatory and the attributes may be defined in any order.

- All forms can be represented using either ISO 646 (ANSI) or UNICODE character encoding. If the UNICODE representation is used then all Names and Strings are restricted to an internal representation of ISO 646 (ANSI) characters. Only the INITIALVALUE and FORMAT keyword values can have double byte values outside of the ISO 646 (ANSI) character set.

- If forms character encoding is UNICODE then, consistent with the UNICODE standard, the file prefix must be in Little Endian (0xFFFE) or Big Endian (0xFEFF) notation, such that UNICODE encoding is recognized.

## 10.2 Form and Media Measurements

The UNIT keyword sections of the form and media definitions specify the base horizontal and vertical resolution as follows:

- The *base* value specifies the base unit of measurement.

- The *x* and *y* values specify the horizontal and vertical resolution as fractions of the base value (e.g. an *x* value of 10 and a base value of MM means that the base horizontal resolution is 0.1mm).

The base resolutions thus defined by the UNIT keyword section of the XFSFORM definition are used as the units of the form definition keyword sections:

- SIZE (*width* and *height* values)

- ALIGNMENT (*xoffset* and *yoffset* values)

and of the sub-form definition keyword sections:

- POSITION (*x* and *y* values)

- SIZE (*width* and *height* values)

and of the field definition keyword sections:

- POSITION (*x* and *y* values)

- SIZE (*width* and *height* values)

- INDEX (*xoffset* and *yoffset* values)

and of the frame definition keyword sections:

- POSITION (*x* and *y* values)

- SIZE (*width* and *height* values)

- REPEATONX (x*offset* value)

- REPEATONY (*yoffset* value)

The base resolutions thus defined by the UNIT keyword section of the XFSMEDIA definition are used as the units of the media definition keyword sections:

- SIZE (*width* and *height* values)

- PRINTAREA (*x*, *y*, *width* and *height* values)

- RESTRICTED (*x*, *y*, *width* and *height* values)

NOTE: The origin for coordinate based systems is (0,0). The origin for row/column based systems can be (0,0) or (1,1) and must be configurable within the Service Provider.

## 10.3 Form Definition [1]

| XFSFORM | | formname* | |
|---|---|---|---|
| **BEGIN** | | | |
| **(required)** | **UNIT** | base, | Base resolution unit for form definition:<br>    MM<br>    INCH<br>    ROWCOLUMN |
| | | x, | Horizontal base unit fraction |
| | | y | Vertical base unit fraction |
| **(required)** | **SIZE** | width, | Width of form |
| | | height | Height of form |
| | **ALIGNMENT** | alignment, | Alignment of the form on the physical media:<br>    TOPLEFT (default)<br>    TOPRIGHT<br>    BOTTOMLEFT<br>    BOTTOMRIGHT<br>This option allows the positioning of a form onto a physical page relative to any combination of the edges of the physical media, to support the variations in how devices sense the edge of page for positioning purposes. |
| | | xoffset, | Horizontal offset relative to the horizontal alignment specified by alignment. Always specified as a positive value (i.e. if aligned to the right side of the media, means offset the form to the left). (default = 0) |
| | | yoffset | Vertical offset relative to the vertical alignment specified by alignment. Always specified as a positive value (i.e. if aligned to the bottom of the media, means offset the form upward). (default = 0) |
| | **ORIENTATION** | type | Orientation of form:<br>    PORTRAIT (default)<br>    LANDSCAPE |
| | **SKEW** | skewfactor | Maximum skew factor in degrees (default = 0) |
| | **VERSION** | major, | Major version number |
| | | minor, | Minor version number |
| | | date*, | Creation/modification date |
| | | author* | Author of form |
| **(required)** | **LANGUAGE** | languageID | Language used in this form - a 16 bit value (LANGID) which is a combination of a primary (10 bits) and a secondary (6 bits) language ID (This is the standard language ID in the Win32 API; standard macros support construction and decomposition of this composite ID) |
| | **CPI** | cpi | Characters per inch.<br>This value specifies the default CPI within the form. When the ROWCOLUMN unit is used, the form CPI and LPI are used to calculate the position and size of all fields within a form, irrespective of the CPI and LPI of the fields themselves. |
| | **LPI** | lpi | Lines per inch.<br>This value specifies the default LPI within the form. When the ROWCOLUMN unit is used, the form CPI and LPI are used to calculate the position and size of all fields within a form, irrespective of the CPI and LPI of the fields themselves. |
| | **POINTSIZE** | pointsize | This value specifies the default POINTSIZE within the form. |
| | **COPYRIGHT** | copyright* | Copyright entry |
| | **TITLE** | title* | Title of form |
| | **COMMENT** | comment* | Comment section |
| | **USERPROMPT** | prompt* | Prompt string for user interaction |

---

[1] Attributes are not required in any mandatory order within a Form definition.

| | [ XFSFIELD<br><br>BEGIN<br>. . .<br>END ] | *fieldname\** | One field definition (as defined in the next section) for each field in the form |
|---|---|---|---|
| | [ XFSFRAME<br><br>BEGIN<br>. . .<br>END ] | *framename\** | One frame definition (as defined in the next section) for each frame in the form |
| | [ XFSSUBFORM<br><br>BEGIN<br>. . .<br>END ] | *subformname\** | One subform definition (as defined in the next section) for each subform in the form |
| END | | | |

## 10.4 SubForm Definition [2]

| XFSSUBFORM | | subformname* | |
|---|---|---|---|
| BEGIN | | | |
| (required) | POSITION | X,<br>Y or (Y, Z) | Horizontal position (relative to left side of form)<br>Vertical position (relative to top of form).<br>Format (Y, Z) is used to indicate vertical positioning relative to top of form when top of form is other than 1st page of form, where Z indicates page number (relative to 0) and Y indicates base resolution units relative to top of the form page number (as indicated by Z).<br>Format Y is used to indicate vertical positioning relative to top of the 1st form page. |
| (required) | SIZE | width,<br>height | Width of subform. Width must not exceed width of form.<br>Height of subform. Height must not exceed height of form. |
| | [ XFSFIELD<br><br>BEGIN<br>. . .<br>END ] | fieldname* | One field definition (as defined in the next section) for each field in the subform |
| | [ XFSFRAME<br><br>BEGIN<br>. . .<br>END ] | framename* | One frame definition (as defined in the next section) for each frame in the subform |
| END | | | |

The XFSSUBFORM definition provides a means to isolate a selected area of a form where the user may want to have a select group of fields, frames, and/or running headers and footers. All field and frame definitions within a subform are relative to the POSITION of the subform. A form definition with an imbedded subform will have a series of statements illustrated as follows:

```
XFSFORM
BEGIN
   *
   *
   XFSSUBFORM
   BEGIN
     XFSFIELD
     BEGIN
        *
        *
     END
     XFSFIELD
     BEGIN
        *
        *
     END
   END
END
```

---

[2] Attributes are not required in any mandatory order within a SubForm definition.

## 10.5 Field Definition [3]

| XFSFIELD | | *fieldname** | |
|---|---|---|---|
| BEGIN | | | |
| (required) | POSITION | *X,*<br>*Y or (Y, Z)* | Horizontal position (relative to left side of form/subform). Vertical position (relative to top of form/subform). Format (Y, Z) is used to indicate vertical positioning relative to top of form/subform when top of form/subform is other than 1<sup>st</sup> page of form/subform, where Z indicates page number (relative to 0) and Y indicates base resolution units relative to top of the form/subform page number (as indicated by Z). Format Y is used to indicate vertical positioning relative to top of the 1<sup>st</sup> form/subform. |
| | FOLLOWS | *fieldname** | Print this field directly following the field with the name <fieldname>; positioning information is ignored. See the description of WFS_CMD_PTR_PRINT_FORM. If FOLLOWS is omitted then fields are printed in the order that they appear in the form definition. |
| | HEADER | *N*<br><br>*N-N*<br><br><br>*ALL* | This field is either a form/subform header field. N represents a form/subform page number (relative to 0) the header field is to print within. N-N represents a form/subform page number range the header field is to print within. Combinations of N and N-N may exist separated by commas. ALL indicates that header field is to be printed on all pages of form/subform. The form/subform page number is intended to supplement the Z parameter of the POSITION keyword. For example 0,2-4,6 indicates that the header field is to print on relative form/subform pages 0, 2, 3, 4, and 6. |
| | FOOTER | *N*<br><br>*N-N*<br><br><br>*All* | This field is either a form/subform footer field. N represents a form/subform page number (relative to 0) the footer field is to print within. N-N represents a form/subform page number range the footer field is to print within. Combinations of N and N-N may exist separated by commas. ALL indicates that footer field is to be printed on all pages of form/subform. The form/subform page number is intended to supplement the Z parameter of the POSITION keyword. For example 0,2-4,6 indicates that the footer field is to print on relative form/subform pages 0, 2, 3, 4, and 6. |
| | SIDE | *side* | Side of form where field is positioned: FRONT (default) BACK |
| (required) | SIZE | *width,*<br>*height* | Field width Field height |
| | INDEX | *repeatcount,*<br><br>*xoffset,*<br>*yoffset* | Count how often this field is repeated in the form, INDEX fields are fixed length. (default is no INDEX field) Horizontal offset for next field Vertical offset for next field |

---

[3] Attributes are not required in any mandatory order within a Field definition.

| | TYPE | *fieldtype* | Type of field:<br>     TEXT (default)<br>     MICR<br>     OCR<br>     MSF<br>     BARCODE<br>     GRAPHIC<br>     PAGEMARK |
|---|---|---|---|
| | SCALING | *scalingtype* | Information on how to size the GRAPHIC within the field:<br>     BESTFIT (default) scale to size indicated<br>     ASIS    render at native size<br>     MAINTAINASPECT<br>         scale as close as possible to size indicated while maintaining the aspect ratio and not losing graphic information.<br>SCALING is only relevant for GRAPHIC field types. |
| | BARCODE | *hriposition* | Position of the HRI (Human Readable Interpretation) characters:<br>     NONE (default)<br>     ABOVE<br>     BELOW<br>     BOTH<br>The type of barcode to print is defined in the FONT field. |
| | CLASS | *class* | Field class:<br>     OPTIONAL (default)<br>     STATIC<br>     REQUIRED |
| | ACCESS | *access* | Access rights of field:<br>     WRITE (default)<br>     READ<br>     READWRITE |
| | OVERFLOW | *overflow* | Action on field overflow:<br>     TERMINATE (default)<br>     TRUNCATE<br>     BESTFIT (the Service Provider fits the data into the field as well as it can)<br>     OVERWRITE (a contiguous write)<br>     WORDWRAP |

| | STYLE | *style* | Display attributes as a combination of the following, ORed together using the "\|" operator:<br>      NORMAL (default)<br>      BOLD<br>      ITALIC<br>      UNDER (single underline)<br>      DOUBLEUNDER (double underline)<br>      DOUBLE (double width)<br>      TRIPLE (triple width)<br>      QUADRUPLE (quadruple width)<br>      STRIKETHROUGH<br>      ROTATE90 (rotate 90 degrees clockwise)<br>      ROTATE270 (rotate 270 degrees clockwise)<br>      UPSIDEDOWN (upside down)<br>      PROPORTIONAL (proportional spacing)<br>      DOUBLEHIGH<br>      TRIPLEHIGH<br>      QUADRUPLEHIGH<br>      CONDENSED<br>      SUPERSCRIPT<br>      SUBSCRIPT<br>      OVERSCORE<br>      LETTERQUALITY<br>      NEARLETTERQUALITY<br>      DOUBLESTRIKE<br>      OPAQUE (If omitted then default attribute<br>             is transparent)<br>Some of these Styles may be mutually exclusive, or may combine to provide unexpected results. |
| | CASE | *case* | Convert field contents to:<br>      NOCHANGE (default)<br>      UPPER<br>      LOWER |
| | HORIZONTAL | *justify* | Horizontal alignment of field contents:<br>      LEFT    (default)<br>      RIGHT<br>      CENTER<br>      JUSTIFY |
| | VERTICAL | *justify* | Vertical alignment of field contents:<br>      BOTTOM (default)<br>      CENTER<br>      TOP |
| | COLOR | *color* | Color name:<br>      BLACK (default)<br>      WHITE<br>      GRAY<br>      RED<br>      BLUE<br>      GREEN<br>      YELLOW |
| | RGBCOLOR | *R, G, B* | Color in RGB 8 bits per color format:<br>      R - Red portion of the RGB value 0-255.<br>      G - Green portion of the RGB value 0-255.<br>      B - Blue portion of the RGB value 0-255.<br>RGBCOLOR overrides the COLOR attribute. |
| | LANGUAGE | *languageID* | Language used in this field - a 16 bit value (LANGID) which is a combination of a primary (10 bits) and a secondary (6 bits) language ID (This is the standard language ID in the Win32 API; standard macros support construction and decomposition of this composite ID).<br>If unspecified defaults to form definition LANGUAGE specification. |

| | FONT | *fontname\** | Font name: This attribute is interpreted by the Service Provider. In some cases it may indicate printer resident fonts, and in others it may indicate the name of a downloadable font. For BARCODE fields it represents the barcode font name. In some cases this predefines the following parameters: |
| --- | --- | --- | --- |
| | POINTSIZE | *pointsize* | Point size. If unspecified, the point size defaults to the POINTSIZE defined for the form. |
| | CPI | *cpi* | Characters per inch. If unspecified, the CPI defaults to the CPI defined for the form. |
| | LPI | *lpi* | Lines per inch. If unspecified, the LPI defaults to the LPI defined for the form. |
| | FORMAT | *formatstring\** | This is an application defined input field describing how the application should format the data. This may be interpreted by the Service Provider. |
| | INITIALVALUE | *value\** | Initial value. For GRAPHIC type fields, this value may contain the filename of the graphic image. The type of this graphic will be determined by the file extension (e.g. BMP for Windows Bitmap). Graphic file name may be full or partial path. For example "C:\BSVC\BSVCLOGO.BMP" illustrates use of full path name. A file name specification of "LOGO.BMP" illustrates partial path name. In this instance file is obtained from current directory. Graphic contents can be changed dynamically at run-time and the new content will be printed on the next print action. |
| END | | | |

The following diagrams illustrate the positioning and sizing of text fields on a form, and, in particular, the vertical alignment of text within a field using **VERTICAL=TOP** and **VERTICAL=BOTTOM** values in the field definition.
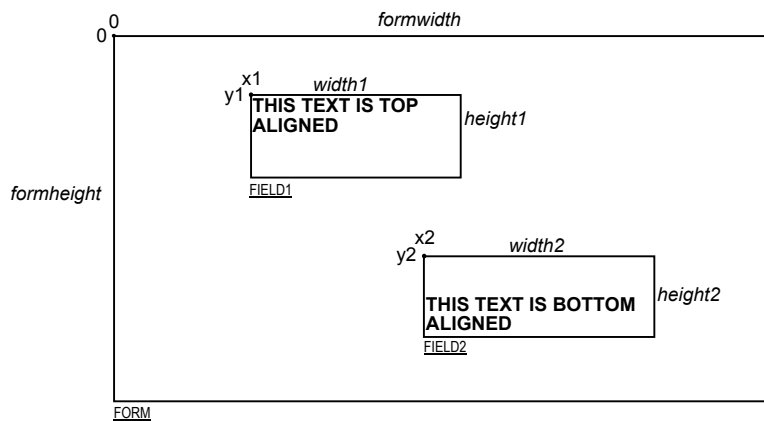


| | | |
|---|---|---|
| **VERTICAL=TOP** | the upper boundary of the character drawing box (shown below) is positioned vertically to the upper field boundary. |
| **VERTICAL=BOTTOM** | the baseline of the character drawing box (shown below) is positioned vertically to the lower field boundary. |

Definition of the character drawing box:
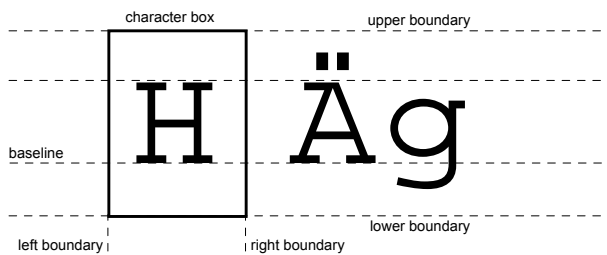


When more than one line of text is to be printed in a field, and the definition includes **VERTICAL=BOTTOM,** the vertical position of the first line is calculated using the specified (or implied) **LPI** value.

## 10.6 Frame Definition [4]

| XFSFRAME | | framename* | |
|---|---|---|---|
| BEGIN | | | |
| (required) | POSITION | X, | Horizontal position of top left corner of the frame (relative to left side of form/subform). |
| | | Y or (Y, Z) | Vertical position of top left corner of frame (relative to top of form/subform). Format (Y, Z) is used to indicate vertical positioning of the top left corner of the frame relative to top of form/subform when top of form/subform is other than 1st page of form/subform, where Z indicates page number (relative to 0) and Y indicates base resolution units relative to top of the form/subform page number (as indicated by Z). Format Y is used to indicate vertical positioning of the left corner of frame relative to top of the 1st form/subform. |
| | FRAMES | fieldname* | Frames the field with the name <fieldname>, positioning and size information are ignored. The frame surrounds the complete field, not just the printed data. If the field is repeated, the frame surrounds the first and last fields that are printed. |
| | HEADER | | This frame is either a form/subform header frame. |
| | | N | N represents a form/subform page number (relative to 0) the header frame is to print within. |
| | | N-N | N-N represents a form/subform page number range the header frame is to print within. Combinations of N and N-N may exist separated by commas. |
| | | ALL | ALL indicates that header frame is to be printed on all pages of form/subform. The form/subform page number is intended to supplement the Z parameter of the POSITION keyword. For example 0,2-4,6 indicates that the header frame is to print on relative form/subform pages 0, 2, 3, 4, and 6. |
| | FOOTER | | This field is either a form/subform footer frame. |
| | | N | N represents a form/subform page number (relative to 0) the footer frame is to print within. |
| | | N-N | N-N represents a form/subform page number range the footer frame is to print within. Combinations of N and N-N may exist separated by commas. |
| | | ALL | ALL indicates that footer frame is to be printed on all pages of form/subform. The form/subform page number is intended to supplement the Z parameter of the POSITION keyword. For example 0,2-4,6 indicates that the footer frame is to print on relative form/subform pages 0, 2, 3, 4, and 6. |
| | SIDE | side | Side of form where this frame is positioned: FRONT (default) BACK |
| (required) | SIZE | width, height | Frame width in base horizontal units for the form Frame height in base vertical units for the form |
| | REPEATONX | repeatcount, xoffset | Count how often this frame is repeated horizontally in the form. Horizontal offset for next frame in base horizontal units. |
| | REPEATONY | repeatcount, yoffset | Count how often this frame is repeated vertically in the form. Vertical offset for next frame in base vertical units. |

---

[4] Attributes are not required in any mandatory order within a Frame definition.

| | TYPE | *frametype* | Type of frame:<br>RECTANGLE (default)<br>ROUNDED_CORNER<br>ELLIPSE |
|---|---|---|---|
| | **CLASS** | *class* | Frame class:<br>STATIC (default)<br>OPTIONAL<br>(The frame is printed only if its name appears in the list of field names given as parameter to the WFSExecute command. In this case, the name of the frame must be different from all the names of the fields.) |
| | **OVERFLOW** | *overflow* | Action on frame overflowing the form:<br>TERMINATE (default)<br>TRUNCATE<br>BESTFIT (the Service Provider fits the frame into the media as well as it can) |
| | **STYLE** | *style* | Frame line attributes:<br>SINGLE_THIN (default)<br>DOUBLE_THIN<br>SINGLE_THICK<br>DOUBLE_THICK<br>DOTTED |
| | **COLOR** | *color* | Color name for frame lines:<br>BLACK (default)<br>WHITE<br>GRAY<br>RED<br>BLUE<br>GREEN<br>YELLOW |
| | **RGBCOLOR** | *R, G, B* | Color in RGB 8 bits per color format:<br>R - Red portion of the RGB value 0-255.<br>G - Green portion of the RGB value 0-255.<br>B - Blue portion of the RGB value 0-255.<br>RGBCOLOR overrides the COLOR attribute. |
| | **FILLCOLOR** | *color* | Color name for interior of frame:<br>BLACK<br>WHITE (default)<br>GRAY<br>RED<br>BLUE<br>GREEN<br>YELLOW |
| | **RGBFILLCOLOR** | *R, G, B* | Color in RGB 8 bits per color format:<br>R - Red portion of the RGB value 0-255.<br>G - Green portion of the RGB value 0-255.<br>B - Blue portion of the RGB value 0-255.<br>RGBFILLCOLOR overrides the FILLCOLOR attribute. |

| | FILLSTYLE | style | Style for filling the interior of frame: |
|---|---|---|---|
| | | | NONE (default)<br>SOLID Solid color<br>BDIAGONAL Downward hatch (left to right) at 45 degrees<br>CROSS Horizontal and vertical crosshatch<br>DIAGCROSS Crosshatch at 45 degrees<br>FDIAGONAL Upward hatch (left to right) at 45 degrees<br>HORIZONTAL Horizontal hatch<br>VERTICAL Vertical hatch |
| | SUBSTSIGN | substitute sign | Character that is used as substitute sign when a character in a read field cannot be read |
| | TITLE | fieldname* | Uses the field with the name <fieldname> as the title of the frame. Positioning information of the field is ignored. |
| | HORIZONTAL | justify | Horizontal alignment of the frame title:<br>LEFT (default)<br>CENTER<br>RIGHT |
| | VERTICAL | justify | Vertical alignment of the frame title:<br>TOP (default)<br>BOTTOM |
| END | | | |

The **XFSFRAME** definition provides a means for framing a **XFSFIELD** text field. The basic concept of a **XFSFRAME** definition and corresponding **XFSFIELD** definition is illustrated as follows:

```
      ┌─────────────────────┐
┌─────┤   Account Owner     ├─────┐
│╔════╧═════════════════════╧════╗│
│║ Mr/Mrs Jean Leroy             ║│
│║ 21560 Hagerty Road            ║│
│║ Troy, MI.                     ║│
│╚═══════════════════════════════╝│
└─────────────────────────────────┘
```
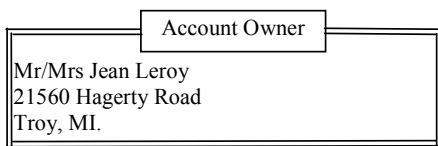
When the **XFSFRAME** frames a field, its positioning and size information are ignored. Instead, Service Providers should position the top left corner of the frame one horizontal base unit to the left and one vertical base unit to the top of the top left corner of the field. Similarly, Service Providers should size the frame so that it bottom right corner is one base unit below and to the right to the field. For instance, if the form units are **ROWCOLUMN**, and a **XFSFRAME** "A" is said to frame the **XFSFIELD** "B" which is positioned at row 1, column 1 with a size of 1 row and 20 columns, the frame will be drawn from row 0, column 0 to row 3, column 22.

The horizontal and vertical positioning of a frame title overrides the position of the named **XFSFIELD**. For instance, if a **XFSFRAME** "A" is said to have the **XFSFIELD** "B" as its title, with the default horizontal and vertical title justification, it is just as if **XFSFIELD** "B" had been positioned at the top left corner of the frame. Note that the **SIZE** information for the title field still is meaningful; it gives the starting and/or ending positions of the frame lines.

The **SIDE** attributes of the **XFSFRAME** and the **XFSFIELD**s it refers to must agree.

The width of the lines and the interval between the lines of doubled frames are vendor specific. Whether the lines are drawn using graphics printing or using pseudo-graphic is vendor specific. However, Service Providers are responsible for rendering intersecting frames.

Depending on the printer technology, framing of fields can substantially slow down the print process.

Support of framing by a Service Provider or the device it controls is not mandatory to be XFS compliant.

## Sample 1: Simple framing

```
XFSFORM "Multiple Balances"

BEGIN
   UNIT INCH, 16, 16
   SIZE 91, 64
   VERSION 1, 0, "13/09/96", "XFS"
   LANGUAGE 0x0409
XFSFIELD "Account Title"
   BEGIN
      POSITION 15, 4
      SIZE 30, 4

      CLASS STATIC

      HORIZONTAL CENTER

      INITIALVALUE "Account"
END
XFSFIELD "Balance Title"

BEGIN
   POSITION 45, 4
   SIZE 30, 4
   CLASS STATIC
   HORIZONTAL CENTER
   INITIALVALUE "Balance"
```

When printed with the following field list:
Account[0]=0123456789123001
Account[1]=0123456789123002
Account[2]=0123456789123003
Balance[0]=$17465.12
Balance[1] =$2458.23
Balance[2]=$6542.78
Will print:

| Account | Balance |
|---|---|
| 012345678912300 1 | $17465.12 |
| 012345678912300 2 | $2458.23 |
| 012345678912300 3 | $6542.78 |

When printed with the following field list:
Account[0]=0123456789123001
Balance[0]=$17465.12
Will print:

| Account | Balance |
|---|---|
| 012345678912300 1 | $17465.12 |

```
    END

    XFSFIELD "Account"
    BEGIN
      POSITION 15, 8
      SIZE 30, 4
      INDEX 10, 0, 3
    END //"Account"
    XFSFIELD "Balance"
    BEGIN
      POSITION 45, 8
      SIZE 30, 4
      INDEX 10, 0, 3
      HORIZONTAL RIGHT
    END //"Balance"
    XFSFRAME "Account Title"
    BEGIN
      POSITION 15, 4
      FRAMES "Account Title"
      SIZE 30, 4
      STYLE DOUBLE_THIN
    END
    XFSFRAME "Balance Title"
    BEGIN
      POSITION 45, 4
      FRAMES "Balance Title"
      SIZE 30, 4
      STYLE DOUBLE_THIN
    END
    XFSFRAME "Account"
    BEGIN
      POSITION 15, 8
      FRAMES "Account"
      SIZE 30, 34
      STYLE DOUBLE_THIN
    END
    XFSFRAME "Balance"
    BEGIN
      POSITION 45, 8
      FRAMES "Balance"
      SIZE 30, 34
      STYLE DOUBLE_THIN
    END
  END
```

## Sample 2: Framing with title

```
  XFSFORM "Bank Details"

  BEGIN
    UNIT INCH, 16, 16
    SIZE 121, 64
    VERSION 1, 0, "13/09/96", "XFS Editor"
    LANGUAGE 0x0409
    XFSFIELD "Owner Frame Title"
    BEGIN
      POSITION 24, 9
      SIZE 27, 3

      CLASS STATIC

      HORIZONTAL CENTER
```

When printed with the following
field list:
Owner = Mr/Mrs Jean Leroy
            21560 Hagerty Road
            Troy, MI.

will print:

```
  ┌──────── Account Owner ────────┐
  │Mr/Mrs Jean                    
  │Leroy                          
  │21560 Hagerty                  
  │Road                           
  │Troy, MI.                      
```

```
    VERTICAL CENTER
    INITIALVALUE "Account Owner"
  END
  XFSFIELD "Owner"
  BEGIN
    POSITION 20, 11
    SIZE 35, 9
    CLASS REQUIRED
    VERTICAL TOP
  END //"Owner"
  XFSFRAME "Owner Frame"
  BEGIN
    POSITION 19, 10
    FRAMES "Owner"
    SIZE 37, 11
    TITLE "Owner Frame Title"
    HORIZONTAL CENTER
  END
END
```

## Sample 3: Framing with filled interior

```
XFSFORM "Bank Details"

BEGIN
  UNIT INCH, 16, 16
  SIZE 121, 64
  VERSION 1, 0, "13/09/96", "XFS Editor"
  LANGUAGE 0x0409
  XFSFIELD "Owner"
  BEGIN
    POSITION 20, 11
    SIZE 35, 9
    CLASS REQUIRED

    VERTICAL TOP

  END
  XFSFRAME "Owner Frame"
  BEGIN
    POSITION 19, 10
    FRAMES "Owner"
    SIZE 37, 11
    FILLCOLOR GRAY
    FILLSTYLE CROSS
  END
END
```

When printed with the following
field list:
Owner = Mr/Mrs Jean Leroy
        21560 Hagerty Road
        Troy, MI.

will print:



## Sample 4: Repeated Framing

```
XFSFORM "Smart Account Number"

BEGIN
  UNIT INCH, 16, 16
  SIZE 121, 64
  VERSION 1, 0, "13/09/96", "XFS Editor"
  LANGUAGE 0x0409
  XFSFIELD "Account Number"
  BEGIN
    POSITION 20, 8
    SIZE 4, 4
    INDEX 12, 4, 0
```

When printed with the following
field list:
Account Number[0]=0
Account Number[1]=1
Account Number[2]=2
Account Number[3]=3
Account Number[4]=4
Account Number[5]=5
Account Number[6]=6
Account Number[7]=7
Account Number[8]=8
Account Number[9]=9

```
    HORIZONTAL CENTER
    VERTICAL CENTER
  END
  XFSFRAME "A/N Frame"
  BEGIN
    POSITION 20, 8
    SIZE 4, 4
    REPEATONX 12, 4
  END
END
```

Account Number[10]=0
Account Number[11]=1


*will print*

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|

## 10.7 Media Definition [5]

The media definition determines those characteristics that result from the combination of a particular media type together with a particular vendor's printer. The aim is to make it easy to move forms between different vendors' printers which might have different constraints on how they handle a specific media type. It is the Service Provider's responsibility to ensure that the form definition does not specify the printing of any fields that conflict with the media definition. An example of such a conflict might be that the form definition asks for a field to be printed in an area that the media definition defines as an unprintable area.

The media definition is also intended to provide the capability of defining media types that are specific to the financial industry. An example is a passbook as shown below.

**Passbook with horizontal fold**        **Passbook with vertical fold**
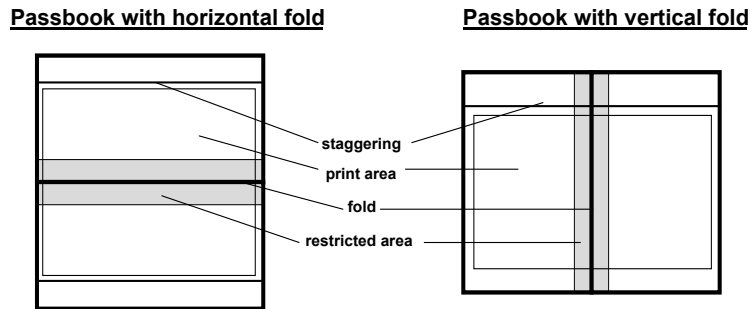


| XFSMEDIA | | medianame* | |
|---|---|---|---|
| BEGIN | | | |
| | TYPE | type | Predefined media types are: |
| | | |     GENERIC (default) |
| | | |     MULTIPART |
| | | |     PASSBOOK |
| | SOURCE | source | Paper source: |
| | | |     ANY    (default) |
| | | |     UPPER |
| | | |     LOWER |
| | | |     EXTERNAL    (envelope tray or single sheet feed tray) |
| | | |     AUX |
| | | |     AUX2 |
| | | |     PARK |
| (required) | UNIT | base, | Base resolution unit for media definition: |
| | | |     MM |
| | | |     INCH |
| | | |     ROWCOLUMN |
| | | x, | Horizontal base unit fraction |
| | | y | Vertical base unit fraction |
| (required) | SIZE | width, | Width of physical media |
| | | height | Height of physical media (0 = unlimited, i.e. roll paper) |
| | PRINTAREA | x, | Printable area relative to top left corner |
| | | y, | of physical media |
| | | width, | (default = physical size of media) |
| | | height | |
| | RESTRICTED | x, | Restricted area relative to top left corner |
| | | y, | of physical media |
| | | width, | (default = no restricted area) |
| | | height | |
| | FOLD | fold | Type of passbook: |
| | | |     HORIZONTAL    (default) |
| | | |     VERTICAL |

---

[5] Attributes are not required in any mandatory order within a Media definition.

| | STAGGERING | *staggering* | Staggering of passbook from top (default = 0) |
|---|---|---|---|
| | **PAGE** | *count* | Number of pages in passbook (default = 0) |
| | **LINES** | *count* | Number of printable lines (default = 0) |
| **END** | | | |

## 10.8 XFS Form/Media Definition Files in Multi-Vendor Environments

Although for most Service Providers directory location and extension of XFS form/media definition files are configurable through the registry, the capabilities of Service Providers and or actual hardware may vary. Therefore the following considerations should be taken into account when applications use XFS form definition files with the purpose of running in a multi-vendor environment:

- Physical print area dimensions of printers are not identical.

- Graphic printout may not be supported, which may limit the use of the FONT, CPI and LPI keywords.

- Some printers may have a resolution of dots/mm rather than dots/inch, which may result in printouts with a specific CPI/LPI font resolution to be slightly off size.

- Some form/media definition keywords may not be supported due to limitations of the hardware or software.

**Deleted:** <#>Just-in-time form loading may not be supported by all Service Providers, which makes it impossible to create dynamic form files just before printing (which in return means that only the print data of the forms can be changed, not the -layout data such as the font and font size) ¶

# 11. Command and Event Flows during Single and Multi Page / Wad Printing

It is possible to print a number of pages or bunches of pages (wads) through the XFS Service Provider. The following sections describe how this is achieved.

## 11.1 Single Page / Single Wad Printing with immediate Media Control

This table illustrates the command and event flows in a successful print command (i.e. WFS_CMD_PTR_PRINT_RAW_FILE, WFS_CMD_PTR_PRINT_FORM and WFS_CMD_PTR_RAW_DATA) where the following conditions apply:

- A single page or single wad of pages is presented.

- The *bMediaPresented* Capability flag is TRUE (indicates that the WFS_EXEE_PTR_MEDIAPRESENTED event can be generated).

- The *dwMediaControl* flag in the command data is set to WFS_PTR_CTRLEJECT.

The WFS_CMD_PTR_PRINT_RAW_FILE command is used as an example.

| Step | End-User | Application | XFS Service Provider | PTR Hardware |
|------|----------|-------------|----------------------|--------------|
| 1. | User wants a statement printed. | | | |
| 2. | | Win32 used to print statement to a native printer file. | | |
| 3. | | WFS_CMD_PTR_PRINT_-RAW_FILE command issued (with *dwMediaControl* set to WFS_PTR_CTRLEJECT). | | |
| 4. | | | One wad or page is required. | |
| 5. | | | | Wad or page presented. |
| 6. | | | WFS_EXEE_PTR_MEDIA-PRESENTED event generated. | |
| 7. | | WFS_CMD_PTR_PRINT_-RAW_FILE completes successfully. | | |
| 8. | User takes wad/page. | | | |
| 9. | | | WFS_SRVE_PTR_MEDIATAKEN event generated. | |

## 11.2 Single Page / Single Wad Printing with separate Media Control

This table illustrates the command and event flows in a successful print command (i.e. WFS_CMD_PTR_PRINT_RAW_FILE, WFS_CMD_PTR_PRINT_FORM and WFS_CMD_PTR_RAW_DATA) where the following conditions apply:

- A single page or single wad of pages is presented.

- The *bMediaPresented* Capability flag is TRUE (indicates that the WFS_EXEE_PTR_MEDIAPRESENTED event can be generated).

- The *dwMediaControl* flag in the command data is set to zero.

- The media is presented to the user through a WFS_CMD_PTR_CONTROL_MEDIA command, with the *lpdwMediaControl* flag set to WFS_PTR_CTRLEJECT.

The WFS_CMD_PTR_PRINT_RAW_DATA command is used as an example.

| Step | End-User | Application | XFS Service Provider | PTR Hardware |
|------|----------|-------------|----------------------|--------------|
| 1. | User wants a statement printed. | | | |
| 2. | | Win32 used to print statement to a native printer file. | | |
| 3. | | WFS_CMD_PTR_PRINT_RAW_FILE command issued (with *dwMediaCotrol* set to zero). | | |
| 4. | | WFS_CMD_PTR_PRINT_RAW_FILE completes successfully. | | |
| 5. | | WFS_CMD_PTR_CONTROL_MEDIA ( with *lpdwMediaControl* set to WFS_PTR_CTRLEJECT). | | |
| 6. | | | One wad or page is required. | |
| 7. | | | | Wad or page presented. |
| 8. | | | WFS_EXEE_PTR_MEDIAPRESENTED event generated. | |
| 9. | | WFS_CMD_PTR_CONTROL_MEDIA completes successfully. | | |
| 10. | User takes wad/page. | | | |
| 11. | | | WFS_SRVE_PTR_MEDIATAKEN event generated. | |

## 11.3 Multi Page / Multi Wad Printing with immediate Media Control

This table illustrates a successful WFS_CMD_PTR_PRINT_RAW_FILE command where multiple page / wads are presented (and the *bMediaPresented* Capability flag indicates that the WFS_EXEE_PTR_MEDIAPRESENTED event can be generated). In addition, the previous page/wad must be removed before subsequent pages/wads can be printed.

This table illustrates the command and event flows in a successful print command (i.e. WFS_CMD_PTR_PRINT_RAW_FILE, WFS_CMD_PTR_PRINT_FORM and WFS_CMD_PTR_RAW_DATA) where the following conditions apply:

- Multiple pages or multiple wads of pages are presented.

- The *bMediaPresented* Capability flag is TRUE (indicates that the WFS_EXEE_PTR_MEDIAPRESENTED event can be generated).

- The *dwMediaControl* flag in the command data is set to WFS_PTR_CTRLEJECT.

- The previous page/wad must be removed before subsequent pages/wads can be presented.

The WFS_CMD_PTR_PRINT_RAW_FILE command is used as an example.

| Step | End-User | Application | XFS Service Provider | PTR Hardware |
|------|----------|-------------|----------------------|--------------|
| 1. | User wants a statement printed. | | | |
| 2. | | Win32 used to print statement to a native printer file. | | |
| 3. | | WFS_CMD_PTR_PRINT_-RAW_FILE command issued (with *dwMediaControl* set to WFS_PTR_CTRLEJECT). | | |
| 4. | | | Three wads or pages are required. | |
| 5. | | | | First wad or page presented. |
| 6. | | | WFS_EXEE_PTR_MEDIA-PRESENTED event generated. | |
| 7. | User takes wad/page. | | | |
| 8. | | | WFS_SRVE_PTR_MEDIATAKEN event generated. | |
| 9. | | | | Second wad or page presented. |
| 10. | | | WFS_EXEE_PTR_MEDIA-PRESENTED event generated. | |
| 11. | User takes wad/page. | | | |
| 12. | | | WFS_SRVE_PTR_MEDIATAKEN event generated. | |

| Step | End-User | Application | XFS Service Provider | PTR Hardware |
|------|----------|-------------|----------------------|--------------|
| 13. | | | | Final wad or page presented. |
| 14. | | | WFS_EXEE_PTR_MEDIA-PRESENTED event generated. | |
| 15. | | WFS_CMD_PTR_PRINT_-RAW_FILE completes successfully. | | |
| 16. | User takes wad/page. | | | |
| 17. | | | WFS_SRVE_PTR_MEDIATAKEN event generated. | |

## 11.4 Multi Page / Multi Wad Printing with separate Media Control

This table illustrates the command and event flows in a successful print command (i.e. WFS_CMD_PTR_PRINT_RAW_FILE, WFS_CMD_PTR_PRINT_FORM and WFS_CMD_PTR_RAW_DATA) where the following conditions apply:

- Multiple pages or multiple wads of pages are presented.

- The *bMediaPresented* Capability flag is TRUE (indicates that the WFS_EXEE_PTR_MEDIAPRESENTED event can be generated).

- The *dwMediaControl* flag in the command data is set to zero.

- The media is presented to the user through a WFS_CMD_PTR_CONTROL_MEDIA command, with the *lpdwMediaControl* flag set to WFS_PTR_CTRLEJECT.

- The previous page/wad must be removed before subsequent pages/wads can be presented.

The WFS_CMD_PTR_PRINT_FORM command is used as a specific example.

| Step | End-User | Application | XFS Service Provider | PTR Hardware |
|---|---|---|---|---|
| 1. | User wants a statement printed. | | | |
| 2. | | WFS_CMD_PTR_PRINT_-FORM command issued (with *dwMediaCotrol* set to zero). | | |
| 3. | | WFS_CMD_PTR_PRINT_-FORM completes successfully. | | |
| 4. | | WFS_CMD_PTR_-CONTROL_MEDIA (with *lpdwMediaControl* set to WFS_PTR_CTRLEJECT). | | |
| 5. | | | Three wads or pages are required. | |
| 6. | | | | First wad or page presented. |
| 7. | | | WFS_EXEE_PTR_MEDIA-PRESENTED event generated. | |
| 8. | User takes wad/page. | | | |
| 9. | | | WFS_SRVE_PTR_MEDIATAKEN event generated. | |
| 10. | | | | Second wad or page presented. |
| 11. | | | WFS_EXEE_PTR_MEDIA-PRESENTED event generated. | |
| 12. | User takes wad/page. | | | |
| 13. | | | WFS_SRVE_PTR_MEDIATAKEN event generated. | |

| Step | End-User | Application | XFS Service Provider | PTR Hardware |
|------|----------|-------------|----------------------|--------------|
| 14. | | | | Final wad or page presented. |
| 15. | | | WFS_EXEE_PTR_MEDIA-PRESENTED event generated. | |
| 16. | | WFS_CMD_PTR_-CONTROL_MEDIA completes successfully. | | |
| 17. | User takes wad/page. | | | |
| 18. | | | WFS_SRVE_PTR_MEDIATAKEN event generated. | |

## 11.5 Printing with immediate Media Control and *bMediaPresented* == FALSE

This table illustrates the command and event flows in a successful print command (i.e.
WFS_CMD_PTR_PRINT_RAW_FILE, WFS_CMD_PTR_PRINT_FORM and WFS_CMD_PTR_RAW_DATA)
where the following conditions apply:

- One or more pages or wads of pages is presented (it is the same flow for one or a number of pages).

- The *bMediaPresented* Capability flag is FALSE (indicates that the
  WFS_EXEE_PTR_MEDIAPRESENTED event cannot be generated).

- The *dwMediaControl* flag in the command data is set to WFS_PTR_CTRLEJECT.

The WFS_CMD_PTR_PRINT_RAW_FILE command is used as an example.

| Step | End-User | Application | XFS Service Provider | PTR Hardware |
|---|---|---|---|---|
| 1. | User wants a statement printed. | | | |
| 2. | | Win32 used to print statement to a native printer file. | | |
| 3. | | WFS_CMD_PTR_PRINT_- RAW_FILE command issued (with *dwMediaControl* set to WFS_PTR_CTRLEJECT). | | |
| 4. | | | One or more wads/pages are required. | |
| 5. | | | | One or more wad or page presented. |
| 7. | | WFS_CMD_PTR_PRINT_- RAW_FILE completes successfully. | | |
| 8. | User takes wads/pages. | | | |
| 9. | | | WFS_SRVE_PTR_MEDIATAKEN event generated. | |

# 12. C-Header File

```
/***************************************************************************
*                                                                         *
* xfsptr.h    XFS - Banking Printer (PTR) definitions                     *
*               (receipt, journal, passbook and document printer)         *
*                                                                         *
*               Version 3.10 (29/11/2007)                                 *
*                                                                         *
***************************************************************************/

#ifndef __INC_XFSPTR__H
#define __INC_XFSPTR__H

#ifdef __cplusplus
extern "C" {
#endif

#include <xfsapi.h>

/*   be aware of alignment   */
#pragma pack(push,1)


/* value of WFSPTRCAPS.wClass */

#define     WFS_SERVICE_CLASS_PTR               (1)
#define     WFS_SERVICE_CLASS_VERSION_PTR       (0x0A03) /* Version 3.10 */
#define     WFS_SERVICE_CLASS_NAME_PTR          "PTR"

#define     PTR_SERVICE_OFFSET                  (WFS_SERVICE_CLASS_PTR * 100)


/* PTR Info Commands */

#define     WFS_INF_PTR_STATUS                  (PTR_SERVICE_OFFSET + 1)
#define     WFS_INF_PTR_CAPABILITIES            (PTR_SERVICE_OFFSET + 2)
#define     WFS_INF_PTR_FORM_LIST               (PTR_SERVICE_OFFSET + 3)
#define     WFS_INF_PTR_MEDIA_LIST              (PTR_SERVICE_OFFSET + 4)
#define     WFS_INF_PTR_QUERY_FORM              (PTR_SERVICE_OFFSET + 5)
#define     WFS_INF_PTR_QUERY_MEDIA             (PTR_SERVICE_OFFSET + 6)
#define     WFS_INF_PTR_QUERY_FIELD             (PTR_SERVICE_OFFSET + 7)
#define     WFS_INF_PTR_CODELINE_MAPPING        (PTR_SERVICE_OFFSET + 8)


/* PTR Execute Commands */

#define     WFS_CMD_PTR_CONTROL_MEDIA           (PTR_SERVICE_OFFSET + 1)
#define     WFS_CMD_PTR_PRINT_FORM              (PTR_SERVICE_OFFSET + 2)
#define     WFS_CMD_PTR_READ_FORM               (PTR_SERVICE_OFFSET + 3)
#define     WFS_CMD_PTR_RAW_DATA                (PTR_SERVICE_OFFSET + 4)
#define     WFS_CMD_PTR_MEDIA_EXTENTS           (PTR_SERVICE_OFFSET + 5)
#define     WFS_CMD_PTR_RESET_COUNT             (PTR_SERVICE_OFFSET + 6)
#define     WFS_CMD_PTR_READ_IMAGE              (PTR_SERVICE_OFFSET + 7)
#define     WFS_CMD_PTR_RESET                   (PTR_SERVICE_OFFSET + 8)
#define     WFS_CMD_PTR_RETRACT_MEDIA           (PTR_SERVICE_OFFSET + 9)
#define     WFS_CMD_PTR_DISPENSE_PAPER          (PTR_SERVICE_OFFSET + 10)
#define     WFS_CMD_PTR_SET_GUIDANCE_LIGHT      (PTR_SERVICE_OFFSET + 11)
#define     WFS_CMD_PTR_PRINT_RAW_FILE          (PTR_SERVICE_OFFSET + 12)
#define     WFS_CMD_PTR_LOAD_DEFINITION         (PTR_SERVICE_OFFSET + 13)
#define     WFS_CMD_PTR_SUPPLY_REPLENISH        (PTR_SERVICE_OFFSET + 14)
#define     WFS_CMD_PTR_POWER_SAVE_CONTROL      (PTR_SERVICE_OFFSET + 15)


/* PTR Messages */

#define     WFS_EXEE_PTR_NOMEDIA                (PTR_SERVICE_OFFSET + 1)
#define     WFS_EXEE_PTR_MEDIAINSERTED          (PTR_SERVICE_OFFSET + 2)
#define     WFS_EXEE_PTR_FIELDERROR             (PTR_SERVICE_OFFSET + 3)
```

Deleted: (0x0003

```
#define      WFS_EXEE_PTR_FIELDWARNING        (PTR_SERVICE_OFFSET + 4)
#define      WFS_USRE_PTR_RETRACTBINTHRESHOLD (PTR_SERVICE_OFFSET + 5)
#define      WFS_SRVE_PTR_MEDIATAKEN          (PTR_SERVICE_OFFSET + 6)
#define      WFS_USRE_PTR_PAPERTHRESHOLD      (PTR_SERVICE_OFFSET + 7)
#define      WFS_USRE_PTR_TONERTHRESHOLD      (PTR_SERVICE_OFFSET + 8)
#define      WFS_SRVE_PTR_MEDIAINSERTED       (PTR_SERVICE_OFFSET + 9)
#define      WFS_USRE_PTR_LAMPTHRESHOLD       (PTR_SERVICE_OFFSET + 10)
#define      WFS_USRE_PTR_INKTHRESHOLD        (PTR_SERVICE_OFFSET + 11)
#define      WFS_SRVE_PTR_MEDIADETECTED       (PTR_SERVICE_OFFSET + 12)
#define      WFS_SRVE_PTR_RETRACTBINSTATUS    (PTR_SERVICE_OFFSET + 13)
#define      WFS_EXEE_PTR_MEDIAPRESENTED      (PTR_SERVICE_OFFSET + 14)
#define      WFS_SRVE_PTR_DEFINITIONLOADED    (PTR_SERVICE_OFFSET + 15)
#define      WFS_EXEE_PTR_MEDIAREJECTED       (PTR_SERVICE_OFFSET + 16)
#define      WFS_SRVE_PTR_MEDIAPRESENTED      (PTR_SERVICE_OFFSET + 17)
#define      WFS_SRVE_PTR_MEDIAAUTORETRACTED  (PTR_SERVICE_OFFSET + 18)
#define      WFS_SRVE_PTR_DEVICEPOSITION      (PTR_SERVICE_OFFSET + 19)
#define      WFS_SRVE_PTR_POWER_SAVE_CHANGE   (PTR_SERVICE_OFFSET + 20)


/* values of WFSPTRSTATUS.fwDevice */

#define      WFS_PTR_DEVONLINE                WFS_STAT_DEVONLINE
#define      WFS_PTR_DEVOFFLINE               WFS_STAT_DEVOFFLINE
#define      WFS_PTR_DEVPOWEROFF              WFS_STAT_DEVPOWEROFF
#define      WFS_PTR_DEVNODEVICE              WFS_STAT_DEVNODEVICE
#define      WFS_PTR_DEVHWERROR               WFS_STAT_DEVHWERROR
#define      WFS_PTR_DEVUSERERROR             WFS_STAT_DEVUSERERROR
#define      WFS_PTR_DEVBUSY                  WFS_STAT_DEVBUSY
#define      WFS_PTR_DEVFRAUDATTEMPT          WFS_STAT_DEVFRAUDATTEMPT


/* values of WFSPTRSTATUS.fwMedia and
           WFSPTRMEDIADETECTED.wPosition */

#define      WFS_PTR_MEDIAPRESENT             (0)
#define      WFS_PTR_MEDIANOTPRESENT          (1)
#define      WFS_PTR_MEDIAJAMMED              (2)
#define      WFS_PTR_MEDIANOTSUPP             (3)
#define      WFS_PTR_MEDIAUNKNOWN             (4)
#define      WFS_PTR_MEDIAENTERING            (5)
#define      WFS_PTR_MEDIARETRACTED           (6)

/* additional values for WFSPTRMEDIADETECTED.wPosition */
#define      WFS_PTR_MEDIAEXPELLED            (7)


/* Size and max index of fwPaper array */

#define      WFS_PTR_SUPPLYSIZE               (16)
#define      WFS_PTR_SUPPLYMAX                (WFS_PTR_SUPPLYSIZE - 1)


/* Indices of WFSPTRSTATUS.fwPaper [...] */

#define      WFS_PTR_SUPPLYUPPER              (0)
#define      WFS_PTR_SUPPLYLOWER              (1)
#define      WFS_PTR_SUPPLYEXTERNAL           (2)
#define      WFS_PTR_SUPPLYAUX                (3)
#define      WFS_PTR_SUPPLYAUX2               (4)
#define      WFS_PTR_SUPPLYPARK               (5)


/* values of WFSPTRSTATUS.fwPaper and
           WFSPTRPAPERTHRESHOLD.wPaperThreshold */

#define      WFS_PTR_PAPERFULL                (0)
#define      WFS_PTR_PAPERLOW                 (1)
#define      WFS_PTR_PAPEROUT                 (2)
#define      WFS_PTR_PAPERNOTSUPP             (3)
#define      WFS_PTR_PAPERUNKNOWN             (4)
```

```
#define      WFS_PTR_PAPERJAMMED              (5)


/* values of WFSPTRSTATUS.fwToner */

#define      WFS_PTR_TONERFULL                (0)
#define      WFS_PTR_TONERLOW                 (1)
#define      WFS_PTR_TONEROUT                 (2)
#define      WFS_PTR_TONERNOTSUPP             (3)
#define      WFS_PTR_TONERUNKNOWN             (4)


/* values of WFSPTRSTATUS.fwInk */

#define      WFS_PTR_INKFULL                  (0)
#define      WFS_PTR_INKLOW                   (1)
#define      WFS_PTR_INKOUT                   (2)
#define      WFS_PTR_INKNOTSUPP               (3)
#define      WFS_PTR_INKUNKNOWN               (4)


/* values of WFSPTRSTATUS.fwLamp */

#define      WFS_PTR_LAMPOK                   (0)
#define      WFS_PTR_LAMPFADING               (1)
#define      WFS_PTR_LAMPINOP                 (2)
#define      WFS_PTR_LAMPNOTSUPP              (3)
#define      WFS_PTR_LAMPUNKNOWN              (4)


/* values of WFSPTRRETRACTBINS.wRetractBin and
             WFSPTRBINTHRESHOLD.wRetractBin */

#define      WFS_PTR_RETRACTBINOK             (0)
#define      WFS_PTR_RETRACTBINFULL           (1)
#define      WFS_PTR_RETRACTNOTSUPP           (2)
#define      WFS_PTR_RETRACTUNKNOWN           (3)
#define      WFS_PTR_RETRACTBINHIGH           (4)


/* additional values of WFSPTRSTATUS.fwRetractBin */

#define      WFS_PTR_RETRACTBINMISSING        (5)


/* Size and max index of dwGuidLights array */

#define      WFS_PTR_GUIDLIGHTS_SIZE          (32)
#define      WFS_PTR_GUIDLIGHTS_MAX           (WFS_PTR_GUIDLIGHTS_SIZE - 1)

/* Indices of WFSPTRSTATUS.dwGuidLights [...]
             WFSPTRCAPS.dwGuidLights [...] */

#define      WFS_PTR_GUIDANCE_PRINTER         (0)


/* Values of WFSPTRSTATUS.dwGuidLights [...]
             WFSPTRCAPS.dwGuidLights [...] */

#define      WFS_PTR_GUIDANCE_NOT_AVAILABLE   (0x00000000)
#define      WFS_PTR_GUIDANCE_OFF             (0x00000001)
#define      WFS_PTR_GUIDANCE_SLOW_FLASH      (0x00000004)
#define      WFS_PTR_GUIDANCE_MEDIUM_FLASH    (0x00000008)
#define      WFS_PTR_GUIDANCE_QUICK_FLASH     (0x00000010)
#define      WFS_PTR_GUIDANCE_CONTINUOUS      (0x00000080)
#define      WFS_PTR_GUIDANCE_RED             (0x00000100)
#define      WFS_PTR_GUIDANCE_GREEN           (0x00000200)
#define      WFS_PTR_GUIDANCE_YELLOW          (0x00000400)
#define      WFS_PTR_GUIDANCE_BLUE            (0x00000800)
#define      WFS_PTR_GUIDANCE_CYAN            (0x00001000)
```

```
#define      WFS_PTR_GUIDANCE_MAGENTA           (0x00002000)
#define      WFS_PTR_GUIDANCE_WHITE             (0x00004000)


/* values of WFSPTRSTATUS.wDevicePosition
              WFSPTRDEVICEPOSITION.wPosition */

#define      WFS_PTR_DEVICEINPOSITION          (0)
#define      WFS_PTR_DEVICENOTINPOSITION       (1)
#define      WFS_PTR_DEVICEPOSUNKNOWN          (2)
#define      WFS_PTR_DEVICEPOSNOTSUPP          (3)

/* values of WFSPTRCAPS.fwType */

#define      WFS_PTR_TYPERECEIPT               (0x0001)
#define      WFS_PTR_TYPEPASSBOOK              (0x0002)
#define      WFS_PTR_TYPEJOURNAL               (0x0004)
#define      WFS_PTR_TYPEDOCUMENT              (0x0008)
#define      WFS_PTR_TYPESCANNER               (0x0010)


/* values of WFSPTRCAPS.wResolution,
              WFSPTRPRINTFORM.wResolution */

#define      WFS_PTR_RESLOW                    (0x0001)
#define      WFS_PTR_RESMED                    (0x0002)
#define      WFS_PTR_RESHIGH                   (0x0004)
#define      WFS_PTR_RESVERYHIGH               (0x0008)


/* values of WFSPTRCAPS.fwReadForm */

#define      WFS_PTR_READOCR                   (0x0001)
#define      WFS_PTR_READMICR                  (0x0002)
#define      WFS_PTR_READMSF                   (0x0004)
#define      WFS_PTR_READBARCODE               (0x0008)
#define      WFS_PTR_READPAGEMARK              (0x0010)
#define      WFS_PTR_READIMAGE                 (0x0020)
#define      WFS_PTR_READEMPTYLINE             (0x0040)


/* values of WFSPTRCAPS.fwWriteForm */

#define      WFS_PTR_WRITETEXT                 (0x0001)
#define      WFS_PTR_WRITEGRAPHICS             (0x0002)
#define      WFS_PTR_WRITEOCR                  (0x0004)
#define      WFS_PTR_WRITEMICR                 (0x0008)
#define      WFS_PTR_WRITEMSF                  (0x0010)
#define      WFS_PTR_WRITEBARCODE              (0x0020)
#define      WFS_PTR_WRITESTAMP                (0x0040)


/* values of WFSPTRCAPS.fwExtents */

#define      WFS_PTR_EXTHORIZONTAL             (0x0001)
#define      WFS_PTR_EXTVERTICAL               (0x0002)


/* values of WFSPTRCAPS.fwControl, dwMediaControl */

#define      WFS_PTR_CTRLEJECT                 (0x0001)
#define      WFS_PTR_CTRLPERFORATE             (0x0002)
#define      WFS_PTR_CTRLCUT                   (0x0004)
#define      WFS_PTR_CTRLSKIP                  (0x0008)
#define      WFS_PTR_CTRLFLUSH                 (0x0010)
#define      WFS_PTR_CTRLRETRACT               (0x0020)
#define      WFS_PTR_CTRLSTACK                 (0x0040)
#define      WFS_PTR_CTRLPARTIALCUT            (0x0080)
#define      WFS_PTR_CTRLALARM                 (0x0100)
#define      WFS_PTR_CTRLATPFORWARD            (0x0200)
```

```
#define     WFS_PTR_CTRLATPBACKWARD        (0x0400)
#define     WFS_PTR_CTRLTURNMEDIA          (0x0800)
#define     WFS_PTR_CTRLSTAMP              (0x1000)
#define     WFS_PTR_CTRLPARK              (0x2000)
#define     WFS_PTR_CTRLEXPEL              (0x4000)
#define     WFS_PTR_CTRLEJECTTOTRANSPORT   (0x8000)


/* values of WFSPTRCAPS.fwPaperSources,
             WFSFRMMEDIA.wPaperSources,
             WFSPTRPRINTFORM.wPaperSource and
             WFSPTRPAPERTHRESHOLD.wPaperSource    */

#define     WFS_PTR_PAPERANY              (0x0001)
#define     WFS_PTR_PAPERUPPER            (0x0002)
#define     WFS_PTR_PAPERLOWER            (0x0004)
#define     WFS_PTR_PAPEREXTERNAL         (0x0008)
#define     WFS_PTR_PAPERAUX              (0x0010)
#define     WFS_PTR_PAPERAUX2             (0x0020)
#define     WFS_PTR_PAPERPARK             (0x0040)


/* values of WFSPTRCAPS.fwImageType,
             WFSPTRIMAGEREQUEST.wFrontImageFormat and
             WFSPTRIMAGEREQUEST.wBackImageFormat */

#define     WFS_PTR_IMAGETIF              (0x0001)
#define     WFS_PTR_IMAGEWMF              (0x0002)
#define     WFS_PTR_IMAGEBMP              (0x0004)
#define     WFS_PTR_IMAGEJPG              (0x0008)


/* values of WFSPTRCAPS.fwFrontImageColorFormat,
             WFSPTRCAPS.fwBackImageColorFormat,
             WFSPTRIMAGEREQUEST.wFrontImageColorFormat and
             WFSPTRIMAGEREQUEST.wBackImageColorFormat */

#define     WFS_PTR_IMAGECOLORBINARY      (0x0001)
#define     WFS_PTR_IMAGECOLORGRAYSCALE   (0x0002)
#define     WFS_PTR_IMAGECOLORFULL        (0x0004)


/* values of WFSPTRCAPS.fwCodelineFormat and
             WFSPTRIMAGEREQUEST.wCodelineFormat */

#define     WFS_PTR_CODELINECMC7          (0x0001)
#define     WFS_PTR_CODELINEE13B          (0x0002)
#define     WFS_PTR_CODELINEOCR           (0x0004)


/* values of WFSPTRCAPS.fwImageSource,
             WFSPTRIMAGEREQUEST.fwImageSource and
             WFSPTRIMAGE.wImageSource */

#define     WFS_PTR_IMAGEFRONT            (0x0001)
#define     WFS_PTR_IMAGEBACK             (0x0002)
#define     WFS_PTR_CODELINE              (0x0004)


/* values of WFSPTRCAPS.fwCharSupport,
             WFSFRMHEADER.fwCharSupport */

#define     WFS_PTR_ASCII                 (0x0001)
#define     WFS_PTR_UNICODE               (0x0002)


/* values of WFSFRMHEADER.wBase,
             WFSFRMMEDIA.wBase,
             WFSPTRMEDIAUNIT.wBase */
```

```
#define     WFS_FRM_INCH                    (0)
#define     WFS_FRM_MM                      (1)
#define     WFS_FRM_ROWCOLUMN               (2)


/* values of WFSFRMHEADER.wAlignment */

#define     WFS_FRM_TOPLEFT                 (0)
#define     WFS_FRM_TOPRIGHT                (1)
#define     WFS_FRM_BOTTOMLEFT              (2)
#define     WFS_FRM_BOTTOMRIGHT             (3)


/* values of WFSFRMHEADER.wOrientation */

#define     WFS_FRM_PORTRAIT                (0)
#define     WFS_FRM_LANDSCAPE               (1)


/* values of WFSFRMMEDIA.fwMediaType */

#define     WFS_FRM_MEDIAGENERIC            (0)
#define     WFS_FRM_MEDIAPASSBOOK           (1)
#define     WFS_FRM_MEDIAMULTIPART          (2)


/* values of WFSFRMMEDIA.fwFoldType */

#define     WFS_FRM_FOLDNONE                (0)
#define     WFS_FRM_FOLDHORIZONTAL          (1)
#define     WFS_FRM_FOLDVERTICAL            (2)


/* values of WFSFRMFIELD.fwType */

#define     WFS_FRM_FIELDTEXT               (0)
#define     WFS_FRM_FIELDMICR               (1)
#define     WFS_FRM_FIELDOCR                (2)
#define     WFS_FRM_FIELDMSF                (3)
#define     WFS_FRM_FIELDBARCODE            (4)
#define     WFS_FRM_FIELDGRAPHIC            (5)
#define     WFS_FRM_FIELDPAGEMARK           (6)


/* values of WFSFRMFIELD.fwClass */

#define     WFS_FRM_CLASSSTATIC             (0)
#define     WFS_FRM_CLASSOPTIONAL           (1)
#define     WFS_FRM_CLASSREQUIRED           (2)


/* values of WFSFRMFIELD.fwAccess */

#define     WFS_FRM_ACCESSREAD              (0x0001)
#define     WFS_FRM_ACCESSWRITE             (0x0002)


/* values of WFSFRMFIELD.fwOverflow */

#define     WFS_FRM_OVFTERMINATE            (0)
#define     WFS_FRM_OVFTRUNCATE             (1)
#define     WFS_FRM_OVFBESTFIT              (2)
#define     WFS_FRM_OVFOVERWRITE            (3)
#define     WFS_FRM_OVFWORDWRAP             (4)


/* values of WFSPTRFIELDFAIL.wFailure */

#define     WFS_PTR_FIELDREQUIRED           (0)
#define     WFS_PTR_FIELDSTATICOVWR         (1)
```

```
#define      WFS_PTR_FIELDOVERFLOW           (2)
#define      WFS_PTR_FIELDNOTFOUND           (3)
#define      WFS_PTR_FIELDNOTREAD            (4)
#define      WFS_PTR_FIELDNOTWRITE           (5)
#define      WFS_PTR_FIELDHWERROR            (6)
#define      WFS_PTR_FIELDTYPENOTSUPPORTED   (7)
#define      WFS_PTR_FIELDGRAPHIC            (8)
#define      WFS_PTR_CHARSETFORM             (9)


/* values of WFSPTRPRINTFORM.wAlignment */

#define      WFS_PTR_ALNUSEFORMDEFN          (0)
#define      WFS_PTR_ALNTOPLEFT              (1)
#define      WFS_PTR_ALNTOPRIGHT             (2)
#define      WFS_PTR_ALNBOTTOMLEFT           (3)
#define      WFS_PTR_ALNBOTTOMRIGHT          (4)


/* values of WFSPTRPRINTFORM.wOffsetX and
             WFSPTRPRINTFORM.wOffsetY */

#define      WFS_PTR_OFFSETUSEFORMDEFN       (0xffff)


/* values of WFSPTRRAWDATA.wInputData */

#define      WFS_PTR_NOINPUTDATA             (0)
#define      WFS_PTR_INPUTDATA               (1)


/* values of WFSPTRIMAGE.wStatus */

#define      WFS_PTR_DATAOK                  (0)
#define      WFS_PTR_DATASRCNOTSUPP          (1)
#define      WFS_PTR_DATASRCMISSING          (2)


/* values of WFSPTRBINSTATUS.wRetractBin */

#define      WFS_PTR_RETRACTBININSERTED      (1)
#define      WFS_PTR_RETRACTBINREMOVED       (2)


/* values of WFSPTRDEFINITIONLOADED.dwDefinitionType */

#define      WFS_PTR_FORMLOADED              (0x00000001)
#define      WFS_PTR_MEDIALOADED             (0x00000002)


/* values of WFSPTRSUPPLYREPLEN.fwSupplyReplen */

#define      WFS_PTR_REPLEN_PAPERUPPER       (0x0001)
#define      WFS_PTR_REPLEN_PAPERLOWER       (0x0002)
#define      WFS_PTR_REPLEN_PAPERAUX         (0x0004)
#define      WFS_PTR_REPLEN_PAPERAUX2        (0x0008)
#define      WFS_PTR_REPLEN_TONER            (0x0010)
#define      WFS_PTR_REPLEN_INK              (0x0020)
#define      WFS_PTR_REPLEN_LAMP             (0x0040)


/* values of WFSPTRMEDIAREJECTED.wMediaRejected */

#define      WFS_PTR_REJECT_SHORT            (0)
#define      WFS_PTR_REJECT_LONG             (1)
#define      WFS_PTR_REJECT_MULTIPLE         (2)
#define      WFS_PTR_REJECT_ALIGN            (3)
#define      WFS_PTR_REJECT_MOVETOALIGN      (4)
#define      WFS_PTR_REJECT_SHUTTER          (5)
#define      WFS_PTR_REJECT_ESCROW           (6)
```

```
#define    WFS_PTR_REJECT_THICK              (7)
#define    WFS_PTR_REJECT_OTHER              (8)


/* values of WFSPTRMEDIARETRACTED.wRetractResult */

#define    WFS_PTR_AUTO_RETRACT_OK           (0)
#define    WFS_PTR_AUTO_RETRACT_MEDIAJAMMED  (1)


/* XFS PTR Errors */

#define    WFS_ERR_PTR_FORMNOTFOUND       (-(PTR_SERVICE_OFFSET + 0))
#define    WFS_ERR_PTR_FIELDNOTFOUND      (-(PTR_SERVICE_OFFSET + 1))
#define    WFS_ERR_PTR_NOMEDIAPRESENT     (-(PTR_SERVICE_OFFSET + 2))
#define    WFS_ERR_PTR_READNOTSUPPORTED   (-(PTR_SERVICE_OFFSET + 3))
#define    WFS_ERR_PTR_FLUSHFAIL          (-(PTR_SERVICE_OFFSET + 4))
#define    WFS_ERR_PTR_MEDIAOVERFLOW      (-(PTR_SERVICE_OFFSET + 5))
#define    WFS_ERR_PTR_FIELDSPECFAILURE   (-(PTR_SERVICE_OFFSET + 6))
#define    WFS_ERR_PTR_FIELDERROR         (-(PTR_SERVICE_OFFSET + 7))
#define    WFS_ERR_PTR_MEDIANOTFOUND      (-(PTR_SERVICE_OFFSET + 8))
#define    WFS_ERR_PTR_EXTENTNOTSUPPORTED (-(PTR_SERVICE_OFFSET + 9))
#define    WFS_ERR_PTR_MEDIAINVALID       (-(PTR_SERVICE_OFFSET + 10))
#define    WFS_ERR_PTR_FORMINVALID        (-(PTR_SERVICE_OFFSET + 11))
#define    WFS_ERR_PTR_FIELDINVALID       (-(PTR_SERVICE_OFFSET + 12))
#define    WFS_ERR_PTR_MEDIASKEWED        (-(PTR_SERVICE_OFFSET + 13))
#define    WFS_ERR_PTR_RETRACTBINFULL     (-(PTR_SERVICE_OFFSET + 14))
#define    WFS_ERR_PTR_STACKERFULL        (-(PTR_SERVICE_OFFSET + 15))
#define    WFS_ERR_PTR_PAGETURNFAIL       (-(PTR_SERVICE_OFFSET + 16))
#define    WFS_ERR_PTR_MEDIATURNFAIL      (-(PTR_SERVICE_OFFSET + 17))
#define    WFS_ERR_PTR_SHUTTERFAIL        (-(PTR_SERVICE_OFFSET + 18))
#define    WFS_ERR_PTR_MEDIAJAMMED        (-(PTR_SERVICE_OFFSET + 19))
#define    WFS_ERR_PTR_FILE_IO_ERROR      (-(PTR_SERVICE_OFFSET + 20))
#define    WFS_ERR_PTR_CHARSETDATA        (-(PTR_SERVICE_OFFSET + 21))
#define    WFS_ERR_PTR_PAPERJAMMED        (-(PTR_SERVICE_OFFSET + 22))
#define    WFS_ERR_PTR_PAPEROUT           (-(PTR_SERVICE_OFFSET + 23))
#define    WFS_ERR_PTR_INKOUT             (-(PTR_SERVICE_OFFSET + 24))
#define    WFS_ERR_PTR_TONEROUT           (-(PTR_SERVICE_OFFSET + 25))
#define    WFS_ERR_PTR_LAMPINOP           (-(PTR_SERVICE_OFFSET + 26))
#define    WFS_ERR_PTR_SOURCEINVALID      (-(PTR_SERVICE_OFFSET + 27))
#define    WFS_ERR_PTR_SEQUENCEINVALID    (-(PTR_SERVICE_OFFSET + 28))
#define    WFS_ERR_PTR_MEDIASIZE          (-(PTR_SERVICE_OFFSET + 29))
#define    WFS_ERR_PTR_INVALID_PORT       (-(PTR_SERVICE_OFFSET + 30))
#define    WFS_ERR_PTR_MEDIARETAINED      (-(PTR_SERVICE_OFFSET + 31))
#define    WFS_ERR_PTR_BLACKMARK          (-(PTR_SERVICE_OFFSET + 32))
#define    WFS_ERR_PTR_DEFINITIONEXISTS   (-(PTR_SERVICE_OFFSET + 33))
#define    WFS_ERR_PTR_MEDIAREJECTED      (-(PTR_SERVICE_OFFSET + 34))
#define    WFS_ERR_PTR_MEDIARETRACTED     (-(PTR_SERVICE_OFFSET + 35))
#define    WFS_ERR_PTR_MSFERROR           (-(PTR_SERVICE_OFFSET + 36))
#define    WFS_ERR_PTR_NOMSF              (-(PTR_SERVICE_OFFSET + 37))
#define    WFS_ERR_PTR_FILENOTFOUND       (-(PTR_SERVICE_OFFSET + 38))
#define    WFS_ERR_PTR_POWERSAVETOOSHORT  (-(PTR_SERVICE_OFFSET + 39))
#define    WFS_ERR_PTR_POWERSAVEMEDIAPRESENT (-(PTR_SERVICE_OFFSET + 40))


/*=================================================================*/
/* PTR Info Command Structures */
/*=================================================================*/

typedef struct _wfs_ptr_retract_bins
{
    WORD              wRetractBin;
    USHORT            usRetractCount;
} WFSPTRRETRACTBINS, *LPWFSPTRRETRACTBINS;

typedef struct _wfs_ptr_status
{
    WORD              fwDevice;
    WORD              fwMedia;
```

```
    WORD                fwPaper[WFS_PTR_SUPPLYSIZE];
    WORD                fwToner;
    WORD                fwInk;
    WORD                fwLamp;
    LPWFSPTRRETRACTBINS *lppRetractBins;
    USHORT              usMediaOnStacker;
    LPSTR               lpszExtra;
    DWORD               dwGuidLights[WFS_PTR_GUIDLIGHTS_SIZE];
    WORD                wDevicePosition;
    USHORT              usPowerSaveRecoveryTime;
} WFSPTRSTATUS, *LPWFSPTRSTATUS;

typedef struct _wfs_ptr_caps
{
    WORD                wClass;
    WORD                fwType;
    BOOL                bCompound;
    WORD                wResolution;
    WORD                fwReadForm;
    WORD                fwWriteForm;
    WORD                fwExtents;
    WORD                fwControl;
    USHORT              usMaxMediaOnStacker;
    BOOL                bAcceptMedia;
    BOOL                bMultiPage;
    WORD                fwPaperSources;
    BOOL                bMediaTaken;
    USHORT              usRetractBins;
    LPUSHORT            lpusMaxRetract;
    WORD                fwImageType;
    WORD                fwFrontImageColorFormat;
    WORD                fwBackImageColorFormat;
    WORD                fwCodelineFormat;
    WORD                fwImageSource;
    WORD                fwCharSupport;
    BOOL                bDispensePaper;
    LPSTR               lpszExtra;
    DWORD               dwGuidLights[WFS_PTR_GUIDLIGHTS_SIZE];
    LPSTR               lpszWindowsPrinter;
    BOOL                bMediaPresented;
    USHORT              usAutoRetractPeriod;
    BOOL                bRetractToTransport;
    BOOL                bPowerSaveControl;
} WFSPTRCAPS, *LPWFSPTRCAPS;

typedef struct _wfs_frm_header
{
    LPSTR               lpszFormName;
    WORD                wBase;
    WORD                wUnitX;
    WORD                wUnitY;
    WORD                wWidth;
    WORD                wHeight;
    WORD                wAlignment;
    WORD                wOrientation;
    WORD                wOffsetX;
    WORD                wOffsetY;
    WORD                wVersionMajor;
    WORD                wVersionMinor;
    LPSTR               lpszUserPrompt;
    WORD                fwCharSupport;
    LPSTR               lpszFields;
    WORD                wLanguageID;
} WFSFRMHEADER, *LPWFSFRMHEADER;

typedef struct _wfs_frm_media
{
    WORD                fwMediaType;
    WORD                wBase;
    WORD                wUnitX;
```

```
    WORD                wUnitY;
    WORD                wSizeWidth;
    WORD                wSizeHeight;
    WORD                wPageCount;
    WORD                wLineCount;
    WORD                wPrintAreaX;
    WORD                wPrintAreaY;
    WORD                wPrintAreaWidth;
    WORD                wPrintAreaHeight;
    WORD                wRestrictedAreaX;
    WORD                wRestrictedAreaY;
    WORD                wRestrictedAreaWidth;
    WORD                wRestrictedAreaHeight;
    WORD                wStagger;
    WORD                wFoldType;
    WORD                wPaperSources;
} WFSFRMMEDIA, *LPWFSFRMMEDIA;

typedef struct _wfs_ptr_query_field
{
    LPSTR               lpszFormName;
    LPSTR               lpszFieldName;
} WFSPTRQUERYFIELD, *LPWFSPTRQUERYFIELD;

typedef struct _wfs_frm_field
{
    LPSTR               lpszFieldName;
    WORD                wIndexCount;
    WORD                fwType;
    WORD                fwClass;
    WORD                fwAccess;
    WORD                fwOverflow;
    LPSTR               lpszInitialValue;
    LPWSTR              lpszUNICODEInitialValue;
    LPSTR               lpszFormat;
    LPWSTR              lpszUNICODEFormat;
    WORD                wLanguageID;
} WFSFRMFIELD, *LPWFSFRMFIELD;

typedef struct _wfs_ptr_hex_data
{
    USHORT              usLength;
    LPBYTE              lpbData;
} WFSPTRXDATA, *LPWFSPTRXDATA;

/* WFS INF PTR CODELINE MAPPING input and output structures */

typedef struct _wfs_ptr_codeline_mapping
{
    WORD                wCodelineFormat;
}WFSPTRCODELINEMAPPING, *LPWFSPTRCODELINEMAPPING;

typedef struct  _wfs_ptr_codeline_mapping_out
{
    WORD                wCodelineFormat;
    LPWFSPTRXDATA       lpxCharMapping;
} WFSPTRCODELINEMAPPINGOUT, *LPWFSPTRCODELINEMAPPINGOUT;


/*===================================================================*/
/* PTR Execute Command Structures */
/*===================================================================*/

typedef struct _wfs_ptr_print_form
{
    LPSTR               lpszFormName;
    LPSTR               lpszMediaName;
    WORD                wAlignment;
    WORD                wOffsetX;
```

```
    WORD                wOffsetY;
    WORD                wResolution;
    DWORD               dwMediaControl;
    LPSTR               lpszFields;
    LPWSTR              lpszUNICODEFields;
    WORD                wPaperSource;
} WFSPTRPRINTFORM, *LPWFSPTRPRINTFORM;


typedef struct _wfs_ptr_read_form
{
    LPSTR               lpszFormName;
    LPSTR               lpszFieldNames;
    LPSTR               lpszMediaName;
    DWORD               dwMediaControl;
} WFSPTRREADFORM, *LPWFSPTRREADFORM;


typedef struct _wfs_ptr_read_form_out
{
    LPSTR               lpszFields;
    LPWSTR              lpszUNICODEFields;
} WFSPTRREADFORMOUT, *LPWFSPTRREADFORMOUT;


typedef struct _wfs_ptr_raw_data
{
    WORD                wInputData;
    ULONG               ulSize;
    LPBYTE              lpbData;
} WFSPTRRAWDATA, *LPWFSPTRRAWDATA;


typedef struct _wfs_ptr_raw_data_in
{
    ULONG               ulSize;
    LPBYTE              lpbData;
} WFSPTRRAWDATAIN, *LPWFSPTRRAWDATAIN;


typedef struct _wfs_ptr_media_unit
{
    WORD                wBase;
    WORD                wUnitX;
    WORD                wUnitY;
} WFSPTRMEDIAUNIT, *LPWFSPTRMEDIAUNIT;


typedef struct _wfs_ptr_media_ext
{
    ULONG               ulSizeX;
    ULONG               ulSizeY;
} WFSPTRMEDIAEXT, *LPWFSPTRMEDIAEXT;


typedef struct _wfs_ptr_image_request
{
    WORD                wFrontImageType;
    WORD                wBackImageType;
    WORD                wFrontImageColorFormat;
    WORD                wBackImageColorFormat;
    WORD                wCodelineFormat;
    WORD                fwImageSource;
    LPSTR               lpszFrontImageFile;
    LPSTR               lpszBackImageFile;
} WFSPTRIMAGEREQUEST, *LPWFSPTRIMAGEREQUEST;


typedef struct _wfs_ptr_image
{
    WORD                wImageSource;
    WORD                wStatus;
    ULONG               ulDataLength;
    LPBYTE              lpbData;
} WFSPTRIMAGE, *LPWFSPTRIMAGE;


typedef struct _wfs_ptr_reset
{
```

```
    DWORD                   dwMediaControl;
    USHORT                  usRetractBinNumber;
} WFSPTRRESET, *LPWFSPTRRESET;


typedef struct _wfs_ptr_set_guidlight
{
    WORD                    wGuidLight;
    DWORD                   dwCommand;
} WFSPTRSETGUIDLIGHT, *LPWFSPTRSETGUIDLIGHT;


typedef struct _wfs_ptr_print_raw_file
{
    LPSTR                   lpszFileName;
    DWORD                   dwMediaControl;
    DWORD                   dwPaperSource;
} WFSPTRPRINTRAWFILE, *LPWFSPTRPRINTRAWFILE;


typedef struct _wfs_ptr_load_definition
{
    LPSTR                   lpszFileName;
    BOOL                    bOverwrite;
} WFSPTRLOADDEFINITION, *LPWFSPTRLOADDEFINITION;


typedef struct _wfs_ptr_supply_replen
{
    WORD                    fwSupplyReplen;
} WFSPTRSUPPLYREPLEN, *LPWFSPTRSUPPLYREPLEN;


typedef struct _wfs_ptr_power_save_control
{
    USHORT                  usMaxPowerSaveRecoveryTime;
} WFSPTRPOWERSAVECONTROL, *LPWFSPTRPOWERSAVECONTROL;



/*=================================================================*/
/* PTR Message Structures */
/*=================================================================*/

typedef struct _wfs_ptr_field_failure
{
    LPSTR                   lpszFormName;
    LPSTR                   lpszFieldName;
    WORD                    wFailure;
} WFSPTRFIELDFAIL, *LPWFSPTRFIELDFAIL;


typedef struct _wfs_ptr_bin_threshold
{
    USHORT                  usBinNumber;
    WORD                    wRetractBin;
} WFSPTRBINTHRESHOLD, *LPWFSPTRBINTHRESHOLD;


typedef struct _wfs_ptr_paper_threshold
{
    WORD                    wPaperSource;
    WORD                    wPaperThreshold;
} WFSPTRPAPERTHRESHOLD, *LPWFSPTRPAPERTHRESHOLD;


typedef struct _wfs_ptr_media_detected
{
    WORD                    wPosition;
    USHORT                  usRetractBinNumber;
} WFSPTRMEDIADETECTED, *LPWFSPTRMEDIADETECTED;


typedef struct _wfs_ptr_bin_status
{
    USHORT                  usBinNumber;
    WORD                    wRetractBin;
} WFSPTRBINSTATUS, *LPWFSPTRBINSTATUS;


typedef struct _wfs_ptr_media_presented
```

```
{
    USHORT                  usWadIndex;
    USHORT                  usTotalWads;
} WFSPTRMEDIAPRESENTED, *LPWFSPTRMEDIAPRESENTED;

typedef struct _wfs_ptr_definition_loaded
{
    LPSTR                   lpszDefinitionName;
    DWORD                   dwDefinitionType;
} WFSPTRDEFINITIONLOADED, *LPWFSPTRDEFINITIONLOADED;

typedef struct _wfs_ptr_media_rejected
{
    WORD                    wMediaRejected;
} WFSPTRMEDIAREJECTED, *LPWFSPTRMEDIAREJECTED;

typedef struct _wfs_ptr_media_retracted
{
    WORD                    wRetractResult;
    USHORT                  usBinNumber;
} WFSPTRMEDIARETRACTED, *LPWFSPTRMEDIARETRACTED;

typedef struct _wfs_ptr_device_position
{
    WORD                    wPosition;
} WFSPTRDEVICEPOSITION, *LPWFSPTRDEVICEPOSITION;

typedef struct _wfs_ptr_power_save_change
{
    USHORT                  usPowerSaveRecoveryTime;
} WFSPTRPOWERSAVECHANGE, *LPWFSPTRPOWERSAVECHANGE;


/*   restore alignment   */
#pragma pack(pop)

#ifdef __cplusplus
}       /*extern "C"*/
#endif
#endif     /* __INC_XFSPTR__H */
```