CEN

WORKSHOP

AGREEMENT

CWA 13449-3

December 1998

ICS 35.200;35.240.15

English version

Extensions for Financial Services (XFS) interface specification - Part 3: Printer Device Class Interface - Programmer's Interface

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 Central Secretariat 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, Czech Republic, Denmark, Finland, France, Germany, Greece, Iceland, Ireland, Italy, Luxembourg, Netherlands, Norway, Portugal, Spain, Sweden, Switzerland and United Kingdom.

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

Central Secretariat: rue de Stassart, 36 B-1050 Brussels

Ref. No. CWA 13449-3:1998 E

Contents

Foreword

This CWA is revision 2.0 of the XFS interface specification. Release 2.0 extends the scope of the XFS interface specification to include both the self service/ATM environment as well as the branch environment. The new specification now fully supports cameras, deposit units, identification cards, PIN pads, sensors and indicator units, text terminals, cash dispenser modules and a wide variety of printing mechanisms.

This specification was originally developed by the Banking Solutions Vendor Council (BSVC), and is endorsed by the CEN/ISSS Workshop on XFS. This Workshop gathers both suppliers (among others the BSVC members) 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.

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 2.00.

This CWA is supplemented by a set of release notes, which are available from the CEN/ISSS Secretariat (an on-line version of these release notes is available from http://www.cenorm.be/isss/Workshop/XFS/release-notes.htm).

0. Introduction

This is part 3 of the multi-part CWA 13449, describing Release 2.0 of the XFS interface specification.

The full CWA 13449 "Extensions for Financial Services (XFS) interface specification"consists of the following parts:

Part 1: Application Programming Interface (API) - Service Provider Interface (SPI); Programmer's Reference
Part 2: Service Classes Definition; Programmer's Reference
Part 3: Printer 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

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 from the CEN/ISSS Secretariat (contact isss@cenorm.be or download from http://www.cenorm.be/isss/Workshop/XFS/release-notes.htm).

The information in this document originally contributed by members of the Banking Solutions Vendor Council and endorsed by the CEN/ISSS Workshop on XFS, 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.

The XFS specifications are now further developed in the CEN/ISSS Workshop on XFS. CEN/ISSS Workshops are open to all interested parties offering to contribute. Parties interested in participating should contact the CEN/ISSS Secretariat (isss@cenorm.be).

A Software Development Kit (SDK) which supplies the components and tools to allow the implementation of compliant applications and services is available from Microsoft[1].

To the extent that date processing occurs, all XFS Workshop participants agree that the XFS specifications are Year 2000 compliant.

Revision History:

1.0	**May 24, 1993**	**Initial release of API and SPI specification**
1.11	**February 3, 1995**	**Separation of specification into separate documents for API/SPI and service class definitions, with updates**
2.00	**November 11, 1996**	**Updated release encompassing self-service environment.**
	October 6, 1998	**WOSA/XFS Release 2.00 as originally developed by the BSVC, has been formally accepted as a CEN Workshop Agreement by the CEN/ISSS XFS Workshop and the name WOSA/XFS has been changed into XFS. In spite of the name change, certain occurrencies of WOSA/XFS however still appear in the documentation, for compatibility reasons**

[1] Microsoft is a registered trademark, and Windows and Windows NT are trademarks of Microsoft Corporation

1. XFS Service-Specific Programming

The service classes are defined by their service-specific commands and the associated data structures, error codes, messages, etc. These commands are used to request functions that are specific to one or more classes of service providers, but not all of them, and therefore are not included in the common API for basic or administration functions.

When a service-specific command is common among two or more classes of service providers, the syntax of the command is as similar as possible across all services, since a major objective of the Extensions for Financial Services specification is to standardize command codes and structures for the broadest variety of services. For example, using the **WFSExecute** function, the commands to read data from various services are as similar as possible to each other in their syntax and data structures.

In general, the specific command set for a service class is defined as the union of the sets of specific capabilities likely to be provided by the developers of the services of that class; thus any particular device will normally support only a subset of the command set defined for the class.

There are three cases in which a service provider may receive a service-specific command that it does not support:

- The requested capability is defined for the class of service providers by the XFS specification, the particular vendor implementation of that service does not support it, and the unsupported capability is ***not*** considered to be fundamental to the service. In this case, the service provider returns a successful completion, but does no operation. An example would be a request from an application to turn on a control indicator on a passbook printer; the service provider recognizes the command, but since the passbook printer it is managing does not include that indicator, the service provider does no operation and returns a successful completion to the application.
- The requested capability is defined for the class of service providers by the XFS specification, the particular vendor implementation of that service does not support it, and the unsupported capability ***is*** considered to be fundamental to the service. In this case, a WFS_ERR_UNSUPP_COMMAND error is returned to the calling application. An example would be a request from an application to a cash dispenser to dispense coins; the service provider recognizes the command but, since the cash dispenser it is managing dispenses only notes, returns this error.
- The requested capability is ***not*** defined for the class of service providers by the XFS specification. In this case, a WFS_ERR_INVALID_COMMAND error is returned to the calling application.

This design allows implementation of applications that can be used with a range of services that provide differing subsets of the functionalities that are defined for their service class. Applications may use the **WFSGetInfo** and **WFSAsyncGetInfo** commands to inquire about the capabilities of the service they are about to use, and modify their behavior accordingly, or they may use functions and then deal with WFS_ERR_UNSUPP_COMMAND error returns to make decisions as to how to use the service.

2. Banking Printers

This specification describes the functionality of the services provided by banking printers under XFS, focusing on three areas:

- application programming for printing
- print document definition
- integration with the Windows architecture

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 insure 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 services, and for the development of those services.

3. Banking Printer Types

The XFS printer service defines and supports four 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.

- **Document Printer**
  Document printing is similar to receipt printing -- a set of fields are positioned on an inserted sheet of paper -- but the focus is on full-size forms. It should be noted that the XFS environment only implements the printing of text fields from the application. The electronic printing of the form image itself is not supported; but can be delivered as an added-value extension by the vendor.

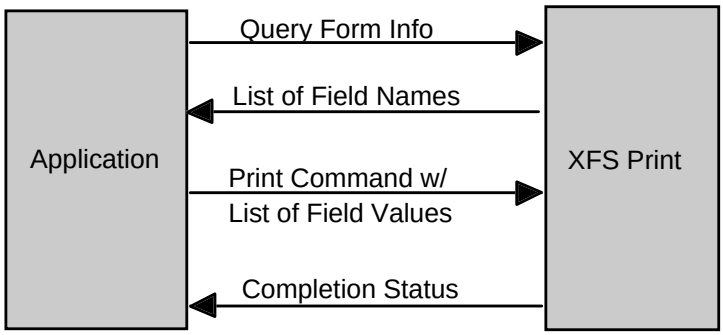
Additional hardware components, like scanners, stripe readers, OCR readers, and stamps, normally attached directly to the printer are also controlled through this interface.

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. 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.

Finally, 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.

6. Info Commands

6.1 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;
   WORD      fwToner;
   WORD      fwInk;
   WORD      fwLamp;
   WORD      fwRetractBin;
```

```
USHORT    usRetractCount;
USHORT    usMediaOnStacker;
LPSTR     lpszExtra;
} 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.
WFS_PTR_DEVOFFLINE	The device is offline.
WFS_PTR_DEVPOWEROFF	The device is powered off.
WFS_PTR_DEVBUSY	The device is busy processing a request.
WFS_PTR_DEVNODEVICE	There is no device connected.
WFS_PTR_DEVUSERERROR	The device is present but a person is preventing proper device operation. The application should suspend the device from service until the service provider generates a device state change event indicating the condition of the device has changed, e.g., the error is removed (WFS_PTR_DEVONLINE) or a permanent error condition has occurred (WFS_PTR_DEVHWERROR).
WFS_PTR_DEVHWERROR	The device is inoperable due to a hardware error.

fwMedia
Specifies the state of the print media (i.e., the paper: passbook, single sheet, roll, etc.) as one of the following flags:

Value	Meaning
WFS_PTR_MEDIAPRESENT	Media is inserted in the device.
WFS_PTR_MEDIANOTPRESENT	Media is not inserted in the device.
WFS_PTR_MEDIAJAMMED	Media is jammed in the device.
WFS_PTR_MEDIAUNKNOWN	The state of the print media cannot be determined with the device in its current state.
WFS_PTR_MEDIANOTSUPP	The capability to report the state of the print media is not supported by the device.
WFS_PTR_MEDIAENTERING	Media is at the entry/exit slot of the device.

fwPaper
Specifies the state of the paper supply as one of the following flags:

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	Capability cannot be determined with device in its current state.

fwToner
Specifies the state of the print device as one of the following flags:

Value	Meaning
WFS_PTR_TONERFULL	The toner supply is full.
WFS_PTR_TONERLOW	The toner supply is low.
WFS_PTR_TONEROUT	The toner supply is empty.
WFS_PTR_TONERNOTSUPP	Capability not supported by device.
WFS_PTR_TONERUNKNOWN	Capability cannot be determined with device in its current state.

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

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	Capability cannot be determined with device in its current state.

fwLamp
Specifies the status of the printer imaging lamp as one of:

Value	Meaning
WFS_PTR_LAMPOK	The lamp is OK.
WFS_PTR_LAMPFADING	The lamp should be changed.
WFS_PTR_LAMPINOP	The lamp is inoperable.
WFS_PTR_LAMPNOTSUPP	Capability not supported by device.
WFS_PTR_LAMPUNKNOWN	Capability cannot be determined with device in its current state.

fwRetractBin
Specifies the state of the printer retract bin as one of the following flags:

Value	Meaning
WFS_PTR_RETRACTBINOK	The retract bin of the printer is not full.
WFS_PTR_RETRACTBINFULL	The retract bin of the printer is full.
WFS_PTR_RETRACTNOTSUPP	The printer does not support retract capability.
WFS_PTR_RETRACTUNKNOWN	Capability cannot be determined with device in its current state.
WFS_PTR_RETRACTBINHIGH	The retract bin of the printer is high.

usRetractCount
The number of media retracted; applicable only to printers with retract capability. This value is persistent: it is 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.

Error Codes There are no additional error codes 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.

6.2 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    usMaxRetract;
   USHORT    usMaxMediaOnStacker;
   BOOL      bAcceptMedia;
   LPSTR     lpszExtra;
   } WFSPTRCAPS, * LPWFSPTRCAPS;
```

wClass
Specifies the logical service class, value is:
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.

bCompound
Specifies whether the logical device is part of a compound physical device and is either TRUE or FALSE.

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:

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.

fwWriteForm
Specifies whether the device can write data to the media, as a combination of the following flags:

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:

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 bit flags:

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
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.

usMaxRetract
Specifies the maximum number of media items that the retract bin can hold (zero if not available).

usMaxMediaOnStacker
Specifies the maximum number of media items that the stacker can hold (zero if this capability is 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.

lpszExtra
Points 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.

Error Codes There are no additional error codes 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.

6.3 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 There are no additional error codes generated by this command.

Comments None.

6.4 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 There are no additional error codes generated by this command.

Comments None.

6.5 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;
   LPSTR     lpszFields;
   } 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:

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:

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.

wVersionMinor
Specifies the minor version of the form.

lpszUserPrompt
Pointer to a null-terminated user prompt string.

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

Error Codes The following additional 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.

6.6 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;
} WFSFRMMEDIA, * LPWFSFRMMEDIA;
```

fwMediaType
Specifies the type of media as one of the following flags:

Value	Meaning
WFS_FRM_MEDIAGENERIC	Generic media, i.e., single sheet.
WFS_FRM_MEDIAMULTIPART	Multipart media.
WFS_FRM_MEDIAPASSBOOK	Passbook media.

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.

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.

w*Stagger*
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.

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.

Error Codes The following additional 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.

6.7 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 this value is NULL, then retrieve details for all fields on the form.

Output Param
```
LPWFSFRMFIELD *     lppFields;
```

lppFields
Pointer to a null-terminated array of pointers to field definition structures:
```
  typedef struct _wfs_frm_field
      {
      LPSTR     lpszFieldName;
      WORD      wIndexCount;
      WORD      fwType;
      WORD      fwClass;
      WORD      fwAccess;
      WORD      fwOverflow;
      LPSTR     lpszInitialValue;
      LPSTR     lpszFormat;
      } 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:

Value	Meaning
WFS_FRM_FIELDTEXT	A text field.
WFS_FRM_FIELDMICR	A Magnetic Ink Character Recognition field.
WFS_FRM_FIELDOCR	An Optical Character Recognition field.
WFS_FRM_FIELDMSF	A Magnetic Stripe Facility field.
WFS_FRM_FIELDBARCODE	A Barcode field.
WFS_FRM_FIELDGRAPHIC	A Graphic field
WFS_FRM_FIELDPAGEMARK	A Page Mark field

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

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 bit-flags:

Value	Meaning
WFS_FRM_ACCESSREAD	The field is used for input.
WFS_FRM_ACCESSWRITE	The field is used for ouput.

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

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.

lpszFormat
Format string as defined in the form for this field.

Error Codes The following additional 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.
WFS_ERR_PTR_FIELDNOTFOUND	The specified field cannot be found.
WFS_ERR_PTR_FIELDINVALID	The specified field is invalid.

Comments None.

7. Execute Commands

7.1 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.

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_CTRLFLUSH	Flush any data to the printer that has not yet been printed from previous WFS_CMD_PTR_PRINT_FORM commands.
WFS_PTR_CTRLEJECT	Flush data as above, 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_CTRLRETRACT	Flush data as above, then retract the media.
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	Caused 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.

Output Param None.

Error Codes The following additional error codes can be generated by this command:

Value	Meaning
WFS_ERR_PTR_NOMEDIAPRESENT	No form 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.

Events The following additional events can be generated by this command:

Value	Meaning
WFS_USRE_PTR_RETRACTBINFULL	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.
WFS_SRVE_PTR_MEDIATAKEN	The media has been taken by the user.

Comments None.

7.2 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.

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;
   } WFSPTRPRINTFORM, * LPWFSPTRPRINTFORM;
```

lpszFormName
Pointer to the null-terminated form name.

lpszMediaName
Pointer to the null-terminated media name.

wAlignment
Specifies the alignment of the form on the physical medium, as one of these values:

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

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 medium, 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 medium, 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, as a combination of the flags described under WFS_CMD_PTR_CONTROL_MEDIA. A NULL 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 final 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.

Output Param None.

Error Codes The following additional error codes can be generated by this command:

Value	Meaning
WFS_ERR_PTR_FORMNOTFOUND	The specified form definition cannot be found.
WFS_ERR_PTR_FORMINVALID	The specified form definition is invalid.
WFS_ERR_PTR_MEDIANOTFOUND	The specified media definition cannot be found.
WFS_ERR_PTR_MEDIAINVALID	The specified media definition is invalid.
WFS_ERR_PTR_MEDIASKEWED	The media skew exceeded the limit in the form definition.
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.

Events The following additional events can be generated by this command:

Value	Meaning
WFS_EXEE_PTR_NOMEDIA	No media is present in the device.
WFS_EXEE_PTR_FIELDERROR	A fatal error occured while processing a field.
WFS_EXEE_PTR_FIELDWARNING	A non-fatal error occured while processing a field.
WFS_EXEE_PTR_MEDIAINSERTED	Media has been inserted into the device.

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.

7.3 WFS_CMD_PTR_READ_FORM

Description This command is used to read data from input fields on the specified 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.

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.

dwMediaControl
Specifies the manner in which the media should be handled and can be a combination of the bit flags described under WFS_CMD_PTR_CONTROL_MEDIA.

Output Param `LPSTR lpszFields;`

lpszFields
Pointer to a series of "<FieldName>=<FieldValue>" strings, where each string is null-terminated with the final 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 The following additional error codes can be generated by this command:

Value	Meaning
WFS_ERR_PTR_READNOTSUPPORTED	The device has no read capability.
WFS_ERR_PTR_FORMNOTFOUND	The specified form cannot be found.
WFS_ERR_PTR_FORMINVALID	The specified form definition is invalid.
WFS_ERR_PTR_MEDIANOTFOUND	The specified media definition cannot be found.
WFS_ERR_PTR_MEDIAINVALID	The specified media definition is invalid.
WFS_ERR_PTR_MEDIASKEWED	The media skew exceeded the limit in the form definition.
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.

Events The following additional events can be generated by this command:

Value	Meaning
WFS_EXEE_PTR_NOMEDIA	No media is present in the device.
WFS_EXEE_PTR_FIELDERROR	A fatal error occured while processing a field.
WFS_EXEE_PTR_FIELDWARNING	A non-fatal error occured while processing a field.
WFS_EXEE_PTR_MEDIAINSERTED	Media has been inserted into the device.

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**.**

7.4 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 There are no additional error codes generated by this command.

Events There are no additional events generated by this command.

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.

7.5 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:

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 The following additional error codes can be generated by this command:

Value	Meaning
WFS_ERR_PTR_EXTENTNOTSUPPORTED	The device cannot report extent(s).

Events The following additional 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.

Comments None.

7.6 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 None.

Output Param None.

Error Codes There are no additional error codes generated by this command.

Events There are no additional events generated by this command.

Comments None.

7.7 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 None.

Output Param `LPWFSPTRIMAGE lpImage;`

```
typedef struct _wfs_ptr_image
   {
   WORD      wImageType;
   ULONG     ulSize;
   LPBYTE    lpImage;
   } WFSPTRIMAGE, * LPWFSPTRIMAGE;
```

wImageType
Specifies the format of the image returned by this command as one of the following flags:

Value	Meaning
WFS_PTR_IMAGETIF	The returned image is in TIF format.
WFS_PTR_IMAGEMTF	The returned image is in MTF format.
WFS_PTR_IMAGEBMP	The returned image is in BMP format.

ulSize
Count of bytes of image data.

lpImage
Points to the image data.

Error Code There are no additional error codes generated by this command.

Events The following additional 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.

Comments None.

8. Events

8.1 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.

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.

8.2 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.

8.3 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:

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.

Comments None.

8.4 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.

8.5 WFS_USRE_PTR_RETRACTBINTHRESHOLD

Description This event specifies that the retract bin holding the retracted media is full.

Event Param `LPWORD lpwRetractBinThreshold;`

lpwRetractBinThreshold
Specified as one of the following flags:

Value	Meaning
WFS_PTR_RETRACTBINFULL	The retract bin of the printer is full.
WFS_PTR_RETRACTBINHIGH	The retract bin of the printer is high.

Comments None.

8.6 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.

Event Param None.

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

8.7 WFS_USRE_PTR_PAPERTHRESHOLD

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

Event Param LPWORD lpwPaperThreshold;

Specified as one of the following flags:

Value	Meaning
WFS_PTR_PAPERLOW	The paper in the printer is low.
WFS_PTR_PAPEROUT	The paper in the printer is out.

Comments None.

8.8 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;

Specified as one of the following flags:

Value	Meaning
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.

8.9 WFS_SRVE_PTR_MEDIAINSERTED

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

Event Param **N**one.

Comments None.

9. Form, Field and Media Definitions

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

9.1 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.

9.2 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 ***form*** 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 field definition keyword sections:

- POSITION (*x* and *y* values)
- SIZE (*width* and *height* values)
- INDEX (*xoffset* and *yoffset* values)

The base resolutions thus defined by the UNIT keyword section of the ***media*** 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)

9.3 Form Definition

XFSFORM		*formname**	
BEGIN			
(required)	**UNIT**	*base,*	Base resolution unit for form definition MM INCH 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 medium: TOPLEFT (default) TOPRIGHT BOTTOMLEFT BOTTOMRIGHT This option allows the positioning of a form onto a physical page relative to any combination of the edges of the physical medium, 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 medium, 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 medium, means offset the form upward). (default = 0)
	ORIENTATION	*type*	Orientation of form: PORTRAIT (default) 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)
	COPYRIGHT	*copyright**	Copyright entry
	TITLE	*title**	Title of form
	COMMENT	*comment**	Comment section
	USERPROMPT	*prompt**	Prompt string for user interaction
	[XFSFIELD	*fieldname**	One field definition (as defined in the next section) for each field in the form
	BEGIN **. . .** **END]**		
	[XFSFRAME	*framename**	One frame definition (as defined in the next section) for each frame in the form
	BEGIN **. . .** **END]**		
END			

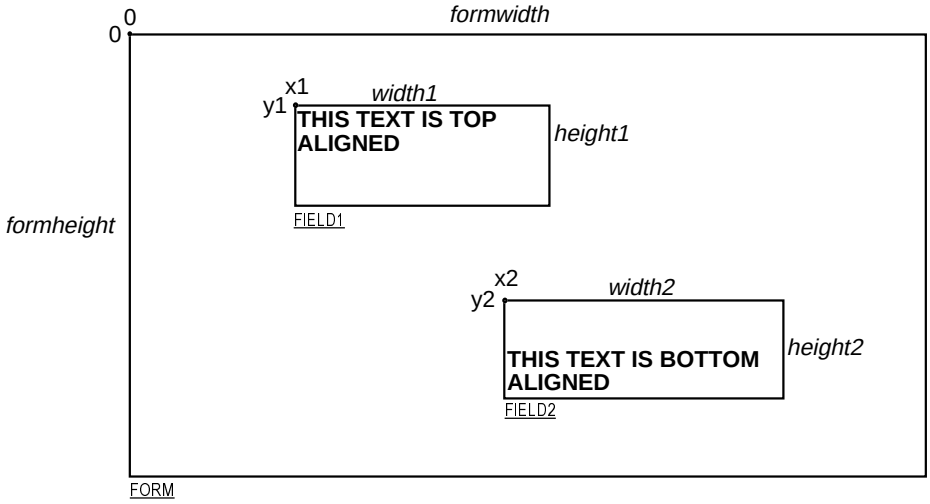
9.4 Field Definition

XFSFIELD		*fieldname**	
BEGIN			
(required)	**POSITION**	*x,* *y*	Horizontal position (relative to left side of form) Vertical position (relative to top of form)
	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.
	SIDE	*side*	Side of form where field is positioned: FRONT (default) BACK
(required)	**SIZE**	*width,* *height*	Field width Field height
	INDEX	*repeatcount,* *xoffset,* *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
	TYPE	*fieldtype*	Type of field: TEXT (default) MICR OCR MSF BARCODE GRAPHIC PAGEMARK
	SCALING	*scalingtype*	Information on how to size the GRAPHIC within the field: BESTFIT (default) scale to size indicated ASIS render at native size MAINTAINASPECT scale as close as possible to size indicated while maintaining the aspect ratio and not losing graphic information. SCALING is only relevant for GRAPHIC field types.
	BARCODE	*hriposition*	Position of the HRI (Human Readable Interpretation) characters: NONE (default) ABOVE BELOW BOTH The type of barcode to print is defined in the FONT field.
	CLASS	*class*	Field class OPTIONAL (default) STATIC REQUIRED
	ACCESS	*access*	Access rights of field WRITE (default) READ READWRITE

	OVERFLOW	*overflow*	Action on field overflow: TERMINATE (default) TRUNCATE BESTFIT (the service provider fits the data into the field as well as it can) OVERWRITE (a contiguous write) WORDWRAP
	STYLE	*style*	Display attributes as a combination of the following, ORed together using the "|" operator: NORMAL (default) BOLD ITALIC UNDER (single underline) DOUBLEUNDER (double underline) DOUBLE (double width) TRIPLE (triple width) QUADRUPLE (quadruple width) STRIKETHROUGH ROTATE90 (rotate +90 degrees clockwise) ROTATE270 (rotate +270 degrees clockwise) UPSIDEDOWN (upside down) PROPORTIONAL (proportional spacing) DOUBLEHIGH TRIPLEHIGH QUADRUPLEHIGH CONDENSED SUPERSCRIPT SUBSCRIPT OVERSCORE LETTERQUALITY NEARLETTERQUALITY DOUBLESTRIKE OPAQUE (If omitted then default attribute is transparent) Some of these Styles may be mutually exclusive, or may combine to provide unexpected results.
	CASE	*case*	Convert field contents to NOCHANGE (default) UPPER LOWER
	HORIZONTAL	*justify*	Horizontal alignment of field contents LEFT (default) RIGHT CENTER JUSTIFY
	VERTICAL	*justify*	Vertical alignment of field contents BOTTOM (default) CENTER TOP
	COLOR	*color*	Color name BLACK (default) WHITE GRAY RED BLUE GREEN YELLOW

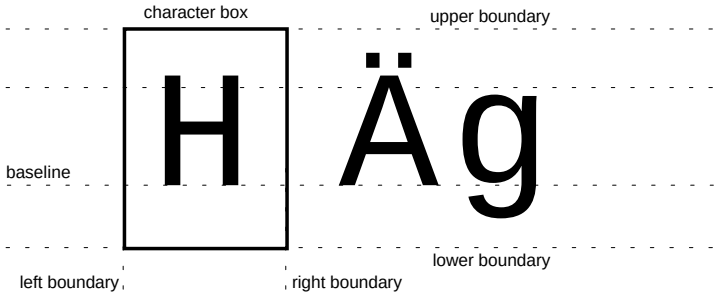
	LANGUAGE	*languageID*	Language used in this field – a 16 bit value (LANGID) which is a combination of a primary (10 bits) and a secondary (6 bits) language ID (This is the standard language ID in the Win32 API; standard macros support construction and decomposition of this composite ID) If unspecified defaults to form definition LANGUAGE specification.
font	**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:
definition	**POINTSIZE**	*pointsize*	Point size
information	**CPI**	*cpi*	Characters per inch
	LPI	*lpi*	Lines per inch
	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.
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:



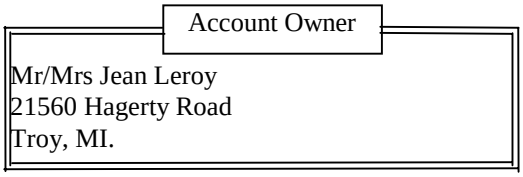
When more than one line of text is to be printed in a field, and the definition includes **VERTICAL=BOTTOM,** the vertical position of the first line is calculated using the specified (or implied) **LPI** value.

9.5 Frame Definition

XFSFRAME		*framename**	
BEGIN			
(required)	**POSITION**	*x,* *y*	Horizontal position of top left corner of the frame (relative to left side of form) Vertical position of top left corner of the frame (relative to top of form)
	FRAMES	*fieldname**	Frames the field with the name <fieldname>, positioning information is 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.
	SIDE	*side*	Side of form where this frame is positionned: 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.
	TYPE	*frametype*	Type of frame: RECTANGLE (default) ROUNDED_CORNER ELLIPSE
	CLASS	*class*	Frame class: STATIC (default) OPTIONAL(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: TERMINATE (default) TRUNCATE BESTFIT (the service provider fits the frame into the media as well as it can)
	STYLE	*style*	Frame line attributes: SINGLE_THIN (default) DOUBLE_THIN SINGLE_THICK DOUBLE_THICK DOTTED

	COLOR	*color*	Color name for frame lines: BLACK (default) WHITE GRAY RED BLUE GREEN YELLOW
	FILLCOLOR	*color*	Color name for interior of frame: BLACK WHITE (default) GRAY RED BLUE GREEN YELLOW
	FILLSTYLE	*style*	Style for filling the interior of frame: NONE (default) SOLID Solid color BDIAGONAL Downward hatch (left to right) at 45 degrees CROSS Horizontal and vertical crosshatch DIAGCROSS Crosshatch at 45 degrees FDIAGONAL Upward hatch (left to right) at 45 degrees HORIZONTAL Horizontal hatch VERTICAL Vertical hatch
Frame title	**TITLE**	*fieldname**	Uses the field with the name <fieldname> as the title of the frame. Positioning information of the field is ignored.
definition	**HORIZONTAL**	*justify*	Horizontal alignment of the frame title: LEFT (default) CENTER RIGHT
information	**VERTICAL**	*justify*	Vertical alignment of the frame title: TOP (default) 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:



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 override 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 provider 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"
  END

  XFSFIELD "Account"
  BEGIN
```

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
0123456789123001	$17465.12
0123456789123002	$2458.23
0123456789123003	$6542.78

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

Account	Balance
0123456789123001	$17465.12

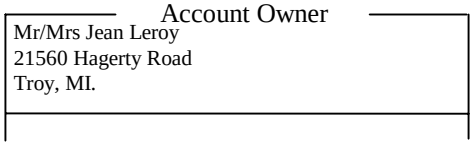
```
    POSITION 15, 8
    SIZE 30, 4
    INDEX 10, 0, 3
  END //"Account"
  XFSFIELD "Balance"
  BEGIN
    POSITION 45, 8
    SIZE 30, 4
    INDEX 10, 0, 3
    HORIZONTAL RIGHT
  END //"Balance"
  XFSFRAME "Account Title"
  BEGIN
    POSITION 15, 4
    FRAMES "Account Title"
    SIZE 30, 4
    STYLE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

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

will print:



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:

Mr/Mrs Jean Leroy
21560 Hagerty Road
Troy, MI.

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

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

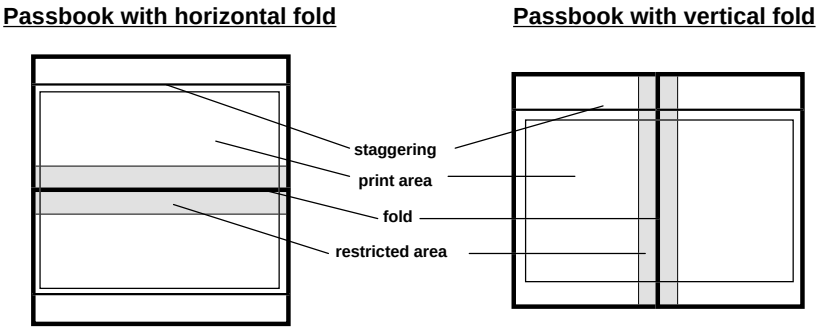
will print

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

9.6 Media Definition

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.



XFSMEDIA		*medianame**	
BEGIN			
	TYPE	*type*	Predefined media types are: GENERIC (default) MULTIPART PASSBOOK
(required)	**UNIT**	*base,* *x,* *y,*	Base resolution unit for media definition MM INCH ROWCOLUMN Horizontal base unit fraction Vertical base unit fraction
(required)	**SIZE**	*width,* *height*	Width of physical media Height of physical media (0 = unlimited, i.e, roll paper)
	PRINTAREA	*x,* *y,* *width,* *height*	Printable area relative to top left corner of physical media (default = physical size of media)
	RESTRICTED	*x,* *y,* *width,* *height*	Restricted area relative to to top left corner of physical media (default = no restricted area)
	FOLD	*fold*	Type of passbook HORIZONTAL VERTICAL
	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. C-Header File

```
/******************************************************************************
*                                                                             *
* xfsptr.h    XFS - Banking Printer (PTR) definitions                         *
*               (receipt, journal, passbook and document printer)             *
*                                                                             *
*            Version 2.00  (11/11/96)                                         *
*                                                                             *
******************************************************************************/

#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    (0x0002) /* Version 2.00 */
#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)

/* 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)

/* 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)
#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)


/* 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_DEVBUSY                  WFS_STAT_DEVBUSY
#define    WFS_PTR_DEVNODEVICE              WFS_STAT_DEVNODEVICE
```

```
#define    WFS_PTR_DEVHWERROR              WFS_STAT_DEVHWERROR
#define    WFS_PTR_DEVUSERERROR            WFS_STAT_DEVUSERERROR

/* values of WFSPTRSTATUS.fwMedia */

#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)

/* values of WFSPTRSTATUS.fwPaper */

#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)

/* 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[2]                (0)
#define    WFS_PTR_LAMPFADING               (1)
#define    WFS_PTR_LAMPINOP                 (2)
#define    WFS_PTR_LAMPNOTSUPP              (3)
#define    WFS_PTR_LAMPUNKNOWN              (4)

/* values of WFSPTRSTATUS.fwRetractBin */

#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)

/* values of WFSPTRCAPS.fwType */

#define    WFS_PTR_TYPERECEIPT              0x0001
#define    WFS_PTR_TYPEPASSBOOK             0x0002
#define    WFS_PTR_TYPEJOURNAL              0x0004
#define    WFS_PTR_TYPEDOCUMENT             0x0008

/* 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
```

```
/* 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

/* 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)

/* 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.wImageType */

#define    WFS_PTR_IMAGETIF                 (1)
#define    WFS_PTR_IMAGEMTF                 (2)
#define    WFS_PTR_IMAGEBMP                 (3)


/* 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))


/*=================================================================*/
/* PTR Info Command Structures and variables */
```

```
/*================================================================*/

typedef struct _wfs_ptr_status
{
    WORD            fwDevice;
    WORD            fwMedia;
    WORD            fwPaper;
    WORD            fwToner;
    WORD            fwInk;
    WORD            fwLamp;
    WORD            fwRetractBin;
    USHORT          usRetractCount;
    USHORT          usMediaOnStacker;
    LPSTR           lpszExtra;
} WFSPTRSTATUS, * LPWFSPTRSTATUS;

typedef struct _wfs_ptr_caps
{
    WORD            wClass;
    WORD            fwType;
    BOOL            bCompound;
    WORD            wResolution;
    WORD            fwReadForm;
    WORD            fwWriteForm;
    WORD            fwExtents;
    WORD            fwControl;
    USHORT          usMaxRetract;
    USHORT          usMaxMediaOnStacker;
    BOOL            bAcceptMedia;
    LPSTR           lpszExtra;
} 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;
    LPSTR           lpszFields;
} 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;
} 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;
    LPSTR           lpszFormat;
} WFSFRMFIELD, * LPWFSFRMFIELD;


/*=================================================================*/
/* 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;
} WFSPTRPRINTFORM, * LPWFSPTRPRINTFORM;

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

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
{
    WORD           wImageType;
    ULONG          ulSize;
    LPBYTE         lpImage;
} WFSPTRIMAGE, * LPWFSPTRIMAGE;
```

```
/*==================================================================*/
/* PTR Message Structures */
/*==================================================================*/

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

/*   restore alignment   */
#pragma pack(pop)

#ifdef __cplusplus
}       /*extern "C"*/
#endif
#endif   /* __INC_XFSPTR__H */
```