CEN

WORKSHOP

AGREEMENT

CWA 15748-18

July 2008

ICS 35.240.50

English version

Extensions for Financial Services (XFS) interface specification - Release 3.10 - Part 18: Item Processing Module Device Class Interface - Programmer's Reference

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

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

Ref. No.:CWA 15748-18:2008 D/E/F

Table of Contents

Foreword

This CWA is revision 3.10 of the XFS interface specification.

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

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

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

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

Part 2: Service Classes Definition - Programmer's Reference

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Parts 19 - 28: Reserved for future use.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Part 42: Reserved for future use.

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

Part 44: XFS MIB Application Management

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

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

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

Parts 48 - 60 are reserved for future use.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Revision History:

3.10	November 29, 2007	Initial Release.

1. Introduction

1.1 Background to Release 3.10

The CEN/ISSS XFS Workshop aims to promote a clear and unambiguous specification defining a multi-vendor software interface to financial peripheral devices. The XFS (eXtensions for Financial Services) specifications are developed within the CEN/ISSS (European Committee for Standardization/Information Society Standardization System) Workshop environment. CEN/ISSS Workshops aim to arrive at a European consensus on an issue that can be published as a CEN Workshop Agreement (CWA).

The CEN/ISSS XFS Workshop encourages the participation of both banks and vendors in the deliberations required to create an industry standard. The CEN/ISSS XFS Workshop achieves its goals by focused sub-groups working electronically and meeting quarterly.

Release 3.10 of the XFS specification is based on a C API and is delivered with the continued promise for the protection of technical investment for existing applications. This release of the XFS specification has been prompted by a series of factors.

There has been a technical imperative to extend the scope of the existing specification to include new devices, such as the Barcode Reader, Card Dispenser and Item Processing Module.

Similarly, there has also been pressure, through implementation experience and additional requirements, to extend the functionality and capabilities of the existing devices covered by the specification.

1.2 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 XFS is to standardize function 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 a superset of the 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 defined command set.

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. Item Processing Module

This specification describes the XFS service class for Item Processing Modules (IPM). The specification of this service class includes definitions of the service-specific commands that can be issued, using the **WFSAsyncExecute**, **WFSExecute, WFSGetInfo** and **WFSAsyncGetInfo** functions.

This service class is currently defined only for self service devices.

In the U.S., checks are always encoded in magnetic ink for reading by Magnetic Ink Character Recognition (MICR), and a single font is always used. In Europe some countries use MICR and some use Optical Character Recognition (OCR) character sets, with different fonts, for their checks.

Item Processing Modules accept one or more media items (Checks, Giros, etc) and process these items according to application requirements. The IPM class supports devices that can handle a single item as well as those devices that can handle bunches of items. The following are the three principle device types:

- Single Item: can accept and process a single item at a time.
- Multi-Item Feed with no stacker (known as an escrow in some environments): can accept a bunch of media from the customer but each item has to be processed fully (i.e. deposited in a bin or returned) before the next item can be processed.
- Multi-Item Feed with a stacker: can accept a bunch of media from the customer and all items can be processed together.

The IPM class provides applications with an interface to control the following functions (depending on the capabilities of the specific underlying device):

- Capture an image of the front of an item in multiple formats and bit depths.
- Capture an image of the back of an item in multiple formats and bit depths.
- Read the code line of an item using MICR reader.
- Read the code line of an item using OCR.
- Endorse (print text) on an item.
- Stamp an item.
- Return an item to the customer.
- Deposit an item in a bin.
- Retract items left by the customer.

The IPM device class uses the concept of a Media-In transaction to track and control a customer's interaction with the device. A Media-In transaction consists of one or more WFS_CMD_IPM_MEDIA_IN commands. The transaction is initiated by the first WFS_CMD_IPM_MEDIA_IN command and remains active until the transaction is either confirmed through WFS_CMD_IPM_MEDIA_IN_END, or terminated by WFS_CMD_IPM_MEDIA_IN_ROLLBACK, WFS_CMD_IPM_RETRACT_MEDIA or WFS_CMD_IPM_RESET. While a transaction is active the WFS_INF_IPM_TRANSACTION_STATUS command reports the status of the current transaction. When a transaction is not active the WFS_INF_IPM_TRANSACTION_STATUS command reports the status of the last transaction.

There are primarily two types of devices supported by the IPM, those devices with a stacker and those without.

2.1 Devices with a Stacker

On devices with stackers, the IPM device class supports two mechanisms for deciding if physically acceptable items should be accepted onto the stacker or refused:

- The device/Service Provider automatically makes the accept/refuse decision.
- The application controls the accept/refuse decision.

2.1.1 Automatic Accept/Refuse

In summary, the following process is followed (the exact order will depend on application requirements):

1. The application initiates the transaction via the WFS_CMD_IPM_MEDIA_IN command. This command accepts a bunch of media items. The images and code line for every media item accepted is sent to the application before the command completes.
2. The application then asks the customer if they have any more items to process.
3. If the customer has more items to deposit then the WFS_CMD_IPM_MEDIA_IN command is called one or more times to add more items to the stacker.
4. Once the customer has inserted all their bunches of items and they have been added to the stacker the application can process each item and pre-define what should happen to each media item during the WFS_CMD_IPM_MEDIA_IN_END command, e.g.:
   a. Define if the item should be stamped and what should be printed on the item (using WFS_CMD_IPM_PRINT_TEXT), set the destination bin (using WFS_CMD_IPM_SET_DESTINATION), and request the item is re-scanned after printing (using WFS_CMD_IPM_GET_IMAGE_AFTER_PRINT), or
   b. Define that the item should be returned to the customer (using WFS_CMD_IPM_SET_DESTINATION).
5. When all items have been processed the application calls WFS_CMD_IPM_MEDIA_IN_END to complete the transaction and carry out the pre-defined actions, e.g. print and deposit some items while returning others.

Note: Before the WFS_CMD_IPM_MEDIA_IN_END command is called, the customer can cancel the transaction at any time and all items are returned to the customer by the application calling WFS_CMD_IPM_ROLLBACK.

2.1.2 Application Controlled Accept/Refuse

In summary, the following process is followed (the exact order will depend on application requirements):

1. The application uses the WFS_CMD_IPM_MEDIA_IN command to accept a bunch of media items (the first use of this command initiates the transaction). The application indicates that it wants to make the accept/refuse decision for each item via an input parameter, and as a result only one item is processed and the code line and images are only produced for a single item.
2. The application processes the item and decides if it should be accepted or refused using the WFS_CMD_IPM_ACCEPT_ITEM command.
3. The application calls WFS_CMD_IPM_GET_NEXT_ITEM to read the next item. If an item is read then the flow continues at step 2. When there are no items left to process the flow continues with the next step.
4. The application can return the refused items to the customer with WFS_CMD_IPM_PRESENT_MEDIA.
5. The application then asks the customer if they have any more items to process or wish to re-insert the refused items after correcting the issue causing the refusal.
6. If the customer has more items to deposit then flow continues at step 1, otherwise the flow continues at the next step.
7. Once the customer has inserted all their bunches of items and they have been added to the stacker the application can process each item and pre-define what should happen to each media item during the WFS_CMD_IPM_MEDIA_IN_END command, e.g.:

   a. Define if the item should be stamped and what should be printed on the item (using WFS_CMD_IPM_PRINT_TEXT), set the destination bin (using WFS_CMD_IPM_SET_DESTINATION), and request the item is re-scanned after printing (using WFS_CMD_IPM_GET_IMAGE_AFTER_PRINT), or
   b. Define that the item should be returned to the customer (using WFS_CMD_IPM_SET_DESTINATION).

8. When all items have been processed the application calls WFS_CMD_IPM_MEDIA_IN_END to complete the transaction and carry out the pre-defined actions, e.g. print and deposit some items while returning others.

Note: Before the WFS_CMD_IPM_MEDIA_IN_END command is called, the customer can cancel the transaction at any time and all items are returned to the customer by the application calling WFS_CMD_IPM_ROLLBACK.

2.2 Device without a Stacker

Devices without a stacker fall into two categories those with a multi-item feed unit and those without. Both of these types of devices can be handled by the same application flow, however they are both documented below for clarity.

2.2.1 Multi-Feed Devices without a Stacker

In summary, the following process is followed (the exact order will depend on application requirements):

1. The application uses the WFS_CMD_IPM_MEDIA_IN command to accept a bunch of media items (the first use of this command initiates the transaction). However as there is no stacker only one item is processed and the code line and images are only produced for a single item.
2. The application processes the item and decides what should be done to the item, e.g.:
    a. Define if the item should be stamped and what should be printed on the item (using WFS_CMD_IPM_PRINT_TEXT), set the destination bin (using WFS_CMD_IPM_SET_DESTINATION), and request the item is re-scanned after printing (using WFS_CMD_IPM_GET_IMAGE_AFTER_PRINT), or
    b. Define that the item should be returned to the customer (using WFS_CMD_IPM_SET_DESTINATION).
3. The application calls WFS_CMD_IPM_ACTION_ITEM to have the pre-defined actions executed.
4. The application calls WFS_CMD_IPM_GET_NEXT_ITEM to read the next item. If an item is read then the flow continues at step 2. When there are not items left to process the flow continues with the next step.
5. The application then asks the customer if they have any more items to process.
6. If the customer has more items to deposit then flow continues at step 1.
7. When the customer is finished the application calls WFS_CMD_IPM_MEDIA_IN_END to terminate the transaction.

2.2.2 Single-Feed Devices

In summary, the following process is followed:

1. The application initiates the transaction via the WFS_CMD_IPM_MEDIA_IN command. This command accepts a single item and produces the image and code line.
2. The application processes the item and decides what should be done to the item, e.g.:
    a. Define if the item should be stamped and what should be printed on the item (using WFS_CMD_IPM_PRINT_TEXT), set the destination bin (using WFS_CMD_IPM_SET_DESTINATION), and request the item is re-scanned after printing (using WFS_CMD_IPM_GET_IMAGE_AFTER_PRINT), or
    b. Define that the item should be returned to the customer (using WFS_CMD_IPM_SET_DESTINATION).
3. The application calls WFS_CMD_IPM_ACTION_ITEM to have the pre-defined actions executed.
4. The application optionally calls WFS_CMD_IPM_GET_NEXT_ITEM to have a single flow for devices with multi-feed and without. The flow continues with the next step.
5. The application then asks the customer if they have any more items to process.
6. If the customer has more items to deposit then flow continues at step 1.
7. When the customer is finished the application calls WFS_CMD_IPM_MEDIA_IN_END to terminate the transaction.

3. References

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

4. Info Commands

4.1 WFS_INF_IPM_STATUS

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

Input Param None.

Output Param LPWFSIPMSTATUS lpStatus;

```
typedef struct _wfs_ipm_status
     {
     WORD                fwDevice;
     WORD                wAcceptor;
     WORD                wMedia;
     WORD                wToner;
     WORD                wInk;
     WORD                wFrontImageScanner;
     WORD                wBackImageScanner;
     WORD                wMICRReader;
     WORD                wStacker;
     WORD                wReBuncher;
     WORD                wMediaFeeder;
     LPWFSIPMPOS         *lppPositions;
     DWORD               dwGuidLights[WFS_IPM_GUIDLIGHTS_SIZE];
     LPSTR               lpszExtra;
     WORD                wDevicePosition;
     USHORT              usPowerSaveRecoveryTime;
} WFSIPMSTATUS, *LPWFSIPMSTATUS;
```

fwDevice
Specifies the state of the device as one of the following values:

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

wAcceptor
Supplies the state of the overall acceptor media bins as one of the following values:

Value	Meaning
WFS_IPM_ACCBINOK	All media bins present are in a good state.

WFS_IPM_ACCBINSTATE	One of the media bins present is in an abnormal state. The acceptor is operational, but one or more of the media bins is in a high, full or inoperative condition. Items can still be accepted into at least one of the media bins. The status of the bins can be obtained through the WFS_INF_IPM_MEDIA_BIN_INFO command.
WFS_IPM_ACCBINSTOP	Due to media bin problem accepting is impossible. No items can be accepted because all of the media bins are in a full or in an inoperative condition.
WFS_IPM_ACCBINUNKNOWN	Due to a hardware error or other condition, the state of the media bins cannot be determined.

wMedia
Specifies the state of the media as one of the following values:

Value	Meaning
WFS_IPM_MEDIAPRESENT	Media is present in the device.
WFS_IPM_MEDIANOTPRESENT	Media is not present in the device.
WFS_IPM_MEDIAJAMMED	Media is jammed in the device.
WFS_IPM_MEDIANOTSUPP	The capability to report the state of the media is not supported by the device.
WFS_IPM_MEDIAUNKNOWN	The state of the media cannot be determined with the device in its current state.
WFS_IPM_MEDIAPOSITION	Media is at one or more of the input, output and refused positions.

wToner
Specifies the state of the toner or ink supply or the state of the ribbon of the endorser as one of the following values:

Value	Meaning
WFS_IPM_TONERFULL	The toner or ink supply is full or the ribbon is OK.
WFS_IPM_TONERLOW	The toner or ink supply is low or the print contrast with a ribbon is weak.
WFS_IPM_TONEROUT	The toner or ink supply is empty or the print contrast with a ribbon is not sufficient any more.
WFS_IPM_TONERNOTSUPP	The physical device does not support endorsing or the capability to report the status of the toner/ink is not supported by the device.
WFS_IPM_TONERUNKNOWN	Status of toner or ink supply or the ribbon cannot be determined with the device in its current state.

wInk
Specifies the status of the stamping ink in the device as one of the following values:

Value	Meaning
WFS_IPM_INKFULL	Ink supply in the device is full.
WFS_IPM_INKLOW	Ink supply in the device is low.
WFS_IPM_INKOUT	Ink supply in the device is empty.
WFS_IPM_INKNOTSUPP	The physical device does not support stamping or the capability to report the status of the stamp ink supply is not supported by the device.
WFS_IPM_INKUNKNOWN	Status of the stamping ink supply cannot be determined with the device in its current state.

wFrontImageScanner
Specifies the status of the image scanner that captures images of the front of the media items. This value can be one of the following values:

Value	Meaning
WFS_IPM_SCANNEROK	The front scanner is OK.
WFS_IPM_SCANNERFADING	The front scanner performance is degraded.
WFS_IPM_SCANNERINOP	The front scanner is inoperative.
WFS_IPM_SCANNERNOTSUPP	The physical device has no front scanner or the capability to report the status of the front scanner is not supported by the device.
WFS_IPM_SCANNERUNKNOWN	Status of the front scanner cannot be determined with the device in its current state.

wBackImageScanner
Specifies the status of the image scanner that captures images of the back of the media items. This value can be one of the following values:

Value	Meaning
WFS_IPM_SCANNEROK	The back scanner is OK.
WFS_IPM_SCANNERFADING	The back scanner performance is degraded.
WFS_IPM_SCANNERINOP	The back scanner is inoperative.
WFS_IPM_SCANNERNOTSUPP	The physical device has no back scanner or the capability to report the status of the back scanner is not supported by the device.
WFS_IPM_SCANNERUNKNOWN	Status of the back scanner cannot be determined with the device in its current state.

wMICRReader
Specifies the status of the MICR code line reader as one of the following values:

Value	Meaning
WFS_IPM_MICROK	The MICR code line reader is OK.
WFS_IPM_MICRFADING	The MICR code line reader performance is degraded.
WFS_IPM_MICRINOP	The MICR code line reader is inoperative.
WFS_IPM_MICRNOTSUPP	The physical device has no MICR code line reader or the capability to report the status of the MICR code line reader is not supported by the device.
WFS_IPM_MICRUNKNOWN	Status of the MICR code line reader cannot be determined with the device in its current state.

wStacker
Supplies the state of the stacker (also known as an escrow). The stacker is where the media items are held while the application decides what to do with them. This field can be one of the following values:

Value	Meaning
WFS_IPM_STACKEREMPTY	The stacker is empty.
WFS_IPM_STACKERNOTEMPTY	The stacker is not empty.
WFS_IPM_STACKERFULL	The stacker is full. This state is set if the number of media items on the stacker has reached *usMaxMediaOnStacker* or some physical limit has been reached.
WFS_IPM_STACKERINOP	The stacker is inoperative.
WFS_IPM_STACKERUNKNOWN	Due to a hardware error or other condition, the state of the stacker cannot be determined.
WFS_IPM_STACKERNOTSUPP	The physical device has no stacker or the capability to report the status of the stacker is not supported by the device.

wReBuncher
Supplies the state of the re-buncher (return stacker). The re-buncher is where media items are re-bunched ready for return to the customer. This field can be one of the following values:

Value	Meaning
WFS_IPM_REBUNCHEREMPTY	The re-buncher is empty.
WFS_IPM_REBUNCHERNOTEMPTY	The re-buncher is not empty.
WFS_IPM_REBUNCHERFULL	The re-buncher is full. This state is set if the number of media items on the re-buncher has reached its physical limit.
WFS_IPM_REBUNCHERINOP	The re-buncher is inoperative.
WFS_IPM_REBUNCHERUNKNOWN	Due to a hardware error or other condition, the state of the re-buncher cannot be determined.
WFS_IPM_REBUNCHERNOTSUPP	The physical device has no re-buncher or the capability to report the status of the re-buncher is not supported by the device.

wMediaFeeder
Supplies the state of the media feeder. This value can be one of the following values:

Value	Meaning
WFS_IPM_FEEDEREMPTY	The media feeder is empty.
WFS_IPM_FEEDERNOTEMPTY	The media feeder is not empty.
WFS_IPM_FEEDERINOP	The media feeder is inoperative.
WFS_IPM_FEEDERUNKNOWN	Due to a hardware error or other condition, the state of the media feeder cannot be determined.
WFS_IPM_FEEDERNOTSUPP	The physical device has no media feeder or the capability to report the status of the media feeder is not supported by the device.

lppPositions
Pointer to a NULL-terminated array of pointers to WFSIPMPOS structures. There is one for each of the three logical position types.

lppPositions[WFS_IPM_POSINPUT]
Points to a WFSIPMPOS structure that specifies the status of the input position. This pointer must not be NULL.

lppPositions[WFS_IPM_POSOUTPUT]
Points to a WFSIPMPOS structure that specifies the status of the output position. This pointer must not be NULL.

lppPositions[WFS_IPM_POSREFUSED]
Points to a WFSIPMPOS structure that specifies the status of the refused position. This pointer must not be NULL.

```
typedef struct _wfs_ipm_pos
     {
     WORD                wShutter;
     WORD                wPositionStatus;
     WORD                wTransport;
     WORD                wTransportMediaStatus;
     } WFSIPMPOS, *LPWFSIPMPOS;
```

wShutter
Specifies the state of the shutter as one of the following values:

Value	Meaning
WFS_IPM_SHTCLOSED	The shutter is closed.
WFS_IPM_SHTOPEN	The shutter is open.
WFS_IPM_SHTJAMMED	The shutter is jammed.
WFS_IPM_SHTUNKNOWN	Due to a hardware error or other condition, the state of the shutter cannot be determined.

WFS_IPM_SHTNOTSUPPORTED	The physical device has no shutter or shutter state reporting is not supported.

wPositionStatus
The status of the input or output position as one of the following values:

Value	Meaning
WFS_IPM_PSEMPTY	The position is empty.
WFS_IPM_PSNOTEMPTY	The position is not empty.
WFS_IPM_PSUNKNOWN	Due to a hardware error or other condition, the state of the position cannot be determined.
WFS_IPM_PSNOTSUPPORTED	The device is not capable of reporting whether or not items are at the position.

wTransport
Specifies the state of the transport mechanism as one of the following values:

Value	Meaning
WFS_IPM_TPOK	The transport is in a good state.
WFS_IPM_TPINOP	The transport is inoperative due to a hardware failure or media jam.
WFS_IPM_TPUNKNOWN	Due to a hardware error or other condition, the state of the transport cannot be determined.
WFS_IPM_TPNOTSUPPORTED	The physical device has no transport or transport state reporting is not supported.

wTransportMediaStatus
Returns information regarding items which may be present on the transport as one of the following values:

Value	Meaning
WFS_IPM_TPMEDIAEMPTY	The transport is empty.
WFS_IPM_TPMEDIANOTEMPTY	The transport is not empty.
WFS_IPM_TPMEDIAUNKNOWN	Due to a hardware error or other condition it is not known whether there are items on the transport.
WFS_IPM_TPMEDIANOTSUPPORTED	The device is not capable of reporting whether or not items are on the transport.

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

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

Value	Meaning	Type
WFS_IPM_GUIDANCE_NOT_AVAILABLE	The status is not available.	A
WFS_IPM_GUIDANCE_OFF	The light is turned off.	A
WFS_IPM_GUIDANCE_SLOW_FLASH	The light is blinking slowly.	B
WFS_IPM_GUIDANCE_MEDIUM_FLASH	The light is blinking medium frequency.	B
WFS_IPM_GUIDANCE_QUICK_FLASH	The light is blinking quickly.	B
WFS_IPM_GUIDANCE_CONTINUOUS	The light is turned on continuous (steady).	B
WFS_IPM_GUIDANCE_RED	The light is red.	C
WFS_IPM_GUIDANCE_GREEN	The light is green.	C
WFS_IPM_GUIDANCE_YELLOW	The light is yellow.	C
WFS_IPM_GUIDANCE_BLUE	The light is blue.	C
WFS_IPM_GUIDANCE_CYAN	The light is cyan.	C
WFS_IPM_GUIDANCE_MAGENTA	The light is magenta.	C
WFS_IPM_GUIDANCE_WHITE	The light is white.	C

dwGuidLights [WFS_IPM_GUIDANCE_MEDIAIN]
Specifies the state of the guidance light indicator on the bunch media in position.

dwGuidLights [WFS_IPM_GUIDANCE_MEDIAOUT]
Specifies the state of the guidance light indicator on the bunch media out position.

dwGuidLights [WFS_IPM_GUIDANCE_MEDIAREFUSED]
Specifies the state of the guidance light indicator on the bunch media refused position.

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

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

Value	Meaning
WFS_IPM_DEVICEINPOSITION	The device is in its normal operating position, or is fixed in place and cannot be moved.
WFS_IPM_DEVICENOTINPOSITION	The device has been removed from its normal operating position.
WFS_IPM_DEVICEPOSUNKNOWN	Due to a hardware error or other condition, the position of the device cannot be determined.
WFS_IPM_DEVICEPOSNOTSUPP	The physical device does not have the capability of detecting the position.

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

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

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

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

1. Report the value as unknown.
2. Report the value as a general h/w error.
3. Report the value as the last known value.

4.2 WFS_INF_IPM_CAPABILITIES

Description This command is used to request device capability information.

Input Param None.

Output Param LPWFSIPMCAPS lpCaps;

```
typedef struct _wfs_ipm_caps
     {
     WORD                 wClass;
     WORD                 fwType;
     BOOL                 bCompound;
     USHORT               usMaxMediaOnStacker;
     LPWFSIPMPRINTSIZE    lpPrintSize;
     BOOL                 bStamp;
     BOOL                 bRescan;
     BOOL                 bPresentControl;
     BOOL                 bApplicationRefuse;
     WORD                 fwRetractLocation;
     WORD                 fwResetControl;
     BOOL                 bRetractCountsItems;
     WORD                 fwImageType;
     WORD                 fwFrontImageColorFormat;
     WORD                 fwBackImageColorFormat;
     WORD                 fwFrontScanColor;
     WORD                 wDefaultFrontScanColor;
     WORD                 fwBackScanColor;
     WORD                 wDefaultBackScanColor;
     WORD                 fwCodelineFormat;
     WORD                 fwDataSource;
     WORD                 fwInsertOrientation;
     LPWFSIPMPOSCAPS      *lppPositions;
     DWORD                dwGuidLights[WFS_IPM_GUIDLIGHTS_SIZE];
     LPSTR                lpszExtra;
     BOOL                 bPowerSaveControl;
} WFSIPMCAPS, *LPWFSIPMCAPS;
```

wClass
Specifies the logical service class as WFS_SERVICE_CLASS_IPM.

fwType
Specifies the type(s) of the physical device driven by the logical service, as one of the following values:

Value	Meaning
WFS_IPM_TYPESINGLEMEDIAINPUT	Device accepts a single media item from the customer.
WFS_IPM_TYPEBUNCHMEDIAINPUT	Device accepts a bunch of media items from the customer.

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

usMaxMediaOnStacker
Specifies the maximum number of media items that the stacker can hold (zero if the device does not have a stacker). If the device has a bunch media input capability and the stacker is not present or has a capacity of one then the application must process each item inserted sequentially as described in section 2.2.1.

lpPrintSize
Pointer to a WFSIPMPRINTSIZE structure, NULL if device has no printing capabilities. If the media item is inserted in one of the orientations specified in *fwInsertOrientation*, the Service Provider will print on the back side of the media. If the media item is inserted in a different orientation to those specified in *fwInsertOrientation* then printing may occur on the front side, upside down or both.

```
typedef struct _wfs_ipm_print_size
     {
     WORD                wRows;
     WORD                wCols;
     } WFSIPMPRINTSIZE, *LPWFSIPMPRINTSIZE;
```

wRows
Specifies the maximum number of rows of text that can be printed on a media item. This value is zero if printing is not supported. This value is one for single line printers.

wCols
Specifies the maximum number of characters that can be printed on a row. This value is zero if printing is not supported.

bStamp
Specifies whether the device has stamping capabilities. If the media item is inserted in one of the orientations specified in *fwInsertOrientation*, the Service Provider will stamp on the front side of the media. If the media item is inserted in a different orientation to those specified in *fwInsertOrientation* then stamping may occur on the back, upside down or both.

bRescan
Specifies whether the device has the capability to either physically rescan media items after they have been inserted into the device or is able to generate any image supported by the device during the WFS_CMD_IPM_READ_IMAGE command (regardless of the images requested during the WFS_CMD_IPM_MEDIA_IN command). If TRUE then the item can be re-scanned or the images can be generated using the parameters passed in the WFS_CMD_IPM_READ_IMAGE command. If FALSE then all images required (various color, file format, bit depth) must be gathered during execution of the WFS_CMD_IPM_MEDIA_IN command.

bPresentControl
Specifies how the presenting of media items is controlled during the WFS_CMD_IPM_MEDIA_IN_END and WFS_CMD_IPM_MEDIA_IN_ROLLBACK commands. If set to TRUE the presenting is controlled implicitly by the Service Provider. If set to FALSE the presenting must be controlled explicitly by the application using the WFS_CMD_IPM_PRESENT_MEDIA command. This field is always set to TRUE if the device has no shutter. This field applies to all output positions.

bApplicationRefuse
Specifies if the Service Provider supports the WFS_CMD_IPM_MEDIA_IN mode where the application decides to accept or refuse each media item that has successfully been accepted by the device. If this value is TRUE then the Service Provider supports this mode. If this value is FALSE then the Service Provider does not support this mode (or the device does not have a stacker).

fwRetractLocation
Specifies the locations to which the media can be retracted using the WFS_CMD_IPM_RETRACT_MEDIA command, as a combination of the following bit-flags (zero if retract is not supported):

Value	Meaning
WFS_IPM_CTRLRETRACTTOBIN	Retract the media to a retract bin.
WFS_IPM_CTRLRETRACTTOTRANSPORT	Retract the media to the transport.
WFS_IPM_CTRLRETRACTTOSTACKER	Retract the media to the stacker.
WFS_IPM_CTRLRETRACTTOREBUNCHER	Retract the media to the re-buncher.

fwResetControl
Specifies the manner in which the media can be handled on WFS_CMD_IPM_RESET, as a combination of the following bit-flags:

Value	Meaning
WFS_IPM_RESETEJECT	Eject the media.
WFS_IPM_RESETRETRACTTOBIN	Retract the media to retract bin.
WFS_IPM_RESETRETRACTTOTRANSPORT	Retract the media to the transport.
WFS_IPM_RESETRETRACTTOREBUNCHER	Retract the media to the re-buncher.

bRetractCountsItems
This field only applies to retract media bins. It specifies whether the bin reports the number of items retracted into the bin or just the number of retract operations. If TRUE then *ulCount* and *ulMediaInCount* include the number of media items retracted and the *ulMaximumItems* value defines when the threshold event is generated. If FALSE then *ulCount* and *ulMediaInCount* do not contain the number of media items retracted but *ulRetractOperations* reports the number of retract operations. In this case the *ulMaximumRetractOperations* defines when the threshold event will be generated.

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

Value	Meaning
WFS_IPM_IMAGETIF	The device can return scanned images in TIFF 6.0 format.
WFS_IPM_IMAGEWMF	The device can return scanned images in WMF (Windows Metafile) format.
WFS_IPM_IMAGEBMP	The device can return scanned images in windows BMP format.
WFS_IPM_IMAGEJPG	The device can return scanned images in JPG format.

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

Value	Meaning
WFS_IPM_IMAGECOLORBINARY	The device can return scanned images in binary.
WFS_IPM_IMAGECOLORGRAYSCALE	The device can return scanned images in gray scale.
WFS_IPM_IMAGECOLORFULL	The device can return scanned images in full color.

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

Value	Meaning
WFS_IPM_IMAGECOLORBINARY	The device can return scanned images in binary.
WFS_IPM_IMAGECOLORGRAYSCALE	The device can return scanned images in gray scale.
WFS_IPM_IMAGECOLORFULL	The device can return scanned images in full color.

fwFrontScanColor
Specifies the front image scan colors supported by this device and individually controllable by the application. Scan colors are used to enhance the scanning results on colored scan media. This value is specified as a combination of the following flags (zero if selection of scan colors is not supported):

Value	Meaning
WFS_IPM_SCANCOLORRED	The device can return images scanned with red light.
WFS_IPM_SCANCOLORGREEN	The device can return images scanned with green light.
WFS_IPM_SCANCOLORBLUE	The device can return images scanned with blue light.
WFS_IPM_SCANCOLORYELLOW	The device can return images scanned with yellow light.
WFS_IPM_SCANCOLORWHITE	The device can return images scanned with white light.

wDefaultFrontScanColor
Specifies the default front image color format used by this device (i.e. when not explicitly set), as one of the following values:

Value	Meaning
WFS_IPM_SCANCOLORRED	The default color is red light.
WFS_IPM_SCANCOLORGREEN	The default color is green light.
WFS_IPM_SCANCOLORBLUE	The default color is blue light.
WFS_IPM_SCANCOLORYELLOW	The default color is yellow light.
WFS_IPM_SCANCOLORWHITE	The default color is white light.

fwBackScanColor
Specifies the back image scan colors supported by this device and individually controllable by the application. Scan colors are used to enhance the scanning results on colored scan media. This value is specified as a combination of the following flags (zero if selection of scan colors is not supported):

Value	Meaning
WFS_IPM_SCANCOLORRED	The device can return images scanned with red light.
WFS_IPM_SCANCOLORGREEN	The device can return images scanned with green light.
WFS_IPM_SCANCOLORBLUE	The device can return images scanned with blue light.
WFS_IPM_SCANCOLORYELLOW	The device can return images scanned with yellow light.
WFS_IPM_SCANCOLORWHITE	The device can return images scanned with white light.

wDefaultBackScanColor
Specifies the default front image color format used by this device (i.e. when not explicitly set), as one of the following values:

Value	Meaning
WFS_IPM_SCANCOLORRED	The default color is red light.
WFS_IPM_SCANCOLORGREEN	The default color is green light.
WFS_IPM_SCANCOLORBLUE	The default color is blue light.
WFS_IPM_SCANCOLORYELLOW	The default color is yellow light.
WFS_IPM_SCANCOLORWHITE	The default color is white light.

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

Value	Meaning
WFS_IPM_CODELINECMC7	The device can read MICR CMC7 code lines.
WFS_IPM_CODELINEE13B	The device can read MICR E13B code lines.
WFS_IPM_CODELINEOCR	The device can read code lines using Optical Character Recognition.

fwDataSource
Specifies the reading/imaging capabilities supported by this device, as a combination of the following flags (zero if not supported):

Value	Meaning
WFS_IPM_IMAGEFRONT	The device can scan the front image of the document.
WFS_IPM_IMAGEBACK	The device can scan the back image of the document.
WFS_IPM_CODELINE	The device can recognize the code line.

fwInsertOrientation
Specifies the media item insertion orientations supported by the Service Provider such that hardware features such as MICR reading, endorsing and stamping will be aligned with the correct edges and sides of the media item. Devices may still return code lines and images even if one of these orientations is not used during media insertion. If the media items are inserted in one of the orientations defined in this capability then any printing or stamping will be on the correct side of the media item. If the media is inserted in a different orientation then any printing or stamping may be on the wrong side, upside down or both. This value is reported based on the customer's perspective. This value is a combination of the following values:

Value	Meaning
WFS_IPM_INSCODELINERIGHT	The media item should be inserted short edge first with the code line to the right.
WFS_IPM_INSCODELINELEFT	The media item should be inserted short edge first with the code line to the left.
WFS_IPM_INSCODELINEBOTTOM	The media item should be inserted long edge first with the code line to the bottom.
WFS_IPM_INSCODELINETOP	The media item should be inserted long edge first with the code line to the top.
WFS_IPM_INSFACEUP	The media item should be inserted with the front of the media item facing up.
WFS_IPM_INSFACEDOWN	The media item should be inserted with the front of the media item facing down.

lppPositions
Pointer to a NULL-terminated array of pointers to WFSIPMPOSCAPS structures. There is one structure for each of the three logical position types.

lppPositions[WFS_IPM_POSINPUT]
Points to a WFSIPMPOSCAPS structure that specifies the capabilities of the input position. This pointer must not be NULL.

lppPositions[WFS_IPM_POSOUTPUT]
Points to a WFSIPMPOSCAPS structure that specifies the capabilities of the output position. This pointer must not be NULL.

lppPositions[WFS_IPM_POSREFUSED]
Points to a WFSIPMPOSCAPS structure that specifies the capabilities of the refused position. This pointer must not be NULL.

```
typedef struct _wfs_ipm_pos_caps
     {
     BOOL                 bItemsTakenSensor;
     BOOL                 bItemsInsertedSensor;
     WORD                 fwRetractAreas;
     } WFSIPMPOSCAPS, *LPWFSIPMPOSCAPS;
```

bItemsTakenSensor
Specifies whether or not the described position can detect when items at the exit position are taken by the user. If set to TRUE the Service Provider generates an accompanying WFS_SRVE_IPM_MEDIA_TAKEN event. If set to FALSE this event is not generated. This field relates to output and refused positions, so will always be set to FALSE for input positions.

bItemsInsertedSensor
Specifies whether the described position has the ability to detect when items have been inserted by the user. If set to TRUE the Service Provider generates an accompanying WFS_SRVE_IPM_MEDIAINSERTED event. If set to FALSE this event is not generated. This field relates to all input positions, so will always be set to FALSE for output and refuse positions.

fwRetractAreas
Specifies the areas to which items may be retracted from this position. This field will be set to a combination of the following flags:

Value	Meaning
WFS_IPM_CTRLRETRACTTOBIN	Can retract items in this position to a retract bin.
WFS_IPM_CTRLRETRACTTOTRANSPORT	Can retract items in this position to the transport.
WFS_IPM_CTRLRETRACTTOSTACKER	Can retract items in this position to the stacker.
WFS_IPM_CTRLRETRACTTOREBUNCHER	Can retract items in this position to the re-buncher.

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

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

Value	Meaning	Type
WFS_IPM_GUIDANCE_NOT_AVAILABLE	There is no guidance light control available at this position.	A
WFS_IPM_GUIDANCE_OFF	The light can be off.	B
WFS_IPM_GUIDANCE_SLOW_FLASH	The light can blink slowly.	B
WFS_IPM_GUIDANCE_MEDIUM_FLASH	The light can blink medium frequency.	B
WFS_IPM_GUIDANCE_QUICK_FLASH	The light can blink quickly.	B
WFS_IPM_GUIDANCE_CONTINUOUS	The light can be continuous (steady).	B
WFS_IPM_GUIDANCE_RED	The light can be red.	C
WFS_IPM_GUIDANCE_GREEN	The light can be green.	C
WFS_IPM_GUIDANCE_YELLOW	The light can be yellow.	C
WFS_IPM_GUIDANCE_BLUE	The light can be blue.	C
WFS_IPM_GUIDANCE_CYAN	The light can be cyan.	C
WFS_IPM_GUIDANCE_MAGENTA	The light can be magenta.	C
WFS_IPM_GUIDANCE_WHITE	The light can be white.	C

dwGuidLights [WFS_IPM_GUIDANCE_MEDIAIN]
Specifies whether the guidance light indicator on the bunch media in position is available.

dwGuidLights [WFS_IPM_GUIDANCE_MEDIAOUT]
Specifies whether the guidance light indicator on the bunch media out position is available.

dwGuidLights [WFS_IPM_GUIDANCE_MEDIAREFUSED]
Specifies whether the guidance light indicator on the bunch media refused position is available.

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

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

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

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

4.3 WFS_INF_IPM_CODELINE_MAPPING

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

Input Param LPWFSIPMCODELINEMAPPING lpCodelineMapping;

```
typedef struct _wfs_ipm_codeline_mapping
     {
     WORD                 wCodelineFormat;
     } WFSIPMCODELINEMAPPING, *LPWFSIPMCODELINEMAPPING;
```

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

Value	Meaning
WFS_IPM_CODELINECMC7	Report the CMC7 mapping.
WFS_IPM_CODELINEE13B	Report the E13B mapping.

Output Param LPWFSIPMCODELINEMAPPINGOUT lpCodelineMapping;

```
typedef struct _wfs_ipm_codeline_mapping_out
     {
     WORD                 wCodelineFormat;
     LPWFSIPMXDATA        lpxCharMapping;
     } WFSIPMCODELINEMAPPINGOUT, *LPWFSIPMCODELINEMAPPINGOUT;
```

wCodeLineFormat
Specifies the code line format that is being reported. This field can be one of the following values:

Value	Meaning
WFS_IPM_CODELINECMC7	Report the CMC7 mapping.
WFS_IPM_CODELINEE13B	Report the E13B mapping.

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

E13B

Index	0	1	2	3	4
Symbol that byte value represents	⑆	⑇	⑈	⑉	N/A
Meaning	Transit	Amount	On Us	Dash	Reject / Unreadable

CMC7

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

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

Comments For code lines defined in the OCR-A font then the ASCII codes will conform to Figure E1 in ANSI X3.17-1981. For code lines defined in the OCR-B font then the ASCII codes will conform to Figure C2 in ANSI X3.49-1975. In both these cases unrecognized characters will be reported as the REJECT code, 0x1A.

4.4 WFS_INF_IPM_MEDIA_BIN_INFO

Description This command is used to obtain information about the status and contents of the media bins that can be used by IPM commands. This command does not report bins that can only be used by the other interface on a compound device.

Input Param None.

Output Param LPWFSIPMMEDIABININFO lpMediaBinInfo;

```
typedef struct _wfs_ipm_media_bin_info
     {
     USHORT              usCount;
     LPWFSIPMMEDIABIN    *lppMediaBin;
     } WFSIPMMEDIABININFO, *LPWFSIPMMEDIABININFO;
```

usCount
Number of WFSIPMMEDIABIN structures returned in *lppMediaBin*.

lppMediaBin
Pointer to an array of pointers to WFSIPMMEDIABIN structures.

```
typedef struct _wfs_ipm_media_bin
     {
     USHORT              usBinNumber;
     LPSTR               lpstrPositionName;
     WORD                fwType;
     WORD                wMediaType;
     LPSTR               lpstrBinID;
     ULONG               ulMediaInCount;
     ULONG               ulCount;
     ULONG               ulRetractOperations;
     BOOL                bHardwareSensors;
     ULONG               ulMaximumItems;
     ULONG               ulMaximumRetractOperations;
     USHORT              usStatus;
     LPSTR               lpszExtra;
     } WFSIPMMEDIABIN, *LPWFSIPMMEDIABIN;
```

usBinNumber
Index number of the media bin structure. Each structure has a unique number starting with a value of one (1) for the first structure, and incrementing by one for each subsequent structure.

lpstrPositionName
The physical position name where the bin is inserted.

fwType
Specifies the type of media bin as one or more of the following values:

Value	Meaning
WFS_IPM_TYPEMEDIAIN	Media bin. This type of bin can be specified as a destination for media items.
WFS_IPM_TYPERETRACT	Retract bin. This type of bin can be specified as a destination for the WFS_CMD_IPM_RETRACT_MEDIA command.

wMediaType
Specifies the type of media the media bin takes. This value is specified as one of the following values:

Value	Meaning
WFS_IPM_MEDIATYPIPM	The media bin takes media items via the IPM device class only.
WFS_IPM_MEDIATYPCOMPOUND	The media bin takes media from the IPM device class and from another device class (e.g. CIM).

lpstrBinID
An application defined Media Bin Identifier.

ulMediaInCount
Count of items that have entered the media bin as a result of operations on the IPM interface. This counter is incremented whenever media enters the media bin for any reason as a result of an operation initiated through the IPM interface. This value is persistent. On a retract-only bin, if the device cannot count media during a retract operation this value will be zero.

ulCount
Total number of media in the media bin (including items that may have been added via a compound device interface). If the bin is a shared bin with a compound device interface then this value may not be the same as the value of *ulMediaInCount.* On a retract-only bin, if the device cannot count media during a retract operation this value will be zero.

ulRetractOperations
The number of Retract operations via WFS_CMD_IPM_RETRACT_MEDIA, WFS_CMD_IPM_RESET and error recovery where media is moved to the bin. This value is persistent.

bHardwareSensors
A capability that specifies whether or not the threshold event, WFS_USRE_IPM_MEDIABINTHRESHOLD (WFS_IPM_STATMBHIGH), can be generated based on hardware sensors in the device. If this value is TRUE then threshold events may be generated based on hardware sensors. If applications want the threshold event to be based on the h/w sensors then the threshold limits, *ulMaximumItems* and *ulMaximumRetractOperations*, must be set to zero. If they are not set to zero then the h/w sensors are ignored.

ulMaximumItems
When *ulCount* reaches this value the threshold event WFS_USRE_IPM_MEDIABINTHRESHOLD (WFS_IPM_STATMBHIGH) will be generated.

ulMaximumRetractOperations
When *ulRetractOperations* reaches this value the threshold event WFS_USRE_IPM_MEDIABINTHRESHOLD (WFS_IPM_STATMBHIGH) will be generated. This value is zero if the bin is not a retract bin (i.e. does not contain the WFS_IPM_TYPERETRACT value in the *fwType* field).

usStatus
Describes the status of the media bin as one of the following values:

Value	Meaning
WFS_IPM_STATMBOK	The media bin is in a good state.
WFS_IPM_STATMBFULL	The media bin is full.
WFS_IPM_STATMBHIGH	The media bin is almost full (threshold).
WFS_IPM_STATMBINOP	The media bin is inoperative.
WFS_IPM_STATMBMISSING	The media bin is missing.
WFS_IPM_STATMBUNKNOWN	The media bin is unknown.

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

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

Comments In the case where the media bin allows both deposit and retract operations but cannot count the number of media items retracted, then the threshold event will be generated when either *ulRetractOperations* or *ulCount* reaches its associated threshold value. Since these counts are unrelated but the media items are being placed in the same bin the threshold event is very in-accurate and should be disabled in favor of h/w sensors.

4.5 WFS_INF_IPM_TRANSACTION_STATUS

Description This command is used to request the status of the current or last media-in transaction. A Media-In-Transaction consists of one or more WFS_CMD_IPM_MEDIA_IN commands. A Media-In transaction is initiated by the WFS_CMD_IPM_MEDIA_IN command and remains active until the transaction is either confirmed through WFS_CMD_IPM_MEDIA_IN_END, or cancelled by WFS_CMD_IPM_MEDIA_IN_ROLLBACK, WFS_CMD_IPM_RETRACT_MEDIA or WFS_CMD_IPM_RESET. Multiple calls to WFS_CMD_IPM_MEDIA_IN can be made while a transaction is active to obtain additional items from the customer.

Input Param None.

Output Param LPWFSIPMTRANSSTATUS lpTransStatus;

```
typedef struct _wfs_ipm_trans_status
     {
     WORD                   wMediaInTransaction;
     USHORT                 usMediaOnStacker;
     USHORT                 usLastMediaInTotal;
     USHORT                 usLastMediaAddedToStacker;
     USHORT                 usTotalItems;
     USHORT                 usTotalItemsRefused;
     USHORT                 usTotalBunchesRefused;
     LPWFSIPMMEDIASTATUS    *lppMediaInfo;
     LPSTR                  lpszExtra;
     } WFSIPMTRANSSTATUS, *LPWFSIPMTRANSSTATUS;
```

wMediaInTransaction
Status of the Media-In transaction. This value is specified as one of the following values:

Value	Meaning
WFS_IPM_MITOK	The Media-In transaction completed successfully.
WFS_IPM_MITACTIVE	There is a Media-In transaction active.
WFS_IPM_MITROLLBACK	The Media-In transaction was successfully rolled back.
WFS_IPM_MITROLLBACKAFTERDEPOSIT	The Media-In transaction was successfully rolled back after some items had been deposited to a bin. This value only applies to devices without a stacker.
WFS_IPM_MITRETRACT	The Media-In transaction ended with the items being successfully retracted.
WFS_IPM_MITFAILURE	The Media-In transaction failed as the result of a device failure.
WFS_IPM_MITUNKNOWN	The state of the Media-In transaction is unknown.
WFS_IPM_MITRESET	The Media-In transaction ended as the result of a WFS_CMD_IPM_RESET command.

usMediaOnStacker
Contains the total number of media items currently on the stacker (including *usLastMediaAddedToStacker*), or WFS_IPM_MEDIANUMBERUNKNOWN if it is unknown. This count only applies to devices with stackers and is persistent.

usLastMediaInTotal
Contains the number of media items processed by the last WFS_CMD_IPM_MEDIA_IN command, or WFS_IPM_MEDIANUMBERUNKNOWN if it is unknown. This count is not modified for bunches of items which are refused as a single entity. This count only applies to devices with stackers and is persistent.

usLastMediaAddedToStacker
Contains the number of media items on the stacker successfully accepted by the last WFS_CMD_IPM_MEDIA_IN command, or WFS_IPM_MEDIANUMBERUNKNOWN if it is unknown. This count only applies to devices with stackers and is persistent.

The number of media items refused during the last command can be determined by *usLastMediaInTotal* - *usLastMediaAddedToStacker*. This is only possible if these values contain known values, and would not include bunches of items refused as a single entity.

usTotalItems
The total number of items that have been allocated a MediaID during the whole of the current transaction (if a transaction is active) or last transaction (if no transaction is active). This count does not include refused items, is WFS_IPM_MEDIANUMBERUNKNOWN if it is unknown, and is persistent.

usTotalItemsRefused
Contains the total number of refused items during the execution of the whole transaction. This count does not include bunches of items which are refused as a single entity without being processed as single items, is WFS_IPM_MEDIANUMBERUNKNOWN if it is unknown, and is persistent.

usTotalBunchesRefused
Contains the total number of refused bunches of items that were not processed as single items, is WFS_IPM_MEDIANUMBERUNKNOWN if it is unknown, and is persistent.

lppMediaInfo
Pointer to a NULL-terminated array of pointers to WFSIPMMEDIASTATUS structures. This array contains details of the media items processed during the current or last transaction (depending on the value of *wMediaInTransaction*). The array contains one element for every item that has been allocated a Media ID (i.e. items that have been reported to the application). If there are no media items then the *lppMediaInfo* is NULL. The LPWFSIPMIMAGEDATA structure is described in the WFS_CMD_IMP_READ_IMAGE command section. The media info is available until a new transaction is started with the WFS_CMD_IPM_MEDIA_IN command. The media location information may be updated after a transaction is completed, e.g. if media that was presented to the customer is subsequently retracted. The media info is persistent.

```
typedef struct _wfs_ipm_mediastatus
     {
     USHORT               usMediaID;
     WORD                 wMediaLocation;
     USHORT               usBinNumber;
     ULONG                ulCodelineDataLength;
     LPBYTE               lpbCodelineData;
     WORD                 wMagneticReadIndicator;
     LPWFSIPMIMAGEDATA    *lppImage;
     WORD                 fwInsertOrientation;
     LPWFSIPMMEDIASIZE    lpMediaSize;
     WORD                 wMediaValidity;
     WORD                 wCustomerAccess;
} WFSIPMMEDIASTATUS, *LPWFSIPMMEDIASTATUS;
```

usMediaID
Specifies the sequence number (starting from 1) of the media item.

wMediaLocation
Specifies the location of the media item as one of the following values:

Value	Meaning
WFS_IPM_LOCATION_DEVICE	The media item is inside the device in some position other than a bin.
WFS_IPM_LOCATION_BIN	The media item is in a bin. The bin number is defined by *usBinNumber*.
WFS_IPM_LOCATION_CUSTOMER	The media has been returned to the customer.
WFS_IPM_LOCATION_UNKNOWN	The media item location is unknown.

usBinNumber
If *wMediaLocation* is WFS_IPM_LOCATION_BIN then this field contains the bin number where the media was stored.

ulCodelineDataLengh
Count of bytes of the following *lpbCodelineData*.

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

wMagneticReadIndicator
Specifies the type of technology used to read a MICR code line. This value is specified as one of the following values:

Value	Meaning
WFS_IPM_MRI_MICR	The MICR code line was read using MICR technology and MICR characters were present.
WFS_IPM_MRI_NOT_MICR	The MICR code line was NOT read using MICR technology.
WFS_IPM_MRI_NO_MICR	The MICR code line was read using MICR technology and no magnetic characters were read.
WFS_IPM_MRI_UNKNOWN	It is unknown how the MICR code line was read.
WFS_IPM_MRI_NOTMICRFORMAT	The code line is not a MICR format code line.
WFS_IPM_MRI_NOT_READ	No code line was read.

lppImage
Pointer to a NULL-terminated array of pointers to WFSIPMIMAGEDATA structures. If there is no image data then *lppImage* will be set to NULL. If the Service Provider has determined the orientation of the media (i.e. *fwInsertOrientation* is not set to WFS_IPM_INSUNKNOWN), then all images returned are in the standard orientation and the images will match the image source requested by the application. This means that images will be returned with the code line at the bottom, and the image of the front and rear of the media item will be returned in the structures associated with the WFS_IPM_IMAGEFRONT and WFS_IPM_IMAGEBACK image sources respectively.

fwInsertOrientation
This value reports how the media item was actually inserted into the input position (from the customers perspective). This value is either WFS_IPM_INSUNKNOWN or a combination of one value from type A and one value from type B.

Value	Meaning	Type
WFS_IPM_INSUNKNOWN	The orientation of the inserted media is unknown.	N/A
WFS_IPM_INSCODELINERIGHT	The code line is to the right.	A
WFS_IPM_INSCODELINELEFT	The code line is to the left.	A
WFS_IPM_INSCODELINEBOTTOM	The code line is to the bottom.	A
WFS_IPM_INSCODELINETOP	The code line is to the top.	A
WFS_IPM_INSFACEUP	The front of the media (the side with the code line) is facing up.	B
WFS_IPM_INSFACEDOWN	The front of the media (the side with the code line) is facing down.	B

lpMediaSize
Pointer to a WFSIPMMEDIASIZE structure that specifies the size of the media item. *lpMediaSize* is NULL if the device does not support media size measurement.

```
typedef struct _wfs_ipm_media_size
     {
     ULONG               ulSizeX;
     ULONG               ulSizeY;
     } WFSIPMMEDIASIZE, *LPWFSIPMMEDIASIZE;
```

ulSizeX
Specifies the width of the media in millimeters, or zero if unknown.

ulSizeY
Specifies the height of the media in millimeters, or zero if unknown.

wMediaValidity
Media items may have special security features which can be detected by the device. This field specifies whether the media item is suspect or valid, allowing the application a choice in how to further process a media item that could not be confirmed as being valid. This value is specified as one of the following values:

Value	Meaning
WFS_IPM_ITEMOK	The media item is valid.
WFS_IPM_ITEMSUSPECT	The validity of the media item is suspect.
WFS_IPM_ITEMUNKNOWN	The validity of the media item is unknown.
WFS_IPM_ITEMNOVALIDATION	No specific security features were evaluated.

wCustomerAccess
Specifies if the media item has been in customer access since it was first deposited, e.g. it has been retracted from a position with customer access. This value is specified as one of the following values:

Value	Meaning
WFS_IPM_ACCESSUNKNOWN	It is not known if the media item has been in a position with customer access.
WFS_IPM_ACCESSCUSTOMER	The media item has been in a position with customer access.
WFS_IPM_ACCESSNONE	The media item has not been in a position with customer access.

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

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

Comments None.

5. Execute Commands

5.1 WFS_CMD_IPM_MEDIA_IN

Description This command accepts media into the device from the input position.

A Media-In-Transaction consists of one or more WFS_CMD_IPM_MEDIA_IN commands. A Media-In transaction is initiated by the first WFS_CMD_IPM_MEDIA_IN command and remains active until the transaction is either confirmed through WFS_CMD_IPM_MEDIA_IN_END, or cancelled by WFS_CMD_IPM_MEDIA_IN_ROLLBACK, WFS_CMD_IPM_RETRACT_MEDIA or WFS_CMD_IPM_RESET. Multiple calls to WFS_CMD_IPM_MEDIA_IN can be made while a transaction is active to obtain additional items from the customer. If a media-in transaction is active (i.e. the transaction status is WFS_IPM_MITACTIVE) when a WFS_CMD_IPM_MEDIA_IN command is successfully cancelled, or the command times-out then the transaction remains active.

When the command is executed, if there is no media in the input slot then the device is enabled for media entry and the WFS_EXEE_IPM_NOMEDIA event is generated when the device is ready to accept media. When the customer inserts the media a WFS_EXEE_IPM_MEDIAINSERTED event is generated and media processing begins. If media is already present at the input slot then a WFS_EXEE_IPM_MEDIAINSERTED event is generated and media processing begins immediately.

The WFS_EXEE_IPM_MEDIADATA event delivers the code line and all requested image data during execution of this command. One event is generated for each media item scanned by this command. The WFS_EXEE_IPM_MEDIADATA event is not generated for refused media items.

A failure during processing a single media item does not mean that the command has failed even if some or all of the media are refused by the media reader. In this case the command will return WFS_SUCCESS and one or more WFS_EXEE_IPM_MEDIAREFUSED events will be sent to report the reasons why the items have been refused.

Refused items are not presented back to the customer with this command. The WFS_EXEE_IPM_MEDIAREFUSED event indicates whether or not media must be returned to the customer before further media movement commands can be executed. If the WFS_EXEE_IPM_MEDIAREFUSED event indicates that the media must be returned then the application must use the WFS_CMD_IPM_PRESENT_MEDIA command to return the refused items. If the event does not indicate the application must return the media items then the application can still elect to return the media items using the WFS_CMD_IPM_PRESENT_MEDIA command or instead allow the refused items to be returned during the WFS_CMD_IPM_MEDIA_IN_END or WFS_CMD_IPM_MEDIA_IN_ROLLBACK commands.

If there is no stacker on the device or *bApplicationRefuse* is TRUE then just one of the media items inserted are processed by this command, and therefore the command completes as soon as the last image for the first item is produced or when the first item is automatically refused. If there is a stacker on the device then the command completes when the last image for the last item is produced or when the last item is refused.

Input Param LPWFSIPMMEDIAINREQUEST lpMediaInRequest;

```
typedef struct _wfs_ipm_media_in_request
     {
     WORD                  wCodelineFormat;
     LPWFSIPMIMAGEREQUEST *lppImage;
     USHORT                usMaxMediaOnStacker;
     BOOL                  bApplicationRefuse;
     } WFSIPMMEDIAINREQUEST, *LPWFSIPMMEDIAINREQUEST;
```

wCodelineFormat
Specifies the code line format, as one of following flags (if zero no code line data is required):

Value	Meaning
WFS_IPM_CODELINECMC7	Read CMC7 code line.
WFS_IPM_CODELINEE13B	Read E13B code line.
WFS_IPM_CODELINEOCR	Read code line using OCR.

lppImage
Pointer to a NULL-terminated array of pointers to WFSIPMIMAGEREQUEST structures. The array contains one pointer to a WFSIPMIMAGEREQUEST structure for every image that should be read for each media item. If *lppImage* is NULL no images are required.

```
typedef struct _wfs_ipm_image_request
     {
     WORD                wImageSource;
     WORD                wImageType;
     WORD                wImageColorFormat;
     WORD                wImageScanColor;
     LPSTR               lpszImagePath;
     } WFSIPMIMAGEREQUEST, *LPWFSIPMIMAGEREQUEST;
```

wImageSource
Specifies the source as one of the following values:

Value	Meaning
WFS_IPM_IMAGEFRONT	The returned image is for the front of the media item.
WFS_IPM_IMAGEBACK	The returned image is for the back of the media item.

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

Value	Meaning
WFS_IPM_IMAGETIF	The returned image is in TIFF 6.0 format. The output file name will have the .tif extension appended to the filename.
WFS_IPM_IMAGEWMF	The returned image is in WMF (Windows Metafile) format. The output file name will have the .wmf extension appended to the filename.
WFS_IPM_IMAGEBMP	The returned image is in Windows BMP format. The output file name will have the .bmp extension appended to the filename.
WFS_IPM_IMAGEJPG	The returned image is in JPG format. The output file name will have the .jpg extension appended to the filename.

wImageColorFormat
Specifies the color format of the requested image as one of the following values:

Value	Meaning
WFS_IPM_IMAGECOLORBINARY	The scanned images has to be returned in binary (image contains two colors, usually the colors black and white).
WFS_IPM_IMAGECOLORGRAYSCALE	The scanned images has to be returned in gray scale (image contains multiple gray colors).
WFS_IPM_IMAGECOLORFULL	The scanned images has to be returned in full color (image contains colors like red, green, blue etc.).

wImageScanColor
Selects the color that should be used to scan the image. The value is specified as one of the following values:

Value	Meaning
WFS_IPM_SCANCOLORDEFAULT	Select the default color for the side of the item being scanned.
WFS_IPM_SCANCOLORRED	Select the red scan color.
WFS_IPM_SCANCOLORGREEN	Select the green scan color.
WFS_IPM_SCANCOLORBLUE	Select the blue scan color.

WFS_IPM_SCANCOLORYELLOW	Select the yellow scan color.
WFS_IPM_SCANCOLORWHITE	Select the white scan color.

lpszImagePath
Specifies the full path name of the folder where the image will be stored, e.g. "C:\TEMP". The actual file name for the image produced will be vendor specific. The name used is reported in the event containing the item data for each media item. The Service Provider may re-use file names from the start of each media in transaction, so applications must manage the file lifetime as required. If NULL is provided for this parameter then the command will be rejected with the WFS_ERR_INVALID_DATA error. If the folder does not exist or cannot be accessed by the Service Provider then the command will be rejected with the WFS_ERR_IPM_FILEIOERROR error.

usMaxMediaOnStacker
Maximum number of media items allowed on the stacker during the Media-In transaction. This value is used to limit the total number of media items on the stacker. When this limit is reached all further media items will be refused and a WFS_EXEE_IPM_MEDIAREFUSED event will be generated reporting WFS_IPM_REFUSED_STAKERFULL. This value cannot exceed the value reported in the *usMaxMediaOnStacker* field of the Capabilities or the Service Provider will return a WFS_ERR_INVALID_DATA error. If this value is zero then the maximum number of items allowed on the stacker reported in the *usMaxMediaOnStacker* field of the Capabilities will be used. This value must be the same during all calls to the WFM_CMD_IPM_MEDIA_IN command within a single Media-In transaction or the Service Provider will return a WFS_ERR_INVALID_DATA error. This value is ignored on devices without stackers.

bApplicationRefuse
Specifies if the application wants to make the decision to accept or refuse each media item that has successfully been accepted by the device. If this value is TRUE then the application must decide to accept or refuse each item. The application must use the WFS_CMD_IPM_ACCEPT_ITEM and WFS_CMD_IPM_GET_NEXT_ITEM commands in a sequential manner to process the bunch of media inserted during the WFS_CMD_IPM_MEDIA_IN command. If this value is FALSE then any decision on whether an item should be refused is left to the device/Service Provider. This value must have the same value within all calls to WFS_CMD_IPM_MEDIA_IN within a transaction. This value must be FALSE when the *bApplicationRefuse* capability is FALSE.

Output Param LPWFSIPMMEDIAIN lpMediaIn;

```
typedef struct _wfs_ipm_media_in
     {
     USHORT              usMediaOnStacker;
     USHORT              usLastMedia;
     USHORT              usLastMediaOnStacker;
     WORD                wMediaFeeder;
     } WFSIPMMEDIAIN, *LPWFSIPMMEDIAIN;
```

usMediaOnStacker
Contains the total number of media items on the stacker (including *usLastMediaOnStacker*), or WFS_IPM_MEDIANUMBERUNKNOWN if it is unknown. This count only applies to devices with stackers.

usLastMedia
Contains the number of media items processed by this instance of the command execution, or WFS_IPM_MEDIANUMBERUNKNOWN if it is unknown. This count only applies to devices with stackers.

usLastMediaOnStacker
Contains the number of media items on the stacker successfully accepted by this instance of the command execution, or WFS_IPM_MEDIANUMBERUNKNOWN if it is unknown. This count only applies to devices with stackers.

The number of refused media items can be determined by *usLastMedia* - *usLastMediaOnStacker*. This is only possible if these values contain known values, and would not be possible if a bunch of items were refused as a single entity.

wMediaFeeder
Supplies the state of the media feeder. This value indicates if there are items on the media feeder waiting processing via the WFS_CMD_IPM_GET_NEXT_ITEM command. This value can be one of the following values:

Value	Meaning
WFS_IPM_FEEDERISEMPTY	The media feeder is empty.
WFS_IPM_FEEDERISNOTEMPTY	The media feeder is not empty.
WFS_IPM_FEEDERISNOTSUPPORTED	The physical device has no media feeder.

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

Value	Meaning
WFS_ERR_IPM_STACKERFULL	The internal stacker is already full or has already reached the limit specified as an input parameter. No media items can be accepted.
WFS_ERR_IPM_SHUTTERFAIL	Open or close of the shutter failed due to manipulation or hardware error.
WFS_ERR_IPM_MEDIAJAMMED	The media is jammed.
WFS_ERR_IPM_FILEIOERROR	Directory does not exist or File IO error while storing the image to the hard disk.
WFS_ERR_IPM_REFUSEDITEMS	Programming Error: refused items that must be returned via WFS_CMD_IPM_PRESENT_MEDIA have not been presented (see *bPresentRequired* in the WFS_EXEE_IPM_MEDIAREFUSED event parameters).
WFS_ERR_IPM_ALLBINSFULL	All media bins are full so no further items can be accepted.
WFS_ERR_IPM_SCANNERINOP	Only images were requested by the application and these cannot be obtained because the image scanner is inoperative.
WFS_ERR_IPM_MICRINOP	Only MICR data was requested by the application and it cannot be obtained because the MICR reader is inoperative.
WFS_ERR_IPM_POSITIONNOTEMPTY	One of the input/output/refused positions is not empty and items cannot be inserted until the media items in the position are removed.
WFS_ERR_IPM_FEEDERNOTEMPTY	The media feeder is not empty. This only applies when the WFS_CMD_IPM_GET_NEXT_ITEM command should be used to retrieve the next media item.
WFS_ERR_IPM_MEDIAREJECTED	The media was rejected before it was fully inserted within the device. The WFS_EXEE_IPM_MEDIAREJECTED execute event is posted with the details. The device is still operational.
WFS_ERR_IPM_FEEDERINOPERATIVE	The media feeder is inoperative.
WFS_ERR_IPM_MEDIAPRESENT	Media from a previous transaction is present in the device when an attempt to start a new media-in transaction was made. The media must be cleared before a new transaction can be started.

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

Value	Meaning
WFS_EXEE_IPM_NOMEDIA	No media is present in the input position and the device is ready for the customer to insert media.

WFS_EXEE_IPM_MEDIAINSERTED	Media has been inserted into the device.
WFS_EXEE_IPM_MEDIAREFUSED	Media has been refused.
WFS_EXEE_IPM_MEDIADATA	Delivers media data (images and code line) during the command.
WFS_EXEE_IPM_MEDIAREJECTED	The media has been rejected before it was fully inserted within the device and has been presented back to the user. It is available at the input position. When the media is removed, a WFS_SRVE_IPM_MEDIATAKEN event will be generated.
WFS_USRE_IPM_SCANNERTHRESHOLD	The imaging scanner is fading or inoperative. Note that this event is sent only once, at the point at which the status changes.
WFS_USRE_IPM_MICRTHRESHOLD	The MICR reader performance is degraded or the reader is inoperative. Note that this event is sent only once, at the point at which the status changes.

Comments None.

5.2 WFS_CMD_IPM_MEDIA_IN_END

Description This command ends a Media-In transaction. If media items are on the stacker as a result of a WFS_CMD_IPM_MEDIA_IN command, the actions pre-defined through WFS_CMD_IPM_PRINT_TEXT (stamping & endorsing) and WFS_CMD_IPM_GET_IMAGE_AFTER_PRINT are executed and then these media items are moved to the destination defined by the WFS_CMD_IPM_SET_DESTINATION command. If no action (print, stamp, re-scan) has been pre-defined then the items are just moved to their destination. If the destination has not been set for a media item then the Service Provider will decide which bin to put the item into. If no items are in the device the command will complete with WFS_ERR_IPM_NOMEDIAPRESENT and the transaction status will be set to WFS_IPM_MITOK.

The way in which media is returned to the customer as a result of this command is defined by the *bPresentControl* flag reported by WFS_INF_IPM_CAPABILITIES. If the *bPresentControl* flag is FALSE the application must call WFS_CMD_IPM_PRESENT_MEDIA to present the media items to be returned as a result of this command. If the *bPresentControl* flag is TRUE the Service Provider presents any returned items implicitly and the application does not need to call WFS_CMD_IPM_PRESENT_MEDIA.

If items have been refused and the WFS_IPM_EXEE_MEDIAREFUSED event has indicated that the items must be returned (i.e. *bPresentRequired* is TRUE) then these items must be returned using WFS_CMD_IPM_PRESENT_MEDIA before WFS_CMD_IPM_MEDIA_IN_END is issued, otherwise a WFS_ERR_IPM_REFUSEDITEMS error will be returned. If items have been refused and the WFS_IPM_EXEE_MEDIAREFUSED event has indicated that the items do not need to be returned (i.e. *bPresentRequired* is FALSE) then the WFS_CMD_IPM_MEDIA_IN_END causes any refused items which have not yet been returned to the customer (via WFS_CMD_IPM_PRESENT_MEDIA) to be returned along with any items that the application has selected to return to the customer (via WFS_CMD_IPM_SET_DESTINATION). Even if all items are being deposited, previously refused items will be returned to the customer by this command. The WFS_EXEE_IPM_MEDIAPRESENTED event(s) inform the application of the output position where the media has been presented.

This command completes when all the media items have been put into their specified bins and in the case where media is returned to the customer as a result of this command, after the last bunch of media items to be returned to the customer has been presented, but before the last bunch is taken.

The Media-In transaction is ended even if this command does not complete successfully.

Input Param None.

Output Param LPWFSIPMMEDIAINEND lpMediaInEnd;

```
typedef struct _wfs_ipm_media_in_end
     {
     USHORT               usItemsReturned;
     USHORT               usItemsRefused;
     USHORT               usBunchesRefused;
     LPWFSIPMMEDIABININFO lpMediaBinInfo;
     } WFSIPMMEDIAINEND, *LPWFSIPMMEDIAINEND;
```

usItemsReturned
Contains the number of media items that were returned to the customer by application selection through the WFS_CMD_IPM_SET_DESTINATION command during the current transaction. This does not include items that were refused.

usItemsRefused
Contains the total number of items automatically returned to the customer during the execution of the whole transaction. This count does not include bunches of items which are refused as a single entity without being processed as single items.

usBunchesRefused
Contains the total number of refused bunches of items that were automatically returned to the customer without being processed as single items.

lpMediaBinInfo
Pointer to a WFSIPMMEDIABININFO structure containing a list of media bins that have taken media during the current transaction. For a description of the WFSIPMMEDIABININFO structure see the definition of the WFS_INF_IPM_MEDIA_BIN_INFO command. This pointer must always point to a WFSIPMMEDIABININFO structure, it cannot be NULL. The structure returned only contains data related to the current transaction, i.e. *ulCount* and *ulMediaInCount* define the number of media in the media bin for this transaction.

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

Value	Meaning
WFS_ERR_IPM_NOMEDIAPRESENT	No media is present in the device.
WFS_ERR_IPM_SHUTTERFAIL	Open or close of the shutter failed due to manipulation or hardware error.
WFS_ERR_IPM_MEDIAJAMMED	The media is jammed.
WFS_ERR_IPM_MEDIABINERROR	A problem occurred with a media bin. A WFS_EXEE_IPM_MEDIABINERROR event will be sent with the details.
WFS_ERR_IPM_POSITIONNOTEMPTY	One of the input/output/refused positions is not empty.
WFS_ERR_IPM_SEQUENCEINVALID	Programming error. Invalid command sequence, e.g. this command was executed when there was no active transaction.
WFS_ERR_IPM_REFUSEDITEMS	Programming Error: refused items that must be returned via WFS_CMD_IPM_PRESENT_MEDIA have not been presented (see *bPresentRequired* in the WFS_EXEE_IPM_MEDIAREFUSED event parameters).
WFS_ERR_IPM_FEEDERNOTEMPTY	The media feeder is not empty.

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

Value	Meaning
WFS_USRE_IPM_MEDIABINTHRESHOLD	A threshold condition has occurred in one of the media bins.
WFS_EXEE_IPM_MEDIADATA	Delivers media images scanned after the item has been printed.
WFS_EXEE_IPM_MEDIABINERROR	A problem occurred with a media bin.
WFS_USRE_IPM_TONERTHRESHOLD	The toner or ink supply is low or empty or the printing contrast with ribbon is weak or not sufficient, operator intervention is required. Note that this event is sent only once, at the point at which the supply becomes low or empty. It is sent with WFS_IPM_TONERLOW or WFS_IPM_TONEROUT status.
WFS_USRE_IPM_INKTHRESHOLD	The stamp ink supply is low or empty, operator intervention is required. Note that this event is sent only once, at the point at which the supply becomes low or empty. It is sent with WFS_IPM_INKLOW or WFS_IPM_INKOUT status.
WFS_USRE_IPM_SCANNERTHRESHOLD	The imaging scanner is fading or inoperative. Note that this event is sent only once, at the point at which the status changes.
WFS_SRVE_IPM_MEDIATAKEN	The media has been taken by the user.
WFS_EXEE_IPM_MEDIAPRESENTED	Media has been presented for removal.

Comments None.

5.3 WFS_CMD_IPM_MEDIA_IN_ROLLBACK

Description This command ends a Media-In transaction. All media that is in the device as a result of WFS_CMD_IPM_MEDIA_IN commands is returned to the customer. Nothing is printed on the media. If no items are in the device the command will complete with WFS_ERR_IPM_NOMEDIAPRESENT and the transaction status will be set to WFS_IPM_MITROLLBACK.

The way in which media is returned to the customer as a result of this command is defined by the *bPresentControl* flag reported by WFS_INF_IPM_CAPABILITIES. If the *bPresentControl* flag is FALSE the application must call WFS_CMD_IPM_PRESENT_MEDIA to present the media items to be returned as a result of this command. If the *bPresentControl* flag is TRUE the Service Provider presents any returned items implicitly and the application does not need to call WFS_CMD_IPM_PRESENT_MEDIA.

If items have been refused and the WFS_IPM_EXEE_MEDIAREFUSED event has indicated that the items must be returned (i.e. *bPresentRequired* is TRUE) then these items must be returned using WFS_CMD_IPM_PRESENT_MEDIA before WFS_CMD_IPM_MEDIA_IN_ROLLBACK is issued, otherwise a WFS_ERR_IPM_REFUSEDITEMS error will be returned. If items have been refused and the WFS_IPM_EXEE_MEDIAREFUSED event has indicated that the items do not need to be returned (i.e. *bPresentRequired* is FALSE) then the WFS_CMD_IPM_MEDIA_IN_ROLLBACK causes any refused items which have not yet been returned to the customer (via WFS_CMD_IPM_PRESENT_MEDIA) to be returned along with any items that are returned as a result of the rollback. The WFS_EXEE_IPM_MEDIAPRESENTED event(s) inform the application of the output position where the media has been presented.

In the case where media is returned to the customer as a result of this command, this command completes when the last bunch of media items to be returned to the customer has been presented, but before the last bunch is taken.

The Media-In transaction is ended even if this command does not complete successfully.

Input Param None.

Output Param None.

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

Value	Meaning
WFS_ERR_IPM_NOMEDIAPRESENT	No media is present in the device.
WFS_ERR_IPM_MEDIAJAMMED	The media is jammed.
WFS_ERR_IPM_SEQUENCEINVALID	Programming error. Invalid command sequence (e.g. no transaction active).
WFS_ERR_IPM_SHUTTERFAIL	Open or close of the shutter failed due to manipulation or hardware error.
WFS_ERR_IPM_POSITIONNOTEMPTY	The output position is not empty.
WFS_ERR_IPM_REFUSEDITEMS	Programming Error: refused items that must be returned via WFS_CMD_IPM_PRESENT_MEDIA have not been presented (see *bPresentRequired* in the WFS_EXEE_IPM_MEDIAREFUSED event parameters).

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

Value	Meaning
WFS_SRVE_IPM_MEDIATAKEN	The media has been taken by the user.
WFS_EXEE_IPM_MEDIAPRESENTED	Media has been presented for removal.

Comments None.

5.4 WFS_CMD_IPM_READ_IMAGE

Description On devices where items can be physically re-scanned or all the supported image formats can be generated during this command (regardless of the images requested during the WFS_CMD_IPM_MEDIA_IN command), i.e. where *bRescan* capability is TRUE, then this command is used to obtain additional images and/or re-read the code line for media already in the device.

On devices where *bRescan* capability is FALSE, this command is used to re-retrieve an image or code line that was initially obtained when the media was initially processed (e.g. during WFS_CMD_IPM_MEDIA_IN or WFS_CMD_IPM_GET_NEXT_ITEM). In this case, all images required must have been previously been requested during the WFS_CMD_IPM_MEDIA_IN command.

The media has to be inserted using the command WFS_CMD_IPM_MEDIA_IN. If no media is present the command returns the error code WFS_ERR_IPM_NOMEDIAPRESENT.

Input Param LPWFSIPMREADIMAGEIN lpReadImageIn;

```
typedef struct _wfs_ipm_read_image_request
     {
     USHORT               usMediaID;
     WORD                 wCodelineFormat;
     LPWFSIPMIMAGEREQUEST *lppImage;
     } WFSIPMREADIMAGEIN, *LPWFSIPMREADIMAGEIN;
```

usMediaID
Specifies the sequence number of a media item. Valid IDs are 1 to the maximum media ID assigned within the transaction.

wCodelineFormat
Specifies the code line format, as a one of following flags (zero if source not selected):

Value	Meaning
WFS_IPM_CODELINECMC7	Read CMC7 code line.
WFS_IPM_CODELINEE13B	Read E13B code line.
WFS_IPM_CODELINEOCR	Read code line using OCR.

lppImage
Pointer to a NULL-terminated array of pointers to WFSIPMIMAGEREQUEST structures describing the required images. If NULL no images are required.

```
typedef struct _wfs_ipm_image_request
     {
     WORD                 wImageSource;
     WORD                 wImageType;
     WORD                 wImageColorFormat;
     WORD                 wImageScanColor;
     LPSTR                lpszImagePath;
     } WFSIPMIMAGEREQUEST, *LPWFSIPMIMAGEREQUEST;
```

wImageSource
Specifies the source as one of the following values:

Value	Meaning
WFS_IPM_IMAGEFRONT	The returned image is for the front of the media item.
WFS_IPM_IMAGEBACK	The returned image is for the back of the media item.

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

Value	Meaning
WFS_IPM_IMAGETIF	The returned image is in TIFF 6.0 format. The output file name will have the .tif extension appended to the filename.

WFS_IPM_IMAGEWMF	The returned image is in WMF (Windows Metafile) format. The output file name will have the .wmf extension appended to the filename.
WFS_IPM_IMAGEBMP	The returned image is in Windows BMP format. The output file name will have the .bmp extension appended to the filename.
WFS_IPM_IMAGEJPG	The returned image is in JPG format. The output file name will have the .jpg extension appended to the filename.

wImageColorFormat
Specifies the color format of the requested image as one of the following values:

Value	Meaning
WFS_IPM_IMAGECOLORBINARY	The scanned images has to be returned in binary (image contains two colors, usually the colors black and white).
WFS_IPM_IMAGECOLORGRAYSCALE	The scanned images has to be returned in gray scale (image contains multiple gray colors).
WFS_IPM_IMAGECOLORFULL	The scanned images has to be returned in full color (image contains colors like red, green, blue etc.).

wImageScanColor
Selects the scan color. The value is specified as one of the following values:

Value	Meaning
WFS_IPM_SCANCOLORDEFAULT	Select the default scan color for the side of the item being scanned.
WFS_IPM_SCANCOLORRED	Select the red scan color.
WFS_IPM_SCANCOLORGREEN	Select the green scan color.
WFS_IPM_SCANCOLORBLUE	Select the blue scan color.
WFS_IPM_SCANCOLORYELLOW	Select the yellow scan color.
WFS_IPM_SCANCOLORWHITE	Select the white scan color.

lpszImagePath
Specifies the full path and file name where the image will be stored. If NULL is provided for this parameter then the command will be rejected with the WFS_ERR_INVALID_DATA error. If the folder does not exist or cannot be accessed by the Service Provider then the command will be rejected with the WFS_ERR_IPM_FILEIOERROR error.

Output Param LPWFSIPMMEDIADATA lpMediaData;

```
typedef struct _wfs_ipm_mediadata
     {
     USHORT               usMediaID;
     ULONG                ulCodelineDataLength;
     LPBYTE               lpbCodelineData;
     WORD                 wMagneticReadIndicator;
     LPWFSIPMIMAGEDATA    *lppImage;
     WORD                 fwInsertOrientation;
     LPWFSIPMMEDIASIZE    lpMediaSize;
     WORD                 wMediaValidity;
     } WFSIPMMEDIADATA, *LPWFSIPMMEDIADATA;
```

usMediaID
Specifies the sequence number (starting from 1) of the media item.

ulCodelineDataLength
Count of bytes of the following *lpbCodelineData*.

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

wMagneticReadIndicator
Specifies the type of technology used to read a MICR code line.

Value	Meaning
WFS_IPM_MRI_MICR	The MICR code line was read using MICR technology and MICR characters were present.
WFS_IPM_MRI_NOT_MICR	The MICR code line was NOT read using MICR technology.
WFS_IPM_MRI_NO_MICR	The MICR code line was read using MICR technology and no magnetic characters were read.
WFS_IPM_MRI_UNKNOWN	It is unknown how the MICR code line was read.
WFS_IPM_MRI_NOTMICRFORMAT	The code line is not a MICR format code line.
WFS_IPM_MRI_NOT_READ	No code line was read.

lppImage
Pointer to a NULL-terminated array of pointers to WFSIPMIMAGEDATA structures. If image data items have not been requested then *lppImage* will be set to NULL If the Service Provider has determined the orientation of the media (i.e. *fwInsertOrientation* is not set to WFS_IPM_INSUNKNOWN), then all images returned are in the standard orientation and the images will match the image source requested by the application. This means that images will be returned with the code line at the bottom, and the image of the front and rear of the media item will be returned in the structures associated with the WFS_IPM_IMAGEFRONT and WFS_IPM_IMAGEBACK image sources respectively.

```
typedef struct _wfs_ipm_image_data
     {
     WORD                wImageSource;
     WORD                wImageType;
     WORD                wImageColorFormat;
     WORD                wImageScanColor;
     WORD                wImageStatus;
     LPSTR               lpszImageFile;
     } WFSIPMIMAGEDATA, *LPWFSIPMIMAGEDATA;
```

wImageSource
Specifies the source of the data returned by this item as one of the following values:

Value	Meaning
WFS_IPM_IMAGEFRONT	The returned image is for the front of the media item.
WFS_IPM_IMAGEBACK	The returned image is for the back of the media item.

wImageType
Specifies the format of the image returned by this item as one of the following values:

Value	Meaning
WFS_IPM_IMAGETIF	The returned image is in TIFF 6.0 format.
WFS_IPM_IMAGEWMF	The returned image is in WMF (Windows Metafile) format.
WFS_IPM_IMAGEBMP	The returned image is in Windows BMP format.
WFS_IPM_IMAGEJPG	The returned image is in JPG format.

wImageColorFormat
Specifies the color format of the image returned by this item as one of following flags:

Value	Meaning
WFS_IPM_IMAGECOLORBINARY	The scanned image is returned in binary format (image contains two colors, usually the colors black and white).
WFS_IPM_IMAGECOLORGRAYSCALE	The scanned image is returned in binary format (image contains multiple gray colors).
WFS_IPM_IMAGECOLORFULL	The scanned image is returned in full color (image contains colors like red, green, blue, etc.).

wImageScanColor
Specifies the scan color of the image returned by this item as one of following flags:

Value	Meaning
WFS_IPM_SCANCOLORRED	The image was scanned with red light.
WFS_IPM_SCANCOLORGREEN	The image was scanned with green light.
WFS_IPM_SCANCOLORBLUE	The image was scanned with blue light.
WFS_IPM_SCANCOLORYELLOW	The image was scanned with yellow light.
WFS_IPM_SCANCOLORWHITE	The image was scanned with white light.

wImageStatus
Status of the requested image data. Possible values are:

Value	Meaning
WFS_IPM_DATAOK	The data is OK.
WFS_IPM_DATASRCNOTSUPP	The data source or image attributes are not supported by the Service Provider, e.g. scan color not supported.
WFS_IPM_DATASRCMISSING	The requested image could not be obtained.

lpszImageFile
Specifies the full path and file name where the image is stored, e.g. “C:\Temp\FrontImage.bmp”. The path and file name used is selected by the input parameters.

fwInsertOrientation
This value reports how the media item was actually inserted into the input position (from the customers perspective). This value is either WFS_IPM_INSUNKNOWN or a combination of one value from type A and one value from type B.

Value	Meaning	Type
WFS_IPM_INSUNKNOWN	The orientation of the inserted media is unknown.	N/A
WFS_IPM_INSCODELINERIGHT	The code line is to the right.	A
WFS_IPM_INSCODELINELEFT	The code line is to the left.	A
WFS_IPM_INSCODELINEBOTTOM	The code line is to the bottom.	A
WFS_IPM_INSCODELINETOP	The code line is to the top.	A
WFS_IPM_INSFACEUP	The front of the media (the side with the code line) is facing up.	B
WFS_IPM_INSFACEDOWN	The front of the media (the side with the code line) is facing down.	B

lpMediaSize
Pointer to a WFSIPMMEDIASIZE structure that specifies the size of the media item. *lpMediaSize* is NULL if the device does not support media size measurement.

```
typedef struct _wfs_ipm_media_size
      {
      ULONG               ulSizeX;
      ULONG               ulSizeY;
      } WFSIPMMEDIASIZE, *LPWFSIPMMEDIASIZE;
```

ulSizeX
Specifies the width of the media in millimeters, or zero if unknown.

ulSizeY
Specifies the height of the media in millimeters, or zero if unknown.

wMediaValidity
Media items may have special security features which can be detected by the device. This field specifies whether the media item is suspect or valid, allowing the application a choice in how to further process a media item that could not be confirmed as being valid. This value is specified as one of the following values:

Value	Meaning
WFS_IPM_ITEMOK	The media item is valid.
WFS_IPM_ITEMSUSPECT	The validity of the media item is suspect.
WFS_IPM_ITEMUNKNOWN	The validity of the media item is unknown.
WFS_IPM_ITEMNOVALIDATION	No specific security features were evaluated.

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

Value	Meaning
WFS_ERR_IPM_MEDIAJAMMED	The media is jammed.
WFS_ERR_IPM_FILEIOERROR	Directory does not exist or File IO error while storing the image to the hard disk.
WFS_ERR_IPM_SCANNERINOP	Only images were requested by the application and these cannot be obtained because the image scanner is inoperative.
WFS_ERR_IPM_MICRINOP	Only MICR data was requested by the application and it cannot be obtained because the MICR reader is inoperative.
WFS_ERR_IPM_NOMEDIAPRESENT	No media is present in the device.
WFS_ERR_IPM_SEQUENCEINVALID	Programming error. Invalid command sequence.
WFS_ERR_IPM_INVALIDMEDIAID	The requested Media ID does not exist.

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

Value	Meaning
WFS_USRE_IPM_SCANNERTHRESHOLD	The imaging scanner is fading or inoperative. Note that this event is sent only once, at the point at which the status changes.
WFS_USRE_IPM_MICRTHRESHOLD	The MICR reader performance is degraded or the reader is inoperative. Note that this event is sent only once, at the point at which the status changes.

Comments None.

5.5 WFS_CMD_IPM_SET_DESTINATION

Description This command is used to predefine the destination of the specified media item. The media is not moved immediately by this command. On devices with stackers, the command WFS_CMD_IPM_MEDIA_IN_END transports the corresponding media item to the defined destination. On devices without stackers, the command WFS_CMD_IPM_ACTION_ITEM transports the corresponding media item to the defined destination.

The Service Provider will determine which bin to use for any items that have not had a destination set by the application.

Input Param LPWFSIPMSETDESTINATION lpSetDestination;

```
typedef struct _wfs_ipm_set_destination
     {
     USHORT              usMediaID;
     USHORT              usBinNumber;
     } WFSIPMSETDESTINATION, *LPWFSIPMSETDESTINATION;
```

usMediaID
Specifies the sequence number of a media item. Valid IDs are 1 to the maximum media ID assigned within the transaction. Zero selects all media on the stacker.

usBinNumber
Specifies the number of a media bin or zero to return the media items to the customer. The media bins that can accept deposited items can be obtained through the WFS_INF_IPM_MEDIA_BIN_INFO command.

Output Param None.

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

Value	Meaning
WFS_ERR_IPM_NOMEDIAPRESENT	No media is present in the device.
WFS_ERR_IPM_SEQUENCEINVALID	Programming error. Invalid command sequence.
WFS_ERR_IPM_INVALIDMEDIAID	The requested Media ID does not exist.
WFS_ERR_IPM_INVALIDBIN	The specified bin cannot take media, either it is a retract only bin or it is missing.
WFS_ERR_IPM_NOBIN	The specified bin does not exist.
WFS_ERR_IPM_MEDIABINFULL	The media bin is already full and no media can be placed in the specified bin.

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

Comments None.

5.6 WFS_CMD_IPM_PRESENT_MEDIA

Description This command is used to present media items to the customer.

Applications can use this command to return refused items without terminating the media-in transaction. This allows customers to correct the problem with the media item and re-insert during execution of a subsequent WFS_CMD_IPM_MEDIA_IN command.

This command is also used to return items after a WFS_CMD_IPM_MEDIA_IN_END or WFS_CMD_IPM_MEDIA_IN_ROLLBACK command when the *bPresentControl* flag reported by WFS_INF_IPM_CAPABILITIES is FALSE.

A WFS_EXEE_IPM_MEDIA_PRESENTED event is generated when media is presented and a WFS_SRVE_MEDIA_TAKEN event is generated when the media is taken (if the position has a taken sensor WFSIPMPOSCAPS.*bItemsTakenSensor*).

This command completes when the last bunch of media items to be returned to the customer has been presented, but before the last bunch is taken.

Input Param LPWFSIPMPRESENTMEDIA lpPresentMedia;

```
typedef struct _wfs_ipm_present_media
     {
     WORD                wPosition;
     } WFSIPMPRESENTMEDIA, *LPWFSIPMPRESENTMEDIA;
```

wPosition
Specifies the position where items are returned from as one of the following values:

Value	Meaning
WFS_IPM_REFUSE_INPUT	Items in the Input position are presented to the customer.
WFS_IPM_REFUSE_REFUSED	Items in the Refused Media Position are presented to the customer.
WFS_IPM_REFUSE_REBUNCHER	Items in the refuse/return re-buncher are presented to the customer.

If *wPosition* is zero then all refused items are returned from all positions in a sequence determined by the Service Provider. In general the media items in the input position should be returned before those in any other position.

Output Param None.

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

Value	Meaning
WFS_ERR_IPM_NOMEDIAPRESENT	The control action could not be completed because there is no media in the position specified.
WFS_ERR_IPM_SHUTTERFAIL	Open of the shutter failed due to manipulation or hardware error.
WFS_ERR_IPM_MEDIAJAMMED	The media is jammed.
WFS_ERR_IPM_SEQUENCEINVALID	Programming error. Invalid command sequence.
WFS_ERR_IPM_POSITIONNOTEMPTY	One of the input/output/refused positions is not empty.

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

Value	Meaning
WFS_SRVE_IPM_MEDIATAKEN	The media has been taken by the user.
WFS_EXEE_IPM_MEDIAPRESENTED	Media has been presented for removal.

Comments None.

5.7 WFS_CMD_IPM_RETRACT_MEDIA

Description The media is removed from its present position (media present in device, media entering, unknown position) and stored in the area specified in the input parameters.

A threshold event is sent if the high or full condition is reached as a result of this command. If the bin is already full and the command cannot be executed, an error is returned and the media remains in its present position.

This command ends the current media-in transaction.

If no items are in the device the command will complete with WFS_ERR_IPM_NOMEDIAPRESENT and the transaction status will be set to WFS_IPM_MITRETRACT.

Input Param LPWFSIPMRETRACTMEDIA lpRetractMedia;

If the application does not wish to specify a position it can set *lpRetractMedia* to NULL. In this case the Service Provider will determine where to move any items found.

```
typedef struct _wfs_ipm_retract_media
     {
     WORD                 wRetractLocation;
     USHORT               usBinNumber;
     } WFSIPMRETRACTMEDIA, *LPWFSIPMRETRACTMEDIA;
```

wRetractLocation
Specifies the location for the retracted media. See the *fwRetractLocation* capability to determine the supported locations. This field can take one of the following values:

Value	Meaning
WFS_IPM_CTRLRETRACTTOBIN	Retract the media to the retract bin specified in *usBinNumber*.
WFS_IPM_CTRLRETRACTTOTRANSPORT	Retract the media to the transport.
WFS_IPM_CTRLRETRACTTOSTACKER	Retract the media to the stacker.
WFS_IPM_CTRLRETRACTTOREBUNCHER	Retract the media to the re-buncher.

usBinNumber
If *wRetractLocation* is WFS_IPM_CTRLRETRACTTOBIN then this field contains the *usBinNumber* of the media bin where the media should be retracted to. This media bin must have a *fwType* field that includes the WFS_IPM_TYPERETRACT flag. If *wRetractLocation* is not WFS_IPM_CTRLRETRACTTOBIN then this field is ignored.

Output Param LPWFSIPMRETRACTMEDIAOUT lpRetractMediaOut;

```
typedef struct _wfs_ipm_retract_media_out
     {
     USHORT               usMedia;
     WORD                 wRetractLocation;
     USHORT               usBinNumber;
     } WFSIPMRETRACTMEDIAOUT, *LPWFSIPMRETRACTMEDIAOUT;
```

usMedia
Contains the number of media items retracted as a result of this command or WFS_IPM_MEDIANUMBERUNKNOWN if the number of items is unknown (e.g. device cannot count retracted items).

wRetractLocation
Contains the location of the retracted items as one of the following values:

Value	Meaning
WFS_IPM_CTRLRETRACTTOBIN	The media has been retracted to the bin specified in *usBinNumber*.
WFS_IPM_CTRLRETRACTTOTRANSPORT	The media has been retracted to the transport.
WFS_IPM_CTRLRETRACTTOSTACKER	The media has been retracted to the stacker.
WFS_IPM_CTRLRETRACTTOREBUNCHER	The media has been retracted to the re-buncher.

usBinNumber
The *usBinNumber* of the media bin where the items were retracted to. This value is zero if the *wRetractLocation* is not WFS_IPM_CTRLRETRACTTOBIN.

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

Value	Meaning
WFS_ERR_IPM_NOMEDIAPRESENT	No media present on retract. Either there was no media present (in a position to be retracted) when the command was called or the media was removed during the retract.
WFS_ERR_IPM_MEDIAJAMMED	The media is jammed.
WFS_ERR_IPM_STACKERFULL	The stacker or re-buncher is full.
WFS_ERR_IPM_INVALIDBIN	The specified bin cannot retract media.
WFS_ERR_IPM_NOBIN	The specified bin does not exist.
WFS_ERR_IPM_MEDIABINERROR	A problem occurred with a media bin. A WFS_EXEE_IPM_MEDIABINERROR event will be sent with the details.
WFS_ERR_IPM_SHUTTERFAIL	Open or close of the shutter failed due to manipulation or hardware error.
WFS_ERR_IPM_FOREIGNITEMSDETECTED	Foreign items have been detected in the input position.

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

Value	Meaning
WFS_USRE_IPM_MEDIABINTHRESHOLD	A threshold condition has occurred in the retract bin.
WFS_EXEE_IPM_MEDIABINERROR	A problem occurred with the retract bin.
WFS_SRVE_IPM_MEDIATAKEN	The media has been taken by the user.

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

5.8 WFS_CMD_IPM_PRINT_TEXT

Description This command is used to predefine the data that will be printed on a media item and nothing is printed during execution of this command. On devices with stackers the data is printed when the bunch is processed through the WFS_CMD_IPM_MEDIA_IN_END command. The request will not be performed if the bunch is returned with the WFS_CMD_IPM_MEDIA_ROLLBACK. On devices without stackers the data is printed when the WFS_CMD_IPM_ACTION_ITEM command is executed.

For devices that can print multiple lines each line is separated by a Carriage Return (Unicode 0x000D) and Line Feed (Unicode 0x000A) sequence.

The media has to be inserted before this command is called. If no media is present the command returns the error code WFS_ERR_IPM_NOMEDIAPRESENT.

Input Param LPWFSIPMPRINTTEXT lpPrintText;

```
typedef struct _wfs_ipm_print_text
     {
     USHORT               usMediaID;
     BOOL                 bStamp;
     LPWSTR               lpszPrintData;
     } WFSIPMPRINTTEXT, *LPWFSIPMPRINTTEXT;
```

usMediaID
Specifies the sequence number of a media item. Valid IDs are 1 to the maximum media ID assigned within the transaction. Zero selects all media on the stacker.

bStamp
Specifies whether the media will be stamped.

lpszPrintData
Specifies the UNICODE data that will be printed on the media item that is entered by the customer. If a UNICODE character is not supported by the device it will be replaced by a vendor dependent substitution character.

Output Param None.

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

Value	Meaning
WFS_ERR_IPM_TONEROUT	Toner or ink supply is empty or printing contrast with ribbon is not sufficient.
WFS_ERR_IPM_INKOUT	No stamping possible, stamping ink supply empty.
WFS_ERR_IPM_NOMEDIAPRESENT	No media is present in the device.
WFS_ERR_IPM_INVALIDMEDIAID	The requested Media ID does not exist.
WFS_ERR_IPM_SEQUENCEINVALID	Programming error. Invalid command sequence.

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

Comments None.

5.9 WFS_CMD_IPM_SET_MEDIA_BIN_INFO

Description This command is used to adjust information about the status and contents of the media bins present in the IPM.

This command generates the service event WFS_SRVE_IPM_MEDIABININFOCHANGED to inform applications that media bin information has been changed.

This command can only be used to change the application defined bin identifier, software counters and thresholds. All other fields in the input structure will be ignored.

The following fields of the WFSIPMMEDIABIN structure may be updated by this command:

lpstrBinID
ulCount
ulMediaInCount
ulRetractOperations
ulMaximumItems
ulMaximumRetractOperations

The WFS_EXEE_IPM_MEDIABINERROR event can be generated if there is problem accessing a media bin on systems that store media bin data on the bin hardware. This event can be generated when the command fails with a WFS_ERR_IPM_MEDIABINERROR error or completes with WFS_SUCCESS. WFS_SUCCESS will be reported when some media bin details are changed successfully but some fail. If no bins are changed WFS_ERR_IPM_MEDIABINERROR will be returned.

Input Param LPWFSIPMMEDIABININFO lpMediaBinInfo;

The LPWFSIPMMEDIABININFO structure is specified in the documentation of the WFS_INF_IPM_MEDIA_BIN_INFO command. All media bins must be included not just the media bins whose values are to be changed.

Output Param None.

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

Value	Meaning
WFS_ERR_IPM_INVALIDBIN	Invalid media bin.
WFS_ERR_IPM_MEDIABINERROR	A problem occurred with the media bins, no bin settings have been changed. The WFS_EXEE_IPM_MEDIABINERROR event will be report the error details.

Events In addition to the generic events defined in [Ref. 1], the following events can be generated as a result of this command:

Value	Meaning
WFS_USRE_IPM_MEDIABINTHRESHOLD	A threshold condition has been reached or cleared in one of the media bins.
WFS_SRVE_IPM_MEDIABININFOCHANGED	A media bin was updated as a result of this command.
WFS_EXEE_IPM_MEDIABINERROR	A problem occurred with a media bin. Note: This event can be generated even when the command completes with WFS_SUCCESS.

Comments None.

5.10 WFS_CMD_IPM_RESET

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

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

This command ends a Media-In transaction started by WFS_CMD_IPM_MEDIA_IN.

Input Param LPWFSIPMRESET lpReset;

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

```
typedef struct _wfs_ipm_reset
     {
     WORD                wMediaControl;
     USHORT              usBinNumber;
     } WFSIPMRESET, *LPWFSIPMRESET;
```

wMediaControl
Specifies the manner in which the media should be handled, as one of the following values:

Value	Meaning
WFS_IPM_RESETEJECT	Eject the media, i.e. return the media to the customer. Note that more than one position may be used to return media.
WFS_IPM_RESETRETRACTTOBIN	Retract the media to the retract bin as specified in *usBinNumber*.
WFS_IPM_RESETRETRACTTOTRANSPORT	Retract the media to the transport.
WFS_IPM_RESETRETRACTTOREBUNCHER	Retract the media to the re-buncher.

usBinNumber
Number of the retract bin the media is retracted to. It is only relevant if *dwMediaControl* equals WFS_IPM_CTRLRETRACTTOBIN. The numbers of available media bins can be obtained through the *usBinNumber* and *fwType* fields returned by the WFS_INF_IPM_MEDIA_BIN_INFO command.

Output Param None.

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

Value	Meaning
WFS_ERR_IPM_SHUTTERFAIL	Open or close of the shutter failed due to manipulation or hardware error.
WFS_ERR_IPM_MEDIAJAMMED	The media is jammed. Operator intervention is required.
WFS_ERR_IPM_MEDIABINERROR	A problem occurred with a media bin. A WFS_EXEE_IPM_MEDIABINERROR event will be sent with the details.
WFS_ERR_IPM_INVALIDBIN	The bin cannot accept retracted items.

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

Value	Meaning
WFS_SRVE_IPM_MEDIADETECTED	A media is detected in the device during a reset operation.

WFS_USRE_IPM_MEDIABINTHRESHOLD	A threshold condition has occurred in the retract bin.
WFS_EXEE_IPM_MEDIABINERROR	A problem occurred with the retract bin.
WFS_SRVE_IPM_MEDIATAKEN	The media has been taken by the user.
WFS_EXEE_IPM_MEDIAPRESENTED	Media has been presented for removal.

Comments None.

5.11 WFS_CMD_IPM_SET_GUIDANCE_LIGHT

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

Input Param LPWFSIPMSETGUIDLIGHT lpSetGuidLight;

```
typedef struct _wfs_ipm_set_guidlight
     {
     WORD                 wGuidLight;
     DWORD                dwCommand;
     } WFSIPMSETGUIDLIGHT, *LPWFSIPMSETGUIDLIGHT;
```

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

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

Value	Meaning	Type
WFS_IPM_GUIDANCE_OFF	The light indicator is turned off.	A
WFS_IPM_GUIDANCE_SLOW_FLASH	The light indicator is set to flash slowly.	B
WFS_IPM_GUIDANCE_MEDIUM_FLASH	The light indicator is set to flash medium frequency.	B
WFS_IPM_GUIDANCE_QUICK_FLASH	The light indicator is set to flash quickly.	B
WFS_IPM_GUIDANCE_CONTINUOUS	The light indicator is turned on continuously (steady).	B
WFS_IPM_GUIDANCE_RED	The light indicator color is set to red.	C
WFS_IPM_GUIDANCE_GREEN	The light indicator color is set to green.	C
WFS_IPM_GUIDANCE_YELLOW	The light indicator color is set to yellow.	C
WFS_IPM_GUIDANCE_BLUE	The light indicator color is set to blue.	C
WFS_IPM_GUIDANCE_CYAN	The light indicator color is set to cyan.	C
WFS_IPM_GUIDANCE_MAGENTA	The light indicator color is set to magenta.	C
WFS_IPM_GUIDANCE_WHITE	The light indicator color is set to white.	C

Output Param None.

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

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

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

Comments None.

5.12 WFS_CMD_IPM_GET_NEXT_ITEM

Description This command is used to get the next item from the multi-item feed unit and capture the item data. The data and the format of the data that is generated by this command are defined by the input parameters of the WFS_CMD_IPM_MEDIA_IN command. The media data is reported via the WFS_EXEE_IPM_MEDIADATA event.

This command must be supported by all Service Providers where the hardware does not have a stacker and where the Service Provider supports the application making the accept/refuse decision. On single item feed devices this command simply returns the error code WFS_ERR_IPM_NOMEDIAPRESENT. This allows a single application flow to be used on all devices without a stacker.

Input Param None.

Output Param LPWFSIPMNEXTITEMOUT lpFeederStatus;

```
typedef struct _wfs_ipm_next_item_out
     {
     WORD                 wMediaFeeder;
     } WFSIPMNEXTITEMOUT, *LPWFSIPMNEXTITEMOUT;
```

wMediaFeeder
Supplies the state of the media feeder. This value indicates if there are items on the media feeder waiting processing via the WFS_CMD_IPM_GET_NEXT_ITEM command. This value can be one of the following values:

Value	Meaning
WFS_IPM_FEEDERISEMPTY	The media feeder is empty.
WFS_IPM_FEEDERISNOTEMPTY	The media feeder is not empty.
WFS_IPM_FEEDERISNOTSUPPORTED	The physical device has no media feeder.

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

Value	Meaning
WFS_ERR_IPM_NOMEDIAPRESENT	No media is present on the media feeder.
WFS_ERR_IPM_MEDIAJAMMED	The media is jammed.
WFS_ERR_IPM_FILEIOERROR	Directory does not exist or File IO error while storing the image to the hard disk.
WFS_ERR_IPM_REFUSEDITEMS	Programming Error: refused items that must be returned via WFS_CMD_IPM_PRESENT_MEDIA have not been presented (see *bPresentRequired* in the WFS_EXEE_IPM_MEDIAREFUSED event parameters).
WFS_ERR_IPM_POSITIONNOTEMPTY	One of the input/output/refused positions is not empty.
WFS_ERR_IPM_SCANNERINOP	Only images were requested by the application and these cannot be obtained because the image scanner is inoperative.
WFS_ERR_IPM_MICRINOP	Only MICR data was requested by the application and it cannot be obtained because the MICR reader is inoperative.
WFS_ERR_IPM_SEQUENCEINVALID	Programming error. Invalid command sequence.
WFS_ERR_IPM_FEEDERINOPERATIVE	The media feeder is inoperative.

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

Value	Meaning
WFS_EXEE_IPM_MEDIAREFUSED	Media has been refused.
WFS_EXEE_IPM_MEDIADATA	Delivers media data (images and code line) during the command.

WFS_USRE_IPM_SCANNERTHRESHOLD	The imaging scanner is fading or inoperative. Note that this event is sent only once, at the point at which the status changes.
WFS_USRE_IPM_MICRTHRESHOLD	The MICR reader performance is degraded or the reader is inoperative. Note that this event is sent only once, at the point at which the status changes.

Comments None.

5.13 WFS_CMD_IPM_ACTION_ITEM

Description This command is used to cause the pre-defined actions (move item to destination, stamping, endorsing, re-imaging) to be executed on the current media item. This command only applies to devices without stackers and on devices with stackers this command is not supported.

Input Param None.

Output Param None.

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

Value	Meaning
WFS_ERR_IPM_SHUTTERFAIL	Open or close of the shutter failed due to manipulation or hardware error.
WFS_ERR_IPM_MEDIABINERROR	A problem occurred with a media bin. A WFS_EXEE_IPM_MEDIABINERROR event will be sent with the details.
WFS_ERR_IPM_MEDIAJAMMED	The media is jammed.
WFS_ERR_IPM_TONEROUT	Toner or ink supply is empty or printing contrast with ribbon is not sufficient.
WFS_ERR_IPM_INKOUT	No stamping possible, stamping ink supply empty.
WFS_ERR_IPM_NOMEDIAPRESENT	No media is present in the device.
WFS_ERR_IPM_SEQUENCEINVALID	Programming error. Invalid command sequence.
WFS_ERR_IPM_FILEIOERROR	Directory does not exist or access denied.
WFS_ERR_IPM_SCANNERINOP	The scanner is inoperative.
WFS_ERR_IPM_REFUSEDITEMS	Programming Error: refused items that must be returned via WFS_CMD_IPM_PRESENT_MEDIA have not been presented (see *bPresentRequired* in the WFS_EXEE_IPM_MEDIAREFUSED event parameters).
WFS_ERR_IPM_POSITIONNOTEMPTY	One of the input/output/refused positions is not empty.

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

Value	Meaning
WFS_SRVE_IPM_MEDIATAKEN	The media has been taken by the user.
WFS_EXEE_IPM_MEDIAPRESENTED	Media has been presented for removal.
WFS_EXEE_IPM_MEDIADATA	Delivers media images scanned after the item has been printed.
WFS_USRE_IPM_MEDIABINTHRESHOLD	A threshold condition has occurred in one of the media bins.
WFS_EXEE_IPM_MEDIABINERROR	A problem occurred with a media bin.
WFS_USRE_IPM_TONERTHRESHOLD	The toner or ink supply is low or empty or the printing contrast with ribbon is weak or not sufficient, operator intervention is required. Note that this event is sent only once, at the point at which the supply becomes low or empty. It is sent with WFS_IPM_TONERLOW or WFS_IPM_TONEROUT status.
WFS_USRE_IPM_INKTHRESHOLD	The stamp ink supply is low or empty, operator intervention is required. Note that this event is sent only once, at the point at which the supply becomes low or empty. It is sent with WFS_IPM_INKLOW or WFS_IPM_INKOUT status.

WFS_USRE_IPM_SCANNERTHRESHOLD	The imaging scanner is fading or inoperative. Note that this event is sent only once, at the point at which the status changes.

Comments None.

5.14 WFS_CMD_IPM_EXPEL_MEDIA

Description The media that has been presented to the customer will be expelled out of the device.

This command completes after the bunch has been expelled from the device.

This command does not end the current media-in transaction. The application must deal with any media remaining within the device, e.g. by using WFS_CMD_IPM_MEDIA_IN_ROLLBACK, WFS_CMD_IPM_MEDIA_IN_END, or WFS_CMD_IPM_RETRACT_MEDIA.

Input Param None.

Output Param None.

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

Value	Meaning
WFS_ERR_IPM_NOMEDIAPRESENT	No media present to expel.
WFS_ERR_IPM_MEDIAJAMMED	The media is jammed.
WFS_ERR_IPM_SHUTTERFAIL	Open or close of the shutter failed due to manipulation or hardware error.
WFS_ERR_IPM_SEQUENCEINVALID	Programming error. Invalid command sequence.

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

Comments None.

5.15 WFS_CMD_IPM_GET_IMAGE_AFTER_PRINT

Description This command is used to indicate that an image of the item should be generated after the text is printed on the item. The image is not generated during execution of this command.

On devices with stackers, the image will be scanned during execution of the WFS_CMD_IPM_MEDIA_IN_END command. On devices without stackers, the image will be scanned during execution of the WFS_CMD_IPM_ACTION_ITEM command.

Input Param LPWFSIPMGETIMAGEAFTERPRINT lpGetImageAfterPrint;

```
typedef struct _wfs_ipm_get_image_after_print
     {
     USHORT               usMediaID;
     LPWFSIPMIMAGEREQUEST *lppImage;
     } WFSIPMGETIMAGEAFTERPRINT, *LPWFSIPMGETIMAGEAFTERPRINT;
```

usMediaID
Specifies the sequence number of a media item. Valid IDs are 1 to the maximum media ID assigned within the transaction. Zero selects all media on the stacker.

lppImage
Pointer to a NULL-terminated array of pointers to WFSIPMIMAGEREQUEST structures describing the required images.

```
typedef struct _wfs_ipm_image_request
     {
     WORD                 wImageSource;
     WORD                 wImageType;
     WORD                 wImageColorFormat;
     WORD                 wImageScanColor;
     LPSTR                lpszImagePath;
     } WFSIPMIMAGEREQUEST, *LPWFSIPMIMAGEREQUEST;
```

wImageSource
Specifies the source as one of the following values:

Value	Meaning
WFS_IPM_IMAGEFRONT	The returned image is for the front of the media item.
WFS_IPM_IMAGEBACK	The returned image is for the back of the media item.

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

Value	Meaning
WFS_IPM_IMAGETIF	The returned image is in TIFF 6.0 format. The output file name will have the .tif extension appended to the filename.
WFS_IPM_IMAGEWMF	The returned image is in WMF (Windows Metafile) format. The output file name will have the .wmf extension appended to the filename.
WFS_IPM_IMAGEBMP	The returned image is in Windows BMP format. The output file name will have the .bmp extension appended to the filename.
WFS_IPM_IMAGEJPG	The returned image is in JPG format. The output file name will have the .jpg extension appended to the filename.

wImageColorFormat
Specifies the color format of the requested image as one of the following values:

Value	Meaning
WFS_IPM_IMAGECOLORBINARY	The scanned images has to be returned in binary (image contains two colors, usually the colors black and white).
WFS_IPM_IMAGECOLORGRAYSCALE	The scanned images has to be returned in gray scale (image contains multiple gray colors).
WFS_IPM_IMAGECOLORFULL	The scanned images has to be returned in full color (image contains colors like red, green, blue etc.).

wImageScanColor
Selects the image scan color. The value is specified as one of the following values:

Value	Meaning
WFS_IPM_SCANCOLORDEFAULT	Select the default scan color for the side of the item being scanned.
WFS_IPM_SCANCOLORRED	Select the red scan color.
WFS_IPM_SCANCOLORGREEN	Select the green scan color.
WFS_IPM_SCANCOLORBLUE	Select the blue scan color.
WFS_IPM_SCANCOLORYELLOW	Select the yellow scan color.
WFS_IPM_SCANCOLORWHITE	Select the white scan color.

lpszImagePath
Specifies the full path name of the folder where the image will be stored, e.g. “C:\TEMP”. The actual file name for the image produced will be vendor specific. The name used is reported in the event containing the image data. The Service Provider may re-use file names from the start of each media in transaction, so applications must manage the file lifetime as required. If NULL is provided for this parameter then the command will be rejected with the WFS_ERR_INVALID_DATA error. If the folder does not exist or cannot be accessed by the Service Provider then the command will be rejected with the WFS_ERR_IPM_FILEIOERROR error.

Output Param None.

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

Value	Meaning
WFS_ERR_IPM_MEDIAJAMMED	The media is jammed.
WFS_ERR_IPM_FILEIOERROR	Directory does not exist or access denied.
WFS_ERR_IPM_SCANNERINOP	Image scanner is inoperative so no image can be produced.
WFS_ERR_IPM_NOMEDIAPRESENT	No media is present in the device.
WFS_ERR_IPM_SEQUENCEINVALID	Programming error. Invalid command sequence.
WFS_ERR_IPM_INVALIDMEDIAID	The requested Media ID does not exist.

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

Comments None.

5.16 WFS_CMD_IPM_ACCEPT_ITEM

Description This command is used by applications to indicate if the current media item should be accepted or refused. Applications only use this command when the WFS_CMD_IPM_MEDIA_IN command is used in the mode where the application can decide if each physically acceptable media item should be accepted or refused, i.e. the *bApplicationRefuse* parameter is TRUE.

Input Param LPWFSIPMACCEPTITEM lpAcceptItem;

```
typedef struct _wfs_ipm_accept_item
     {
     BOOL                bAccept;
     } WFSIPMACCEPTITEM, *LPWFSIPMACCEPTITEM;
```

bAccept
Specifies if the item should be accepted or refused. If this value is TRUE then the item is accepted and moved to the stacker. If this value is FALSE then the item is moved to the re-buncher/refuse position.

Output Param None.

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

Value	Meaning
WFS_ERR_IPM_MEDIAJAMMED	The media is jammed.
WFS_ERR_IPM_NOMEDIAPRESENT	No media is present in the device.
WFS_ERR_IPM_SEQUENCEINVALID	Programming error. Invalid command sequence.
WFS_ERR_IPM_REFUSEDITEMS	Programming Error: refused items that must be returned via WFS_CMD_IPM_PRESENT_MEDIA have not been presented (see *bPresentRequired* in the WFS_EXEE_IPM_MEDIAREFUSED event parameters).
WFS_ERR_IPM_POSITIONNOTEMPTY	One of the input/output/refused positions is not empty.

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

Comments None.

5.17 WFS_CMD_IPM_SUPPLY_REPLENISH

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

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

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

Input Param LPWFSIPMSUPPLYREPLEN lpSupplyReplen;

```
typedef struct _wfs_ipm_supply_replen
     {
     WORD                 fwSupplyReplen;
     } WFSIPMSUPPLYREPLEN, *LPWFSIPMSUPPLYREPLEN;
```

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

Value	Meaning
WFS_IPM_REPLEN_TONER	The toner supply was replenished.
WFS_IPM_REPLEN_INK	The ink supply was replenished.

Output Param None.

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

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

Value	Meaning
WFS_USRE_IPM_TONERTHRESHOLD	This user event is used to specify that the state of the toner (or ink) supply threshold has been cleared.
WFS_USRE_IPM_INKTHRESHOLD	This user event is used to specify that the state of the stamping ink supply threshold has been cleared.

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

5.18 WFS_CMD_IPM_POWER_SAVE_CONTROL

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

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

Input Param LPWFSIPMPOWERSAVECONTROL lpPowerSaveControl;

```
typedef struct _wfs_ipm_power_save_control
     {
     USHORT              usMaxPowerSaveRecoveryTime;
     } WFSIPMPOWERSAVECONTROL, *LPWFSIPMPOWERSAVECONTROL;
```

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

Output Param None.

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

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

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

Value	Meaning
WFS_SRVE_IPM_POWER_SAVE_CHANGE	The power save recovery time has changed.

Comments None.

6. Events

6.1 WFS_EXEE_IPM_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	None.
Comments	None.

6.2 WFS_EXEE_IPM_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 media.

6.3 WFS_USRE_IPM_MEDIABINTHRESHOLD

Description This user event specifies that a threshold condition has occurred in one of the media bins or the threshold condition is removed.

Event Param LPWFSIPMMEDIABIN lpMediaBin;

lpMediaBin
Pointer to WFSIPMMEDIABIN structure, describing the media bin on which the threshold condition occurred. See *lpMediaBin->usStatus* for the type of condition. For a description of the WFSIPMMEDIABIN structure, see the definition of the WFS_INF_IPM_MEDIA_BIN_INFO command.

Comments None.

6.4 WFS_SRVE_IPM_MEDIABININFOCHANGED

Description This service event specifies that a media bin has changed in configuration. A media bin may have been removed or inserted or a media bin parameter may have changed. This event will also be posted on successful completion of the following commands:

- WFS_CMD_IPM_SET_MEDIA_BIN_INFO

If an application receives this event it should issue a WFS_INF_IPM_MEDIA_BIN_INFO command to obtain updated media bin information.

Event Param LPWFSIPMMEDIABIN lpMediaBin;

lpMediaBin
Pointer to the changed media bin structure. For a description of the WFSIPMMEDIABIN structure see the definition of the WFS_INF_IPM_MEDIA_BIN_INFO command.

Comments None.

6.5 WFS_EXEE_IPM_MEDIABINERROR

Description This execute event specifies that a media bin was addressed which caused a problem.

Event Param LPWFSIPMMBERROR lpMediaBinError;

```
typedef struct _wfs_ipm_mb_error
     {
     WORD                 wFailure;
     LPWFSIPMMEDIABIN     lpMediaBin;
     } WFSIPMMBERROR, *LPWFSIPMMBERROR;
```

wFailure
Specifies the kind of failure that occurred in the media bin. This value is specified as one of the following values:

Value	Meaning
WFS_IPM_MEDIABINJAMMED	Specified media bin is jammed.
WFS_IPM_MEDIABINERROR	Specified media bin has malfunctioned.
WFS_IPM_MEDIABINFULL	Specified media bin is full.
WFS_IPM_MEDIABINNOTCONF	Specified media bin is not configured due to being removed and/or replaced with a different media bin.
WFS_IPM_MEDIABININVALID	Specified media bin ID is invalid.
WFS_IPM_MEDIABINCONFIG	Attempt to change the setting of a self-configuring media bin.
WFS_IPM_MEDIABINFEEDPROBLEM	A problem has been detected with the feeding module.

lpMediaBin
Pointer to a WFSIPMMEDIABIN structure containing the details of the media bin structure that caused the problem. For a description of the WFSIPMMEDIABIN structure see the definition of the WFS_INF_IPM_MEDIA_BIN_INFO command.

Comments None.

6.6 WFS_SRVE_IPM_MEDIATAKEN

Description This event is sent when the media is taken by the customer.

Event Param LPWFSIPMPOSITION lpPosition;

```
typedef struct _wfs_ipm_position
     {
     WORD                wPosition;
     } WFSIPMPOSITION, *LPWFSIPMPOSITION;
```

wPosition
Specifies the position where the media has been taken from. This value can be one of the following values:

Value	Meaning
WFS_IPM_POSINPUT	Input position.
WFS_IPM_POSOUTPUT	Output position.
WFS_IPM_POSREFUSED	Refused media item position.

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

6.7 WFS_USRE_IPM_TONERTHRESHOLD

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

Event Param LPWFSIPMTHRESHOLD lpwTonerThreshold;

```
typedef struct _wfs_ipm_threshold
     {
     WORD                 wThreshold;
     } WFSIPMTHRESHOLD, *LPWFSIPMTHRESHOLD;
```

wThreshold
Specified as one of the following values:

Value	Meaning
WFS_IPM_TONERFULL	The toner (or ink) in the device is in a good state.
WFS_IPM_TONERLOW	The toner (or ink) in the device is low.
WFS_IPM_TONEROUT	The toner (or ink) in the device is out.

Comments None.

6.8 WFS_USRE_IPM_SCANNERTHRESHOLD

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

Event Param LPWFSIPMSCANNERTHRESHOLD lpwScannerThreshold;

```
typedef struct _wfs_ipm_scanner_threshold
     {
     WORD                wScanner;
     WORD                wThreshold;
     } WFSIPMSCANNERTHRESHOLD, *LPWFSIPMSCANNERTHRESHOLD;
```

wScanner
Identifies the scanner where the threshold has been reached, specified as one of the following values:

Value	Meaning
WFS_IPM_FRONTSCANNER	Front image scanner.
WFS_IPM_BACKSCANNER	Back image scanner.

wThreshold
Specified as one of the following values:

Value	Meaning
WFS_IPM_SCANNEROK	The imaging scanner is in a good state.
WFS_IPM_SCANNERFADING	The imaging scanner performance is degraded.
WFS_IPM_SCANNERINOP	The imaging scanner is inoperative.

Comments None.

6.9 WFS_USRE_IPM_INKTHRESHOLD

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

Event Param LPWFSIPMTHRESHOLD lpwInkThreshold;

```
typedef struct _wfs_ipm_threshold
     {
     WORD                 wThreshold;
     } WFSIPMTHRESHOLD, *LPWFSIPMTHRESHOLD;
```

wThreshold
Specified as one of the following values:

Value	Meaning
WFS_IPM_INKFULL	The stamping ink in the device is in a good state.
WFS_IPM_INKLOW	The stamping ink in the device is low.
WFS_IPM_INKOUT	The stamping ink in the device is out.

Comments None.

6.10 WFS_SRVE_IPM_MEDIADETECTED

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

Event Param LPWFSIPMMEDIADETECTED lpMediaDetected;

```
typedef struct _wfs_ipm_media_detected
     {
     WORD                 wPosition;
     USHORT               usRetractBinNumber;
     } WFSIPMMEDIADETECTED, *LPWFSIPMMEDIADETECTED;
```

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

Value	Meaning
WFS_IPM_MEDIARETRACTED	The media was retracted during the reset operation.
WFS_IPM_MEDIAPRESENT	The media is in the device.
WFS_IPM_MEDIAPOSITION	The media is at one or more of the input, output and refused positions.
WFS_IPM_MEDIAJAMMED	The media is jammed in the device.
WFS_IPM_MEDIARETURNED	The media has been returned and taken by the customer.
WFS_IPM_MEDIAUNKNOWN	The media is in an unknown position.

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

Comments None.

6.11 WFS_EXEE_IPM_MEDIAPRESENTED

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

Event Param LPWFSIPMMEDIAPRESENTED lpMediaPresented;

```
typedef struct _wfs_ipm_media_presented
     {
     WORD                 wPosition;
     USHORT               usBunchIndex;
     USHORT               usTotalBunches;
     } WFSIPMMEDIAPRESENTED, *LPWFSIPMMEDIAPRESENTED;
```

wPosition
Specifies the position where the media has been presented to. This value can be one of the following values:

Value	Meaning
WFS_IPM_POSINPUT	Input position.
WFS_IPM_POSOUTPUT	Output position.
WFS_IPM_POSREFUSED	Refused media item position.

usBunchIndex
Specifies the index (starting from one) of the presented bunch (one or more items presented as a bunch).

usTotalBunches
Specifies the total number of bunches to be returned from all positions. The total represents the number of bunches that will be returned as a result of a single command that presents media. This value is zero if the total number of bunches is not known.

Comments None.

6.12 WFS_EXEE_IPM_MEDIAREFUSED

Description This event is sent when a media item is refused. One event is sent for every media item or bunch of media items that has been refused.

Event Param LPWFSIPMMEDIAREFUSED lpMediaRefused;

```
typedef struct _wfs_ipm_media_refused
     {
     WORD                 wReason;
     WORD                 wMediaLocation;
     BOOL                 bPresentRequired;
     LPWFSIPMMEDIASIZE    lpMediaSize;
     } WFSIPMMEDIAREFUSED, *LPWFSIPMMEDIAREFUSED;
```

wReason
Specified as one of the following values:

Value	Meaning
WFS_IPM_REFUSED_FOREIGNITEMS	Foreign items were detected in the input position.
WFS_IPM_REFUSED_STACKERFULL	The stacker is full or the maximum number of items that the application wants to be allowed on the stacker has been reached (see *usMaxMediaOnStacker* input parameter in WFS_CMD_IPM_MEDIA_IN*)*.
WFS_IPM_REFUSED_CODELINEINVALID	The code line data was found but was invalid.
WFS_IPM_REFUSED_INVALIDMEDIA	The media item was invalid, e.g. some devices may reject blank paper, or money, etc.
WFS_IPM_REFUSED_TOOLONG	The media item (or bunch of items) was too long.
WFS_IPM_REFUSED_TOOSHORT	The media item (or bunch of items) was too short.
WFS_IPM_REFUSED_TOOWIDE	The media item (or bunch of items) was too wide.
WFS_IPM_REFUSED_TOONARROW	The media item (or bunch of items) was too narrow.
WFS_IPM_REFUSED_TOOTHICK	The media item was too thick.
WFS_IPM_REFUSED_INVALIDORIENTATION	The media item was inserted in an invalid orientation.
WFS_IPM_REFUSED_DOUBLEDETECT	The media items could not be separated.
WFS_IPM_REFUSED_REFUSEPOSFULL	There are too many items in the refuse area. The refused items must be returned to the customer before any additional media items can be accepted.
WFS_IPM_REFUSED_RETURNBLOCKED	Processing of the items did not take place as the bunch of items is blocking the return of other items.
WFS_IPM_REFUSED_INVALIDBUNCH	Processing of the items did not take place as the bunch of items presented is invalid, e.g. it is too large or was presented incorrectly.
WFS_IPM_REFUSED_OTHERITEM	The item was refused for some reason not covered by the other reasons.
WFS_IPM_REFUSED_OTHERBUNCH	The bunch was refused for some reason not covered by the other reasons.
WFS_IPM_REFUSED_JAMMING	The media item is causing a jam.
WFS_IPM_REFUSED_METAL	Metal (e.g. staple, paperclip, etc) was detected in the input position.

wMediaLocation
Specifies where the refused media should be presented to the customer from. It can be one of the following values:

Value	Meaning
WFS_IPM_REFUSE_INPUT	The media is presented to the customer by passing WFS_IPM_REFUSE_INPUT to the WFS_CMD_IPM_PRESENT_MEDIA command. The media needs to be presented before any further input can take place.
WFS_IPM_REFUSE_REFUSED	The media is presented to the customer by passing WFS_IPM_REFUSE_REFUSED to the WFS_CMD_IPM_PRESENT_MEDIA command.
WFS_IPM_REFUSE_REBUNCHER	The media is presented to the customer by passing WFS_IPM_REFUSE_REBUNCHER to the WFS_CMD_IPM_PRESENT_MEDIA command.

bPresentRequired
This flag indicates if the media needs to be presented to the customer before any additional media movement commands can be executed. If this value is TRUE then the media must be presented to the customer via the WFS_CMD_IPM_PRESENT_MEDIA command before further media movement commands can be executed. If this value is FALSE then the device can continue without the media being returned to the customer.

lpMediaSize
Pointer to a WFSIPMMEDIASIZE structure that specifies the size of the refused media (or bunch of media). *lpMediaSize* is NULL if the device does not support media size measurement.

```
typedef struct _wfs_ipm_media_size
     {
     ULONG               ulSizeX;
     ULONG               ulSizeY;
     } WFSIPMMEDIASIZE, *LPWFSIPMMEDIASIZE;
```

ulSizeX
Specifies the width of the media in millimeters, or zero if unknown.

ulSizeY
Specifies the height of the media in millimeters, or zero if unknown.

Comments None.

6.13 WFS_EXEE_IPM_MEDIADATA

Description This event returns the code line and all the images requested for each item processed. This event can be generated during the WFS_CMD_IPM_MEDIA_IN, WFS_CMD_IPM_MEDIA_IN_END, WFS_CMD_IPM_GET_NEXT_ITEM and WFS_CMD_IPM_ACTION_ITEM commands. One event is generated for each item processed, no event is generated for refused items.

Event Param LPWFSIPMMEDIADATA lpMediaData;

```
typedef struct _wfs_ipm_mediadata
     {
     USHORT               usMediaID;
     ULONG                ulCodelineDataLength;
     LPBYTE               lpbCodelineData;
     WORD                 wMagneticReadIndicator;
     LPWFSIPMIMAGEDATA    *lppImage;
     WORD                 fwInsertOrientation;
     LPWFSIPMMEDIASIZE    lpMediaSize;
     WORD                 wMediaValidity;
     } WFSIPMMEDIADATA, *LPWFSIPMMEDIADATA;
```

usMediaID
Specifies the sequence number (starting from 1) of the media item.

ulCodelineDataLength
Number of bytes of the following *lpbCodelineData*.

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

wMagneticReadIndicator
Specifies the type of technology used to read a MICR code line.

Value	Meaning
WFS_IPM_MRI_MICR	The MICR code line was read using MICR technology and MICR characters were present.
WFS_IPM_MRI_NOT_MICR	The MICR code line was NOT read using MICR technology.
WFS_IPM_MRI_NO_MICR	The MICR code line was read using MICR technology and no magnetic characters were read.
WFS_IPM_MRI_UNKNOWN	It is unknown how the MICR code line was read.
WFS_IPM_MRI_NOTMICRFORMAT	The code line is not a MICR format code line.
WFS_IPM_MRI_NOT_READ	No code line was read.

lppImage
Pointer to a NULL-terminated array of pointers to WFSIPMIMAGEDATAITEM structures. If image data items are not used *lppImage* will be set to NULL. If the Service Provider has determined the orientation of the media (i.e. *fwInsertOrientation* is not set to WFS_IPM_INSUNKNOWN), then all images returned are in the standard orientation and the images will match the image source requested by the application. This means that images will be returned with the code line at the bottom, and the image of the front and rear of the media item will be returned in the structures associated with the WFS_IPM_IMAGEFRONT and WFS_IPM_IMAGEBACK image sources respectively.

```
typedef struct _wfs_ipm_image_data
     {
     WORD                wImageSource;
     WORD                wImageType;
     WORD                wImageColorFormat;
     WORD                wImageScanColor;
     WORD                wImageStatus;
     LPSTR               lpszImageFile;
     } WFSIPMIMAGEDATA, *LPWFSIPMIMAGEDATA;
```

wImageSource
Specifies the source of the data returned by this item as one of the following values:

Value	Meaning
WFS_IPM_IMAGEFRONT	The returned image is for the front of the media item.
WFS_IPM_IMAGEBACK	The returned image is for the back of the media item.

wImageType
Specifies the format of the image returned by this item as one of the following values:

Value	Meaning
WFS_IPM_IMAGETIF	The returned image is in TIFF 6.0 format.
WFS_IPM_IMAGEWMF	The returned image is in WMF (Windows Metafile) format.
WFS_IPM_IMAGEBMP	The returned image is in Windows BMP format.
WFS_IPM_IMAGEJPG	The returned image is in JPG format.

wImageColorFormat
Specifies the color format of the image returned by this item as one of following flags:

Value	Meaning
WFS_IPM_IMAGECOLORBINARY	The scanned image is returned in binary format (image contains two colors, usually the colors black and white).
WFS_IPM_IMAGECOLORGRAYSCALE	The scanned image is returned in binary format (image contains multiple gray colors).
WFS_IPM_IMAGECOLORFULL	The scanned image is returned in full color (image contains colors like red, green, blue, etc.).

wImageScanColor
Specifies the scan color of the image returned by this item as one of following flags:

Value	Meaning
WFS_IPM_SCANCOLORRED	The image was scanned with red light.
WFS_IPM_SCANCOLORGREEN	The image was scanned with green light.
WFS_IPM_SCANCOLORBLUE	The image was scanned with blue light.
WFS_IPM_SCANCOLORYELLOW	The image was scanned with yellow light.
WFS_IPM_SCANCOLORWHITE	The image was scanned with white light.

wImageStatus
Status of the requested image data. Possible values are:

Value	Meaning
WFS_IPM_DATAOK	The data is OK.
WFS_IPM_DATASRCNOTSUPP	The data source or image attributes are not supported by the Service Provider, e.g. scan color not supported.
WFS_IPM_DATASRCMISSING	The requested image could not be obtained.

lpszImageFile
Specifies the full path and file name where the image is stored, e.g. "C:\Temp\FrontImage.bmp". Each image requested is stored in a unique file with a unique name allocated by the Service Provider. The folder location where the file is stored is specified in the input parameters of the WFS_CMD_IPM_MEDIA_IN command. File names which are allocated by the Service Provider will be reused in the next transaction.

fwInsertOrientation
This value reports how the media item was actually inserted into the input position (from the customers perspective). This value is either WFS_IPM_INSUNKNOWN or a combination of one value from type A and one value from type B.

Value	Meaning	Type
WFS_IPM_INSUNKNOWN	The orientation of the inserted media is unknown.	N/A
WFS_IPM_INSCODELINERIGHT	The code line is to the right.	A
WFS_IPM_INSCODELINELEFT	The code line is to the left.	A
WFS_IPM_INSCODELINEBOTTOM	The code line is to the bottom.	A
WFS_IPM_INSCODELINETOP	The code line is to the top.	A
WFS_IPM_INSFACEUP	The front of the media (the side with the code line) is facing up.	B
WFS_IPM_INSFACEDOWN	The front of the media (the side with the code line) is facing down.	B

lpMediaSize
Pointer to a WFSIPMMEDIASIZE structure that specifies the size of the media item. *lpMediaSize* is NULL if the device does not support media size measurement.

```
typedef struct _wfs_ipm_media_size
     {
     ULONG               ulSizeX;
     ULONG               ulSizeY;
     } WFSIPMMEDIASIZE, *LPWFSIPMMEDIASIZE;
```

ulSizeX
Specifies the width of the media in millimeters, or zero if unknown.

ulSizeY
Specifies the height of the media in millimeters, or zero if unknown.

wMediaValidity
Media items may have special security features which can be detected by the device. This field specifies whether the media item is suspect or valid, allowing the application a choice in how to further process a media item that could not be confirmed as being valid. This value is specified as one of the following values:

Value	Meaning
WFS_IPM_ITEMOK	The media item is valid.
WFS_IPM_ITEMSUSPECT	The validity of the media item is suspect.
WFS_IPM_ITEMUNKNOWN	The validity of the media item is unknown.
WFS_IPM_ITEMNOVALIDATION	No specific security features were evaluated.

Comments None.

6.14 WFS_USRE_IPM_MICRTHRESHOLD

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

Event Param LPWFSIPMTHRESHOLD lpwMICRThreshold;

```
typedef struct _wfs_ipm_threshold
     {
     WORD                 wThreshold;
     } WFSIPMTHRESHOLD, *LPWFSIPMTHRESHOLD;
```

wThreshold
Specified as one of the following values:

Value	Meaning
WFS_IPM_MICROK	The MICR reader is in a good state.
WFS_IPM_MICRFADING	The MICR reader performance is degraded.
WFS_IPM_MICRINOP	The MICR reader is inoperative.

Comments None.

6.15 WFS_EXEE_IPM_MEDIAREJECTED

Description This event is generated to report that an attempt to insert media into the device has been rejected before the media was fully inside the device, i.e. no WFS_EXEE_IPM_MEDIAINSERTED event has been generated. Rejection of the media will cause the WFS_CMD_IPM_MEDIA_IN command to complete with a WFS_ERR_IPM_MEDIAREJECTED error, at which point the media should be removed.

Event Param LPWFSIPMMEDIAREJECTED lpMediaRejected;

```
typedef struct _wfs_ipm_media_rejected
     {
     WORD                 wReason;
     } WFSIPMMEDIAREJECTED, *LPWFSIPMMEDIAREJECTED;
```

wReason
Specified as one of the following values:

Value	Meaning
WFS_IPM_REJECT_LONG	The rejected media was too long.
WFS_IPM_REJECT_THICK	The rejected media was too thick.
WFS_IPM_REJECT_DOUBLE	More than one media item was detected (this value only applies to devices without a media feeder).
WFS_IPM_REJECT_TRANSPORT	The media could not be moved inside the device.
WFS_IPM_REJECT_SHUTTER	The media was rejected due to the shutter failing to close.
WFS_IPM_REJECT_REMOVED	The media was removed (no taken event is expected).
WFS_IPM_REJECT_METAL	Metal (e.g. staple, paperclip, etc) was detected in the input position.
WFS_IPM_REJECT_FOREIGNITEMS	The media was rejected because foreign items were detected in the input position.
WFS_IPM_REJECT_OTHER	The media was rejected due to a reason other than those listed above.

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

6.16 WFS_SRVE_IPM_DEVICEPOSITION

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

Event Param LPWFSIPMDEVICEPOSITION lpDevicePosition;

```
typedef struct _wfs_ipm_device_position
     {
     WORD                 wPosition;
     } WFSIPMDEVICEPOSITION, *LPWFSIPMDEVICEPOSITION;
```

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

Value	Meaning
WFS_IPM_DEVICEINPOSITION	The device is in its normal operating position.
WFS_IPM_DEVICENOTINPOSITION	The device has been removed from its normal operating position.
WFS_IPM_DEVICEPOSUNKNOWN	The position of the device cannot be determined.

Comments None.

6.17 WFS_SRVE_IPM_POWER_SAVE_CHANGE

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

Event Param LPWFSIPMPOWERSAVECHANGE lpPowerSaveChange;

```
typedef struct _wfs_ipm_power_save_change
     {
     USHORT               usPowerSaveRecoveryTime;
     } WFSIPMPOWERSAVECHANGE, *LPWFSIPMPOWERSAVECHANGE;
```

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

Comments None.

7. Command and Event Flows

7.1 Devices with Stacker

7.1.1 Bunch Media Processing (OK flow)

	Application / Customer	**XFS IPM Service**
1	WFS_CMD_IPM_MEDIA_IN	- Event: WFS_EXEE_IPM_NOMEDIA - Wait for media insertion
2	Customer deposits a bunch of media items	- Event: WFS_EXEE_IPM_MEDIAINSERTED - Test and separate media items. - Send one WFS_EXEE_IPM_MEDIADATA event for every media item. - Completion: WFS_CMD_IPM_MEDIA_IN
3	WFS_INF_IPM_TRANSACTION_STATUS	- Report media status and positions
4	If more media is to be inserted: Goto 1 Otherwise loop over all accepted media items: Step 5-8	
5	If additional images are required then WFS_CMD_IPM_READ_IMAGE	- Reads data from the selected media item - Writes image data to the specified files. - Completion: WFS_CMD_IPM_READ_IMAGE
6	WFS_CMD_IPM_PRINT_TEXT	- Specifies if the item is to be stamped and specifies the data to print on the selected media item. - Completion: WFS_CMD_IPM_PRINT_TEXT
7	WFS_CMD_IPM_SET_DESTINATION	- Specifies the destination of the selected media item. - Completion: WFS_CMD_IPM_SET_DESTINATION
8	Continue with individual media item processing: Goto 5	
9	WFS_CMD_IPM_MEDIA_IN_END	- End processing for the inserted media items - Print on the individual media items - Transport the individual media items to the specified destinations
10		- Completion: WFS_CMD_IPM_MEDIA_IN_END

7.1.2 Bunch Media Processing (Some Media Items Returned)

	Application / Customer	**XFS IPM Service**
1	WFS_CMD_IPM_MEDIA_IN	- Event: WFS_EXEE_IPM_NOMEDIA - Wait for media insertion.
2	Customer deposits a bunch of media items	- Event: WFS_EXEE_IPM_MEDIAINSERTED - Test and separate media items - Send one WFS_EXEE_IPM_MEDIADATA event for every media item. - Completion: WFS_CMD_IPM_MEDIA_IN
3	WFS_INF_IPM_TRANSACTION_STATUS	- Report media status and positions.
4	If more media is to be inserted: Goto 1 Otherwise loop over all accepted media items: Step 5-8	
5	If additional images are required then WFS_CMD_IPM_READ_IMAGE	- Reads data from the selected media item - Writes image data to the specified files. - Completion: WFS_CMD_IPM_READ_IMAGE
6	WFS_CMD_IPM_PRINT_TEXT	- Specifies if the item is to be stamped and specifies the data to print on the selected media item. - Completion: WFS_CMD_IPM_PRINT_TEXT
7	WFS_CMD_IPM_SET_DESTINATION	- Specifies the destination of the selected media item (bin or output) - For some media items the output position is selected. - Completion: WFS_CMD_IPM_SET_DESTINATION
8	Continue with individual media item processing: Goto 5	
9	WFS_CMD_IPM_MEDIA_IN_END	- End processing for the inserted media items - Print on the individual media items - Transport the individual media items to the specified destinations
10		If *bPresentControl*=TRUE - Present the returned media items to the customer - Event: WFS_EXEE_IPM_MEDIAPRESENTED
11		- Completion: WFS_CMD_IPM_MEDIA_IN_END
12	If *bPresentControl*=FALSE WFS_CMD_IPM_PRESENT_MEDIA	- Present the returned media items to the customer - Event: WFS_EXEE_IPM_MEDIAPRESENTED - Completion: WFS_CMD_IPM_PRESENT_MEDIA
13	Customer takes returned media items	- Event: WFS_SRVE_IPM_MEDIATAKEN

7.1.3 Bunch Media Processing with Errors

	Application / Customer	XFS IPM Service
1	WFS_CMD_IPM_MEDIA_IN	- Event: WFS_EXEE_IPM_NOMEDIA - Wait for media insertion
2	Customer deposits a bunch of media items	- Event: WFS_EXEE_IPM_MEDIAINSERTED - Test and separate media items - Send one WFS_EXEE_IPM_MEDIADATA event for every accepted media item - Event: WFS_EXEE_IPM_MEDIAREFUSED (Reason= WFS_IPM_REFUSED_FOREIGNITEMS) if foreign items are detected. - and/or - Event: WFS_EXEE_IPM_MEDIAREFUSED (Reason= WFS_IPM_REFUSED_STACKERFULL) if the stacker becomes full - and/or - Event: WFS_EXEE_IPM_MEDIAREFUSED (Reason = WFS_IPM_REFUSED_CODELINEINVALID) if the code line could not be read
3		- Completion: WFS_CMD_IPM_MEDIA_IN
4	If the application chooses to return refused items before the end of transaction WFS_CMD_IPM_PRESENT_MEDIA. Otherwise, continue with Step 4 of the OK flow	
5	For all bunches except for the last bunch returned to the customer repeat steps 6-7. For the last bunch go to step 8	
6		- Present the media items to the customer - Event: WFS_EXEE_IPM_MEDIAPRESENTED
7	Customer takes returned media items	- Event: WFS_SRVE_IPM_MEDIATAKEN
8	Present last bunch to customer	- Present the media items to the customer - Event: WFS_EXEE_IPM_MEDIAPRESENTED
9		- Completion: WFS_CMD_IPM_PRESENT_MEDIA
10	Customer takes returned media items	- Event: WFS_SRVE_IPM_MEDIATAKEN
11	Continue with Step 4 of the OK flow	

7.1.4 Bunch media processing with Rollback

	Application / Customer	XFS IPM Service
	Step 1 to 8 see OK flow	
9	WFS_CMD_IPM_ MEDIA_IN_ROLLBACK	- Without printing, all media items from the stacker (plus any refused notes not already returned) are transported to the output position.
10		If bPresentControl=TRUE - Present the media items to the customer - Event: WFS_EXEE_IPM_MEDIAPRESENTED
11		- Completion: WFS_CMD_IPM_ROLLBACK
12	If *bPresentControl*=FALSE WFS_CMD_IPM_PRESENT_MEDIA	- Present the returned media items to the customer - Event: WFS_EXEE_IPM_MEDIAPRESENTED - Completion: WFS_CMD_IPM_PRESENT_MEDIA
13	Customer takes returned media items	- Event: WFS_SRVE_IPM_MEDIATAKEN

7.1.5 Bunch media processing with Retract

	Application / Customer	**XFS IPM Service**
	Step 1 to 8 see OK flow	
9	WFS_CMD_IPM_RETRACT_MEDIA	- Stops processing of media items - Without printing, all media items from the stacker are transported to the retract cassette. - Completion: WFS_CMD_IPM_RETRACT_MEDIA

7.1.6 Bunch Media Processing - Application Refuse Decision (All OK flow)

	Application / Customer	**XFS IPM Service**
1	WFS_CMD_IPM_MEDIA_IN (*bApplicationRefuse* = TRUE)	- Event: WFS_EXEE_IPM_NOMEDIA - Wait for media insertion
2	Customer deposits a bunch of media items	- Event: WFS_EXEE_IPM_MEDIAINSERTED - Test and separate media item. - Send one WFS_EXEE_IPM_MEDIADATA event for first media item. - Completion: WFS_CMD_IPM_MEDIA_IN
3	Application processes media data and decides to keep media item WFS_CMD_IPM_ACCEPT_ITEM (TRUE) - keep item	- Move item to stacker. - Completion: WFS_CMD_IPM_ACCEPT_ITEM
4	WFS_CMD_IPM_GET_NEXT_ITEM	- If item successfully read then send one WFS_EXEE_IPM_MEDIADATA event for next media item. - Completion: WFS_CMD_IPM_GET_NEXT_ITEM
5	If the item was read successfully continue with step 3. Otherwise if there are no more items then continue with step 6.	
6	If more media is to be inserted: Goto 1 Otherwise loop over all accepted media items: Step 7-9	
7	WFS_CMD_IPM_PRINT_TEXT	- Specifies if the item is to be stamped and specifies the data to print on the selected media item. - Completion: WFS_CMD_IPM_PRINT_TEXT
8	WFS_CMD_IPM_SET_DESTINATION	- Specifies the destination of the selected media item. - Completion: WFS_CMD_IPM_SET_DESTINATION
9	Continue with individual media item processing: Goto 5	
10	WFS_CMD_IPM_MEDIA_IN_END	- End processing for the inserted media items - Print on the individual media items - Transport the individual media items to the specified destinations
11		- Completion: WFS_CMD_IPM_MEDIA_IN_END

7.1.7 Bunch Media Processing - Application Refuse Decision (Some items refused)

	Application / Customer	XFS IPM Service
1	WFS_CMD_IPM_MEDIA_IN (*bApplicationRefuse* = TRUE)	- Event: WFS_EXEE_IPM_NOMEDIA - Wait for media insertion
2	Customer deposits a bunch of media items	- Event: WFS_EXEE_IPM_MEDIAINSERTED - Test and separate media item. - Send one WFS_EXEE_IPM_MEDIADATA event for first media item. - Completion: WFS_CMD_IPM_MEDIA_IN
3	Application processes media data and decides to keep/or refuse media item WFS_CMD_IPM_ACCEPT_ITEM (TRUE/FALSE)	- Move item to stacker or refuse position/re-buncher. - Completion: WFS_CMD_IPM_ACCEPT_ITEM
4	WFS_CMD_IPM_GET_NEXT_ITEM	- If item successfully read then send one WFS_EXEE_IPM_MEDIADATA event for next media item. - Completion: WFS_CMD_IPM_GET_NEXT_ITEM
5	If the item was read successfully continue with step 3. Otherwise if there are no more items then continue with step 6.	
6	If the application chooses to return refused items before the end of transaction WFS_CMD_IPM_PRESENT_MEDIA. Otherwise, continue with Step 13	
7	For all bunches except for the last bunch returned to the customer repeat steps 8-9. For the last bunch go to step 10	
8		- Present the media items to the customer - Event: WFS_EXEE_IPM_MEDIAPRESENTED
9	Customer takes returned media items	- Event: WFS_SRVE_IPM_MEDIATAKEN
10	Present last bunch to customer	- Present the media items to the customer - Event: WFS_EXEE_IPM_MEDIAPRESENTED
11		- Completion: WFS_CMD_IPM_PRESENT_MEDIA
12	Customer takes returned media items	- Event: WFS_SRVE_IPM_MEDIATAKEN
13	If more media is to be inserted: Goto 1 Otherwise loop over all accepted media items: Step 14-16	
14	WFS_CMD_IPM_PRINT_TEXT	- Specifies if the item is to be stamped and specifies the data to print on the selected media item. - Completion: WFS_CMD_IPM_PRINT_TEXT
15	WFS_CMD_IPM_SET_DESTINATION	- Specifies the destination of the selected media item. - Completion: WFS_CMD_IPM_SET_DESTINATION
16	Continue with individual media item processing: Goto 5	
17	WFS_CMD_IPM_MEDIA_IN_END	- End processing for the inserted media items - Print on the individual media items - Transport the individual media items to the specified destinations
18		- Completion: WFS_CMD_IPM_MEDIA_IN_END

7.2 Devices without Stacker

Note that in the following flows that the single and bunch media devices follow the same flow except only one item is inserted and WFS_CMD_GET_NEXT_ITEM always returns reporting that there are no more items to process.

7.2.1 Bunch Media Processing (OK flow)

	Application / Customer	**XFS IPM Service**
1	WFS_CMD_IPM_MEDIA_IN	- Event: WFS_EXEE_IPM_NOMEDIA - Wait for media insertion
2	Customer deposits a bunch of media items	- Event: WFS_EXEE_IPM_MEDIAINSERTED - Send one WFS_EXEE_IPM_MEDIADATA event for first media item. - Completion: WFS_CMD_IPM_MEDIA_IN
3	WFS_INF_IPM_TRANSACTION_STATUS	- Report media status and positions
4	If additional images are required then WFS_CMD_IPM_READ_IMAGE	- Reads data from the selected media item - Writes image data to the specified files. - Completion: WFS_CMD_IPM_READ_IMAGE
5	WFS_CMD_IPM_PRINT_TEXT	- Specifies if the item is to be stamped and specifies the data to print on the selected media item. - Completion: WFS_CMD_IPM_PRINT_TEXT
6	WFS_CMD_IPM_SET_DESTINATION	- Specifies the destination of the selected media item. - Completion: WFS_CMD_IPM_SET_DESTINATION
7	WFS_CMD_IPM_ACTION_ITEM	- Print and deposit item in bin as specified by application in previous commands - Completion: WFS_CMD_IPM_ACTION_ITEM
8	WFS_CMD_IPM_GET_NEXT_ITEM	- If item successfully read then send one WFS_EXEE_IPM_MEDIADATA event for next media item. - Completion: WFS_CMD_IPM_GET_NEXT_ITEM
9	If the item was read successfully continue with step 3. Otherwise if there are no more items then continue with step 10.	
10	If more media is to be inserted: Goto 1, other wise continue with step 11	
11	WFS_CMD_IPM_MEDIA_IN_END	- End transaction
12		- Completion: WFS_CMD_IPM_MEDIA_IN_END

7.2.2 Bunch Media Processing (Some Media Items Returned)

	Application / Customer	**XFS IPM Service**
1	WFS_CMD_IPM_MEDIA_IN	- Event: WFS_EXEE_IPM_NOMEDIA - Wait for media insertion
2	Customer deposits a bunch of media items	- Event: WFS_EXEE_IPM_MEDIAINSERTED - Send one WFS_EXEE_IPM_MEDIADATA event for first media item. - Completion: WFS_CMD_IPM_MEDIA_IN
3	WFS_INF_IPM_TRANSACTION_STATUS	- Report media status and positions
4	If additional images are required then WFS_CMD_IPM_READ_IMAGE	- Reads data from the selected media item - Writes image data to the specified files. - Completion: WFS_CMD_IPM_READ_IMAGE
5	If Item is to be kept continue at step 6, otherwise continue at step 10	
6	WFS_CMD_IPM_PRINT_TEXT	- Specifies if the item is to be stamped and specifies the data to print on the selected media item. - Completion: WFS_CMD_IPM_PRINT_TEXT
7	WFS_CMD_IPM_SET_DESTINATION	- Specifies the destination of the selected media item. - Completion: WFS_CMD_IPM_SET_DESTINATION
8	WFS_CMD_IPM_ACTION_ITEM	- Print and deposit item in bin as specified by application in previous commands - Completion: WFS_CMD_IPM_ ACTION_ITEM
9	Continue at step 13	
10	WFS_CMD_IPM_SET_DESTINATION	- Specifies the destination of the selected media item as Return to Customer - Completion: WFS_CMD_IPM_SET_DESTINATION
11	WFS_CMD_IPM_ACTION_ITEM	- Present the returned media item to the customer - Event: WFS_EXEE_IPM_MEDIAPRESENTED - Completion: WFS_CMD_IPM_ ACTION_ITEM
12	Customer takes returned item.	- WFS_SRVE_IPM_MEDIATAKEN
13	WFS_CMD_IPM_GET_NEXT_ITEM	- If item successfully read then send one WFS_EXEE_IPM_MEDIADATA event for next media item. - Completion: WFS_CMD_IPM_GET_NEXT_ITEM
14	If the item was read successfully continue with step 3. Otherwise if there are no more items then continue with step 15	
15	If more media is to be inserted: Goto 1, other wise continue with step 16	
16	WFS_CMD_IPM_MEDIA_IN_END	- End transaction
17		- Completion: WFS_CMD_IPM_MEDIA_IN_END

7.2.3 Bunch Media Processing with Errors

	Application / Customer	**XFS IPM Service**
1	WFS_CMD_IPM_MEDIA_IN	- Event: WFS_EXEE_IPM_NOMEDIA - Wait for media insertion
2	Customer deposits a bunch of media items	- Event: WFS_EXEE_IPM_MEDIAINSERTED - Send one WFS_EXEE_IPM_MEDIADATA event for first media item. - Completion: WFS_CMD_IPM_MEDIA_IN
3	WFS_INF_IPM_TRANSACTION_STATUS	- Report media status and positions
4	If additional images are required then WFS_CMD_IPM_READ_IMAGE	- Reads data from the selected media item - Writes image data to the specified files. - Completion: WFS_CMD_IPM_READ_IMAGE
5	If Item is to be kept continue at step 6, otherwise continue at step 10	
6	WFS_CMD_IPM_PRINT_TEXT	- Specifies if the item is to be stamped and specifies the data to print on the selected media item. - Completion: WFS_CMD_IPM_PRINT_TEXT
7	WFS_CMD_IPM_SET_DESTINATION	- Specifies the destination of the selected media item. - Completion: WFS_CMD_IPM_SET_DESTINATION
8	WFS_CMD_IPM_ACTION_ITEM	- Print and deposit item in bin as specified by application in previous commands - Completion: WFS_CMD_IPM_ACTION_ITEM
9	Continue at step 13	-
10	WFS_CMD_IPM_SET_DESTINATION	- Specifies the destination of the selected media item as Return to Customer - Completion: WFS_CMD_IPM_SET_DESTINATION
11	WFS_CMD_IPM_ACTION_ITEM	- Present the returned media item to the customer - Event: WFS_EXEE_IPM_MEDIAPRESENTED - Completion: WFS_CMD_IPM_ ACTION_ITEM
12	Customer takes returned item.	- WFS_SRVE_IPM_MEDIATAKEN
13	WFS_CMD_IPM_GET_NEXT_ITEM	- Event: WFS_EXEE_IPM_MEDIAREFUSED (Reason= WFS_IPM_REFUSED_CODELINEINVALID) if code line could not be read - Present the media items to the customer - Completion: WFS_CMD_IPM_GET_NEXT_ITEM (ITEM REFUSED)
14	WFS_CMD_IPM_PRESENT_MEDIA	- Event: WFS_EXEE_IPM_MEDIAPRESENTED
15		- Completion: WFS_CMD_IPM_PRESENT_MEDIA
16	Customer takes returned media item	- Event: WFS_SRVE_IPM_MEDIATAKEN
17	If the item was REFUSED continue with step 13. If the item was read successfully continue with step 3. Otherwise if there are no more items then continue with step 18	
18	If more media is to be inserted: Goto 1, other wise continue with step 19	
19	WFS_CMD_IPM_MEDIA_IN_END	- End transaction
20		- Completion: WFS_CMD_IPM_MEDIA_IN_END

8. C-Header File

```
/******************************************************************************
*                                                                             *
* xfsipm.h    XFS - Item Processing Module (IPM) definitions                  *
*                                                                             *
*             Version 3.10  (29/11/2007)                                      *
*                                                                             *
******************************************************************************/

#ifndef __INC_XFSIPM__H
#define  __INC_XFSIPM__H

#ifdef __cplusplus
extern "C" {
#endif

#include <xfsapi.h>

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

/* Value of WFSIPMCAPS.wClass */

#define     WFS_SERVICE_CLASS_IPM               (16)
#define     WFS_SERVICE_CLASS_VERSION_IPM       (0x0A03) /* Version 3.10 */
#define     WFS_SERVICE_CLASS_NAME_IPM          "IPM"

#define     IPM_SERVICE_OFFSET                  (WFS_SERVICE_CLASS_IPM * 100)

/* IPM Info Commands */

#define     WFS_INF_IPM_STATUS                  (IPM_SERVICE_OFFSET + 1)
#define     WFS_INF_IPM_CAPABILITIES            (IPM_SERVICE_OFFSET + 2)
#define     WFS_INF_IPM_CODELINE_MAPPING        (IPM_SERVICE_OFFSET + 3)
#define     WFS_INF_IPM_MEDIA_BIN_INFO          (IPM_SERVICE_OFFSET + 4)
#define     WFS_INF_IPM_TRANSACTION_STATUS      (IPM_SERVICE_OFFSET + 5)

/* IPM Execute Commands */

#define     WFS_CMD_IPM_MEDIA_IN                (IPM_SERVICE_OFFSET + 1)
#define     WFS_CMD_IPM_MEDIA_IN_END            (IPM_SERVICE_OFFSET + 2)
#define     WFS_CMD_IPM_MEDIA_IN_ROLLBACK       (IPM_SERVICE_OFFSET + 3)
#define     WFS_CMD_IPM_READ_IMAGE              (IPM_SERVICE_OFFSET + 4)
#define     WFS_CMD_IPM_SET_DESTINATION         (IPM_SERVICE_OFFSET + 5)
#define     WFS_CMD_IPM_PRESENT_MEDIA           (IPM_SERVICE_OFFSET + 6)
#define     WFS_CMD_IPM_RETRACT_MEDIA           (IPM_SERVICE_OFFSET + 7)
#define     WFS_CMD_IPM_PRINT_TEXT              (IPM_SERVICE_OFFSET + 8)
#define     WFS_CMD_IPM_SET_MEDIA_BIN_INFO      (IPM_SERVICE_OFFSET + 9)
#define     WFS_CMD_IPM_RESET                   (IPM_SERVICE_OFFSET + 10)
#define     WFS_CMD_IPM_SET_GUIDANCE_LIGHT      (IPM_SERVICE_OFFSET + 11)
#define     WFS_CMD_IPM_GET_NEXT_ITEM           (IPM_SERVICE_OFFSET + 12)
#define     WFS_CMD_IPM_ACTION_ITEM             (IPM_SERVICE_OFFSET + 13)
#define     WFS_CMD_IPM_EXPEL_MEDIA             (IPM_SERVICE_OFFSET + 14)
#define     WFS_CMD_IPM_GET_IMAGE_AFTER_PRINT   (IPM_SERVICE_OFFSET + 15)
#define     WFS_CMD_IPM_ACCEPT_ITEM             (IPM_SERVICE_OFFSET + 16)
#define     WFS_CMD_IPM_SUPPLY_REPLENISH        (IPM_SERVICE_OFFSET + 17)
#define     WFS_CMD_IPM_POWER_SAVE_CONTROL      (IPM_SERVICE_OFFSET + 18)

/* IPM Messages */

#define     WFS_EXEE_IPM_NOMEDIA                (IPM_SERVICE_OFFSET + 1)
#define     WFS_EXEE_IPM_MEDIAINSERTED          (IPM_SERVICE_OFFSET + 2)
#define     WFS_USRE_IPM_MEDIABINTHRESHOLD      (IPM_SERVICE_OFFSET + 3)
#define     WFS_SRVE_IPM_MEDIABININFOCHANGED    (IPM_SERVICE_OFFSET + 4)
#define     WFS_EXEE_IPM_MEDIABINERROR          (IPM_SERVICE_OFFSET + 5)
#define     WFS_SRVE_IPM_MEDIATAKEN             (IPM_SERVICE_OFFSET + 6)
#define     WFS_USRE_IPM_TONERTHRESHOLD         (IPM_SERVICE_OFFSET + 7)
#define     WFS_USRE_IPM_SCANNERTHRESHOLD       (IPM_SERVICE_OFFSET + 8)
```

```
#define     WFS_USRE_IPM_INKTHRESHOLD           (IPM_SERVICE_OFFSET + 9)
#define     WFS_SRVE_IPM_MEDIADETECTED          (IPM_SERVICE_OFFSET + 10)
#define     WFS_EXEE_IPM_MEDIAPRESENTED         (IPM_SERVICE_OFFSET + 11)
#define     WFS_EXEE_IPM_MEDIAREFUSED           (IPM_SERVICE_OFFSET + 12)
#define     WFS_EXEE_IPM_MEDIADATA              (IPM_SERVICE_OFFSET + 13)
#define     WFS_USRE_IPM_MICRTHRESHOLD          (IPM_SERVICE_OFFSET + 14)
#define     WFS_EXEE_IPM_MEDIAREJECTED          (IPM_SERVICE_OFFSET + 15)
#define     WFS_SRVE_IPM_DEVICEPOSITION         (IPM_SERVICE_OFFSET + 16)
#define     WFS_SRVE_IPM_POWER_SAVE_CHANGE      (IPM_SERVICE_OFFSET + 17)

/* Values of WFSIPMSTATUS.fwDevice */

#define     WFS_IPM_DEVONLINE                   WFS_STAT_DEVONLINE
#define     WFS_IPM_DEVOFFLINE                  WFS_STAT_DEVOFFLINE
#define     WFS_IPM_DEVPOWEROFF                 WFS_STAT_DEVPOWEROFF
#define     WFS_IPM_DEVNODEVICE                 WFS_STAT_DEVNODEVICE
#define     WFS_IPM_DEVHWERROR                  WFS_STAT_DEVHWERROR
#define     WFS_IPM_DEVUSERERROR                WFS_STAT_DEVUSERERROR
#define     WFS_IPM_DEVBUSY                     WFS_STAT_DEVBUSY
#define     WFS_IPM_DEVFRAUDATTEMPT             WFS_STAT_DEVFRAUDATTEMPT

/* Values of WFSIPMSTATUS.wAcceptor */

#define     WFS_IPM_ACCBINOK                    (0)
#define     WFS_IPM_ACCBINSTATE                 (1)
#define     WFS_IPM_ACCBINSTOP                  (2)
#define     WFS_IPM_ACCBINUNKNOWN               (3)

/* Values of WFSIPMSTATUS.wMedia and
             WFSIPMMEDIADETECTED.wPosition */

#define     WFS_IPM_MEDIAPRESENT                (0)
#define     WFS_IPM_MEDIANOTPRESENT             (1)
#define     WFS_IPM_MEDIAJAMMED                 (2)
#define     WFS_IPM_MEDIANOTSUPP                (3)
#define     WFS_IPM_MEDIAUNKNOWN                (4)
#define     WFS_IPM_MEDIAPOSITION               (5)
#define     WFS_IPM_MEDIARETRACTED              (6)
#define     WFS_IPM_MEDIARETURNED               (7)

/* Values of WFSIPMSTATUS.wToner and
             WFSIPMTHRESHOLD.wThreshold */

#define     WFS_IPM_TONERFULL                   (0)
#define     WFS_IPM_TONERLOW                    (1)
#define     WFS_IPM_TONEROUT                    (2)
#define     WFS_IPM_TONERNOTSUPP                (3)
#define     WFS_IPM_TONERUNKNOWN                (4)

/* Values of WFSIPMSTATUS.wInk and
             WFSIPMTHRESHOLD.wThreshold */

#define     WFS_IPM_INKFULL                     (0)
#define     WFS_IPM_INKLOW                      (1)
#define     WFS_IPM_INKOUT                      (2)
#define     WFS_IPM_INKNOTSUPP                  (3)
#define     WFS_IPM_INKUNKNOWN                  (4)

/* Values of WFSIPMSTATUS.wFrontImageScanner,
             WFSIPMSTATUS.wBackImageScanner and
             WFSIPMSCANNERTHRESHOLD.wThreshold */

#define     WFS_IPM_SCANNEROK                   (0)
#define     WFS_IPM_SCANNERFADING               (1)
#define     WFS_IPM_SCANNERINOP                 (2)
#define     WFS_IPM_SCANNERNOTSUPP              (3)
#define     WFS_IPM_SCANNERUNKNOWN              (4)

/* Values of WFSIPMSTATUS.wMICRReader and
             WFSIPMTHRESHOLD.wThreshold */
```

```
#define     WFS_IPM_MICROK                      (0)
#define     WFS_IPM_MICRFADING                  (1)
#define     WFS_IPM_MICRINOP                    (2)
#define     WFS_IPM_MICRNOTSUPP                 (3)
#define     WFS_IPM_MICRUNKNOWN                 (4)

/* Values of WFSIPMSTATUS.wStacker  */

#define     WFS_IPM_STACKEREMPTY                (0)
#define     WFS_IPM_STACKERNOTEMPTY             (1)
#define     WFS_IPM_STACKERFULL                 (2)
#define     WFS_IPM_STACKERINOP                 (3)
#define     WFS_IPM_STACKERUNKNOWN              (4)
#define     WFS_IPM_STACKERNOTSUPP              (5)

/* Values of WFSIPMSTATUS.wReBuncher       */

#define     WFS_IPM_REBUNCHEREMPTY              (0)
#define     WFS_IPM_REBUNCHERNOTEMPTY           (1)
#define     WFS_IPM_REBUNCHERFULL               (2)
#define     WFS_IPM_REBUNCHERINOP               (3)
#define     WFS_IPM_REBUNCHERUNKNOWN            (4)
#define     WFS_IPM_REBUNCHERNOTSUPP            (5)

/* Values of WFSIPMSTATUS.wMediaFeeder */

#define     WFS_IPM_FEEDEREMPTY                 (0)
#define     WFS_IPM_FEEDERNOTEMPTY              (1)
#define     WFS_IPM_FEEDERINOP                  (2)
#define     WFS_IPM_FEEDERUNKNOWN               (3)
#define     WFS_IPM_FEEDERNOTSUPP               (4)

/* values of WFSIPMSTATUS.wDevicePosition
             WFSIPMDEVICEPOSITION.wPosition */

#define      WFS_IPM_DEVICEINPOSITION           (0)
#define      WFS_IPM_DEVICENOTINPOSITION        (1)
#define      WFS_IPM_DEVICEPOSUNKNOWN           (2)
#define      WFS_IPM_DEVICEPOSNOTSUPP           (3)

/* Values of WFSIPMTRANSSTATUS.usMediaOnStacker,
             WFSIPMTRANSSTATUS.usLastMediaInTotal, and
             WFSIPMTRANSSTATUS.usLastMediaAddedToStacker */

#define     WFS_IPM_MEDIANUMBERUNKNOWN          (0xFFFF)

/* Indices for WFSIPMSTATUS.lppPositions and
               WFSIPMCAPS.lppPositions       */

#define     WFS_IPM_POSINPUT                    (0)
#define     WFS_IPM_POSOUTPUT                   (1)
#define     WFS_IPM_POSREFUSED                  (2)

/* Values of WFSIPMPOS.wShutter */

#define     WFS_IPM_SHTCLOSED                   (0)
#define     WFS_IPM_SHTOPEN                     (1)
#define     WFS_IPM_SHTJAMMED                   (2)
#define     WFS_IPM_SHTUNKNOWN                  (3)
#define     WFS_IPM_SHTNOTSUPPORTED             (4)

/* Values of WFSIPMPOS.wPositionStatus */

#define     WFS_IPM_PSEMPTY                     (0)
#define     WFS_IPM_PSNOTEMPTY                  (1)
#define     WFS_IPM_PSUNKNOWN                   (2)
#define     WFS_IPM_PSNOTSUPPORTED              (3)

/* Values of WFSIPMPOS.wTransport */
```

```
#define     WFS_IPM_TPOK                        (0)
#define     WFS_IPM_TPINOP                      (1)
#define     WFS_IPM_TPUNKNOWN                   (2)
#define     WFS_IPM_TPNOTSUPPORTED              (3)

/* Values of WFSIPMPOS.wTransportMediaStatus */

#define     WFS_IPM_TPMEDIAEMPTY                (0)
#define     WFS_IPM_TPMEDIANOTEMPTY             (1)
#define     WFS_IPM_TPMEDIAUNKNOWN              (2)
#define     WFS_IPM_TPMEDIANOTSUPPORTED         (3)

/* Size and max index of dwGuidLights array */

#define     WFS_IPM_GUIDLIGHTS_SIZE             (32)
#define     WFS_IPM_GUIDLIGHTS_MAX              (WFS_IPM_GUIDLIGHTS_SIZE - 1)

/* Indices of WFSIPMSTATUS.dwGuidLights [...] and
              WFSIPMCAPS.dwGuidLights [...] */

#define     WFS_IPM_GUIDANCE_MEDIAIN            (0)
#define     WFS_IPM_GUIDANCE_MEDIAOUT           (1)
#define     WFS_IPM_GUIDANCE_MEDIAREFUSED       (2)

/* Values of WFSIPMSTATUS.dwGuidLights [...] and
             WFSIPMCAPS.dwGuidLights [...] */

#define     WFS_IPM_GUIDANCE_NOT_AVAILABLE      (0x00000000)
#define     WFS_IPM_GUIDANCE_OFF                (0x00000001)
#define     WFS_IPM_GUIDANCE_SLOW_FLASH         (0x00000004)
#define     WFS_IPM_GUIDANCE_MEDIUM_FLASH       (0x00000008)
#define     WFS_IPM_GUIDANCE_QUICK_FLASH        (0x00000010)
#define     WFS_IPM_GUIDANCE_CONTINUOUS         (0x00000080)
#define     WFS_IPM_GUIDANCE_RED                (0x00000100)
#define     WFS_IPM_GUIDANCE_GREEN              (0x00000200)
#define     WFS_IPM_GUIDANCE_YELLOW             (0x00000400)
#define     WFS_IPM_GUIDANCE_BLUE               (0x00000800)
#define     WFS_IPM_GUIDANCE_CYAN               (0x00001000)
#define     WFS_IPM_GUIDANCE_MAGENTA            (0x00002000)
#define     WFS_IPM_GUIDANCE_WHITE              (0x00004000)

/* values of WFSIPMCAPS.fwType */

#define     WFS_IPM_TYPESINGLEMEDIAINPUT        (0x0001)
#define     WFS_IPM_TYPEBUNCHMEDIAINPUT         (0x0002)

/* values of WFSIPMCAPS.fwRetractLocation
             WFSIPMPOSCAPS.fwRetractAreas
             WFSIPMRETRACTMEDIA.wRetractLocation
             WFSIPMRETRACTMEDIAOUT.wRetractLocation        */

#define     WFS_IPM_CTRLRETRACTTOBIN            (0x0001)
#define     WFS_IPM_CTRLRETRACTTOTRANSPORT      (0x0002)
#define     WFS_IPM_CTRLRETRACTTOSTACKER        (0x0004)
#define     WFS_IPM_CTRLRETRACTTOREBUNCHER      (0x0008)

/* Values of WFSIPMCAPS.fwResetControl and
             WFSIPMRESET.wMediaControl */

#define     WFS_IPM_RESETEJECT                  (0x0001)
#define     WFS_IPM_RESETRETRACTTOBIN           (0x0002)
#define     WFS_IPM_RESETRETRACTTOTRANSPORT     (0x0004)
#define     WFS_IPM_RESETRETRACTTOREBUNCHER     (0x0008)

/* values of WFSIPMCAPS.fwImageType,
             WFSIPMIMAGEREQUEST.wFrontImageFormat and
             WFSIPMIMAGEREQUEST.wBackImageFormat */

#define     WFS_IPM_IMAGETIF                    (0x0001)
```

```
#define     WFS_IPM_IMAGEWMF                    (0x0002)
#define     WFS_IPM_IMAGEBMP                    (0x0004)
#define     WFS_IPM_IMAGEJPG                    (0x0008)

/* Values of WFSIPMCAPS.fwFrontImageColorFormat,
             WFSIPMCAPS.fwBackImageColorFormat and
             WFSIPMIMAGEREQUEST.wImageColorFormat */

#define     WFS_IPM_IMAGECOLORBINARY            (0x0001)
#define     WFS_IPM_IMAGECOLORGRAYSCALE         (0x0002)
#define     WFS_IPM_IMAGECOLORFULL              (0x0004)

/* Values of WFSIPMCAPS.fwFrontScanColor,
             WFSIPMCAPS.fwBackScanColor,
             WFSIPMCAPS.wDefaultFrontScanColor,
             WFSIPMCAPS.wDefaultBackScanColor and
             WFSIPMIMAGEREQUEST.wImageScanColor  */

#define     WFS_IPM_SCANCOLORDEFAULT            (0x0000)
#define     WFS_IPM_SCANCOLORRED                (0x0001)
#define     WFS_IPM_SCANCOLORBLUE               (0x0002)
#define     WFS_IPM_SCANCOLORGREEN              (0x0004)
#define     WFS_IPM_SCANCOLORYELLOW             (0x0008)
#define     WFS_IPM_SCANCOLORWHITE              (0x0010)

/* Values of WFSIPMCAPS.fwCodelineFormat and
             WFSIPMMEDIAINREQUEST.wCodelineFormat */

#define     WFS_IPM_CODELINECMC7                (0x0001)
#define     WFS_IPM_CODELINEE13B                (0x0002)
#define     WFS_IPM_CODELINEOCR                 (0x0004)

/* Values of WFSIPMCAPS.fwDataSource,
             WFSIPMIMAGEREQUEST.wImageSource, and
             WFSIPMIMAGEDATA.wImageSource */

#define     WFS_IPM_IMAGEFRONT                  (0x0001)
#define     WFS_IPM_IMAGEBACK                   (0x0002)
#define     WFS_IPM_CODELINE                    (0x0004)

/* Values of WFSIPMMEDIABIN.fwType */

#define     WFS_IPM_TYPEMEDIAIN                 (0x0001)
#define     WFS_IPM_TYPERETRACT                 (0x0002)

/* Values of WFSIPMMEDIABIN.wMediaType */

#define     WFS_IPM_MEDIATYPIPM                 (0x0001)
#define     WFS_IPM_MEDIATYPCOMPOUND            (0x0002)

/* Values of WFSIPMMEDIABIN.usStatus */

#define     WFS_IPM_STATMBOK                    (1)
#define     WFS_IPM_STATMBFULL                  (2)
#define     WFS_IPM_STATMBHIGH                  (3)
#define     WFS_IPM_STATMBINOP                  (4)
#define     WFS_IPM_STATMBMISSING               (5)
#define     WFS_IPM_STATMBUNKNOWN               (6)

/* Values of WFSIPMTRANSSTATUS.wMediaInTransaction */

#define     WFS_IPM_MITOK                       (0)
#define     WFS_IPM_MITACTIVE                   (1)
#define     WFS_IPM_MITROLLBACK                 (2)
#define     WFS_IPM_MITROLLBACKAFTERDEPOSIT     (3)
#define     WFS_IPM_MITRETRACT                  (4)
#define     WFS_IPM_MITFAILURE                  (5)
#define     WFS_IPM_MITUNKNOWN                  (6)
#define     WFS_IPM_MITRESET                    (7)
```

```
/* Values of WFSIPMMEDIASTATUS.wMediaLocation */

#define     WFS_IPM_LOCATION_DEVICE             (0)
#define     WFS_IPM_LOCATION_BIN                (1)
#define     WFS_IPM_LOCATION_CUSTOMER           (2)
#define     WFS_IPM_LOCATION_UNKNOWN            (3)

/* Values of WFSIPMMEDIASTATUS.wCustomerAccess */

#define     WFS_IPM_ACCESSUNKNOWN               (0)
#define     WFS_IPM_ACCESSCUSTOMER              (1)
#define     WFS_IPM_ACCESSNONE                  (2)

/* Values of WFSIPMIMAGEDATA.wImageStatus */

#define     WFS_IPM_DATAOK                      (0)
#define     WFS_IPM_DATASRCNOTSUPP              (1)
#define     WFS_IPM_DATASRCMISSING              (2)

/* Values of WFSIPMMEDIADATA.wMagneticReadIndicator */

#define     WFS_IPM_MRI_MICR                    (0)
#define     WFS_IPM_MRI_NOT_MICR                (1)
#define     WFS_IPM_MRI_NO_MICR                 (2)
#define     WFS_IPM_MRI_UNKNOWN                 (3)
#define     WFS_IPM_MRI_NOTMICRFORMAT           (4)
#define     WFS_IPM_MRI_NOT_READ                (5)

/* Values of WFSIPMCAPS.fwInsertOrientation  and
             WFSIPMMEDIADATA.fwInsertOrientation */

#define     WFS_IPM_INSUNKNOWN                  (0x0000)
#define     WFS_IPM_INSCODELINERIGHT            (0x0001)
#define     WFS_IPM_INSCODELINELEFT             (0x0002)
#define     WFS_IPM_INSCODELINEBOTTOM           (0x0004)
#define     WFS_IPM_INSCODELINETOP              (0x0008)
#define     WFS_IPM_INSFACEUP                   (0x0010)
#define     WFS_IPM_INSFACEDOWN                 (0x0020)

/* Values of WFSIPMMEDIADATA.wMediaValidity */

#define     WFS_IPM_ITEMOK                      (0)
#define     WFS_IPM_ITEMSUSPECT                 (1)
#define     WFS_IPM_ITEMUNKNOWN                 (2)
#define     WFS_IPM_ITEMNOVALIDATION            (3)

/* values of WFSIPMSUPPLYREPLEN.fwSupplyReplen */
#define  WFS_IPM_REPLEN_TONER                   (0x0001)
#define  WFS_IPM_REPLEN_INK                     (0x0002)

/* Values of WFSIPMMEDIAREFUSED.wReason */

#define     WFS_IPM_REFUSED_FOREIGNITEMS        (1)
#define     WFS_IPM_REFUSED_STACKERFULL         (2)
#define     WFS_IPM_REFUSED_CODELINEINVALID     (3)
#define     WFS_IPM_REFUSED_INVALIDMEDIA        (4)
#define     WFS_IPM_REFUSED_TOOLONG             (5)
#define     WFS_IPM_REFUSED_TOOSHORT            (6)
#define     WFS_IPM_REFUSED_TOOWIDE             (7)
#define     WFS_IPM_REFUSED_TOONARROW           (8)
#define     WFS_IPM_REFUSED_TOOTHICK            (9)
#define     WFS_IPM_REFUSED_INVALIDORIENTATION  (10)
#define     WFS_IPM_REFUSED_DOUBLEDETECT        (11)
#define     WFS_IPM_REFUSED_REFUSEPOSFULL       (12)
#define     WFS_IPM_REFUSED_RETURNBLOCKED       (13)
#define     WFS_IPM_REFUSED_INVALIDBUNCH        (14)
#define     WFS_IPM_REFUSED_OTHERITEM           (15)
#define     WFS_IPM_REFUSED_OTHERBUNCH          (16)
#define     WFS_IPM_REFUSED_JAMMING             (17)
#define     WFS_IPM_REFUSED_METAL               (18)
```

```
/* Values of WFSIPMMEDIAREFUSED.wMediaLocation and
             WFSIPMPRESENTMEDIA.wPosition */

#define     WFS_IPM_REFUSE_INPUT                 (1)
#define     WFS_IPM_REFUSE_REFUSED               (2)
#define     WFS_IPM_REFUSE_REBUNCHER             (3)

/* Values of WFSIPMMBERROR.wFailure */

#define     WFS_IPM_MEDIABINJAMMED               (1)
#define     WFS_IPM_MEDIABINERROR                (2)
#define     WFS_IPM_MEDIABINFULL                 (3)
#define     WFS_IPM_MEDIABINNOTCONF              (4)
#define     WFS_IPM_MEDIABININVALID              (5)
#define     WFS_IPM_MEDIABINCONFIG               (6)
#define     WFS_IPM_MEDIABINFEEDPROBLEM          (7)

/* Values of WFSIPMMEDIAREJECTED.wReason) */

#define     WFS_IPM_REJECT_LONG                  (1)
#define     WFS_IPM_REJECT_THICK                 (2)
#define     WFS_IPM_REJECT_DOUBLE                (3)
#define     WFS_IPM_REJECT_TRANSPORT             (4)
#define     WFS_IPM_REJECT_SHUTTER               (5)
#define     WFS_IPM_REJECT_REMOVED               (6)
#define     WFS_IPM_REJECT_METAL                 (7)
#define     WFS_IPM_REJECT_FOREIGNITEMS          (8)
#define     WFS_IPM_REJECT_OTHER                 (9)

/* Values of WFSIPMSCANNERTHRESHOLD.wScanner */

#define     WFS_IPM_FRONTSCANNER                 (1)
#define     WFS_IPM_BACKSCANNER                  (2)

/* XFS IPM Errors */

#define     WFS_ERR_IPM_NOMEDIAPRESENT           (-(IPM_SERVICE_OFFSET + 1))
#define     WFS_ERR_IPM_MEDIABINFULL             (-(IPM_SERVICE_OFFSET + 2))
#define     WFS_ERR_IPM_STACKERFULL              (-(IPM_SERVICE_OFFSET + 3))
#define     WFS_ERR_IPM_SHUTTERFAIL              (-(IPM_SERVICE_OFFSET + 4))
#define     WFS_ERR_IPM_MEDIAJAMMED              (-(IPM_SERVICE_OFFSET + 5))
#define     WFS_ERR_IPM_FILEIOERROR              (-(IPM_SERVICE_OFFSET + 6))
#define     WFS_ERR_IPM_INKOUT                   (-(IPM_SERVICE_OFFSET + 7))
#define     WFS_ERR_IPM_TONEROUT                 (-(IPM_SERVICE_OFFSET + 8))
#define     WFS_ERR_IPM_SCANNERINOP              (-(IPM_SERVICE_OFFSET + 9))
#define     WFS_ERR_IPM_MICRINOP                 (-(IPM_SERVICE_OFFSET + 10))
#define     WFS_ERR_IPM_SEQUENCEINVALID          (-(IPM_SERVICE_OFFSET + 11))
#define     WFS_ERR_IPM_INVALID_PORT             (-(IPM_SERVICE_OFFSET + 12))
#define     WFS_ERR_IPM_FOREIGNITEMSDETECTED     (-(IPM_SERVICE_OFFSET + 13))
#define     WFS_ERR_IPM_INVALIDMEDIAID           (-(IPM_SERVICE_OFFSET + 14))
#define     WFS_ERR_IPM_MEDIABINERROR            (-(IPM_SERVICE_OFFSET + 15))
#define     WFS_ERR_IPM_POSITIONNOTEMPTY         (-(IPM_SERVICE_OFFSET + 16))
#define     WFS_ERR_IPM_INVALIDBIN               (-(IPM_SERVICE_OFFSET + 17))
#define     WFS_ERR_IPM_NOBIN                    (-(IPM_SERVICE_OFFSET + 18))
#define     WFS_ERR_IPM_REFUSEDITEMS             (-(IPM_SERVICE_OFFSET + 19))
#define     WFS_ERR_IPM_ALLBINSFULL              (-(IPM_SERVICE_OFFSET + 20))
#define     WFS_ERR_IPM_FEEDERNOTEMPTY           (-(IPM_SERVICE_OFFSET + 21))
#define     WFS_ERR_IPM_MEDIAREJECTED            (-(IPM_SERVICE_OFFSET + 22))
#define     WFS_ERR_IPM_FEEDERINOPERATIVE        (-(IPM_SERVICE_OFFSET + 23))
#define     WFS_ERR_IPM_MEDIAPRESENT             (-(IPM_SERVICE_OFFSET + 24))
#define     WFS_ERR_IPM_POWERSAVETOOSHORT        (-(IPM_SERVICE_OFFSET + 25))
#define     WFS_ERR_IPM_POWERSAVEMEDIAPRESENT    (-(IPM_SERVICE_OFFSET + 26))

/*=================================================================*/
/* IPM Info Command Structures */
/*=================================================================*/
typedef struct _wfs_ipm_pos
{
    WORD                   wShutter;
    WORD                   wPositionStatus;
    WORD                   wTransport;
    WORD                   wTransportMediaStatus;
} WFSIPMPOS, *LPWFSIPMPOS;

typedef struct _wfs_ipm_status
{
    WORD                   fwDevice;
    WORD                   wAcceptor;
    WORD                   wMedia;
    WORD                   wToner;
    WORD                   wInk;
    WORD                   wFrontImageScanner;
    WORD                   wBackImageScanner;
    WORD                   wMICRReader;
    WORD                   wStacker;
    WORD                   wReBuncher;
    WORD                   wMediaFeeder;
    LPWFSIPMPOS           *lppPositions;
    DWORD                  dwGuidLights[WFS_IPM_GUIDLIGHTS_SIZE];
    LPSTR                  lpszExtra;
    WORD                   wDevicePosition;
    USHORT                 usPowerSaveRecoveryTime;
} WFSIPMSTATUS, *LPWFSIPMSTATUS;

typedef struct _wfs_ipm_print_size
{
    WORD                   wRows;
    WORD                   wCols;
} WFSIPMPRINTSIZE, *LPWFSIPMPRINTSIZE;

typedef struct _wfs_ipm_pos_caps
{
    BOOL                   bItemsTakenSensor;
    BOOL                   bItemsInsertedSensor;
    WORD                   fwRetractAreas;
} WFSIPMPOSCAPS, *LPWFSIPMPOSCAPS;

/* WFS_INF_IPM_CAPABILITIES output structures */
typedef struct _wfs_ipm_caps
{
    WORD                   wClass;
    WORD                   fwType;
    BOOL                   bCompound;
    USHORT                 usMaxMediaOnStacker;
    LPWFSIPMPRINTSIZE      lpPrintSize;
    BOOL                   bStamp;
    BOOL                   bRescan;
    BOOL                   bPresentControl;
    BOOL                   bApplicationRefuse;
    WORD                   fwRetractLocation;
    WORD                   fwResetControl;
    BOOL                   bRetractCountsItems;
    WORD                   fwImageType;
    WORD                   fwFrontImageColorFormat;
    WORD                   fwBackImageColorFormat;
    WORD                   fwFrontScanColor;
    WORD                   wDefaultFrontScanColor;
    WORD                   fwBackScanColor;
    WORD                   wDefaultBackScanColor;
    WORD                   fwCodelineFormat;
    WORD                   fwDataSource;
    WORD                   fwInsertOrientation;
    LPWFSIPMPOSCAPS       *lppPositions;
    DWORD                  dwGuidLights[WFS_IPM_GUIDLIGHTS_SIZE];
    LPSTR                  lpszExtra;
    BOOL                   bPowerSaveControl;
} WFSIPMCAPS, *LPWFSIPMCAPS;

typedef struct _wfs_ipm_hex_data
```

```
{
    USHORT                  usLength;
    LPBYTE                  lpbData;
} WFSIPMXDATA, *LPWFSIPMXDATA;

/* WFS_INF_IPM_CODELINE_MAPPING input and output structures */
typedef struct _wfs_ipm_codeline_mapping
{
    WORD                    wCodelineFormat;
} WFSIPMCODELINEMAPPING, *LPWFSIPMCODELINEMAPPING;

typedef struct _wfs_ipm_codeline_mapping_out
{
    WORD                    wCodelineFormat;
    LPWFSIPMXDATA           lpxCharMapping;
} WFSIPMCODELINEMAPPINGOUT, *LPWFSIPMCODELINEMAPPINGOUT;

/* WFS_INF_IPM_MEDIA_BIN_INFO output structures */
typedef struct _wfs_ipm_media_bin
{
    USHORT                  usBinNumber;
    LPSTR                   lpstrPositionName;
    WORD                    fwType;
    WORD                    wMediaType;
    LPSTR                   lpstrBinID;
    ULONG                   ulMediaInCount;
    ULONG                   ulCount;
    ULONG                   ulRetractOperations;
    BOOL                    bHardwareSensors;
    ULONG                   ulMaximumItems;
    ULONG                   ulMaximumRetractOperations;
    USHORT                  usStatus;
    LPSTR                   lpszExtra;
} WFSIPMMEDIABIN, *LPWFSIPMMEDIABIN;

typedef struct _wfs_ipm_media_bin_info
{
    USHORT                  usCount;
    LPWFSIPMMEDIABIN       *lppMediaBin;
} WFSIPMMEDIABININFO, *LPWFSIPMMEDIABININFO;

typedef struct _wfs_ipm_image_data
{
    WORD                    wImageSource;
    WORD                    wImageType;
    WORD                    wImageColorFormat;
    WORD                    wImageScanColor;
    WORD                    wImageStatus;
    LPSTR                   lpszImageFile;
} WFSIPMIMAGEDATA, *LPWFSIPMIMAGEDATA;

typedef struct _wfs_ipm_media_size
{
    ULONG                   ulSizeX;
    ULONG                   ulSizeY;
} WFSIPMMEDIASIZE, *LPWFSIPMMEDIASIZE;

typedef struct _wfs_ipm_mediastatus
{
    USHORT                  usMediaID;
    WORD                    wMediaLocation;
    USHORT                  usBinNumber;
    ULONG                   ulCodelineDataLength;
    LPBYTE                  lpbCodelineData;
    WORD                    wMagneticReadIndicator;
    LPWFSIPMIMAGEDATA      *lppImage;
    WORD                    fwInsertOrientation;
    LPWFSIPMMEDIASIZE       lpMediaSize;
    WORD                    wMediaValidity;
    WORD                    wCustomerAccess;
```

```
} WFSIPMMEDIASTATUS, *LPWFSIPMMEDIASTATUS;

/* WFS_INF_IPM_TRANSACTION_STATUS output structures */
typedef struct _wfs_ipm_trans_status
{
    WORD                   wMediaInTransaction;
    USHORT                 usMediaOnStacker;
    USHORT                 usLastMediaInTotal;
    USHORT                 usLastMediaAddedToStacker;
    USHORT                 usTotalItems;
    USHORT                 usTotalItemsRefused;
    USHORT                 usTotalBunchesRefused;
    LPWFSIPMMEDIASTATUS   *lppMediaInfo;
    LPSTR                  lpszExtra;
} WFSIPMTRANSSTATUS, *LPWFSIPMTRANSSTATUS;

/*=================================================================*/
/* IPM Execute Command Structures */
/*=================================================================*/
typedef struct _wfs_ipm_image_request
{
    WORD                   wImageSource;
    WORD                   wImageType;
    WORD                   wImageColorFormat;
    WORD                   wImageScanColor;
    LPSTR                  lpszImagePath;
} WFSIPMIMAGEREQUEST, *LPWFSIPMIMAGEREQUEST;

typedef struct _wfs_ipm_media_in_request
{
    WORD                   wCodelineFormat;
    LPWFSIPMIMAGEREQUEST  *lppImage;
    USHORT                 usMaxMediaOnStacker;
    BOOL                   bApplicationRefuse;
} WFSIPMMEDIAINREQUEST, *LPWFSIPMMEDIAINREQUEST;

typedef struct _wfs_ipm_media_in
{
    USHORT                 usMediaOnStacker;
    USHORT                 usLastMedia;
    USHORT                 usLastMediaOnStacker;
    WORD                   wMediaFeeder;
} WFSIPMMEDIAIN, *LPWFSIPMMEDIAIN;

/* WFS_CMD_IPM_MEDIA_IN_END structures */
typedef struct _wfs_ipm_media_in_end
{
    USHORT                 usItemsReturned;
    USHORT                 usItemsRefused;
    USHORT                 usBunchesRefused;
    LPWFSIPMMEDIABININFO   lpMediaBinInfo;
} WFSIPMMEDIAINEND, *LPWFSIPMMEDIAINEND;

typedef struct _wfs_ipm_read_image_request
{
    USHORT                 usMediaID;
    WORD                   wCodelineFormat;
    LPWFSIPMIMAGEREQUEST  *lppImage;
} WFSIPMREADIMAGEIN, *LPWFSIPMREADIMAGEIN;

typedef struct _wfs_ipm_mediadata
{
    USHORT                 usMediaID;
    ULONG                  ulCodelineDataLength;
    LPBYTE                 lpbCodelineData;
    WORD                   wMagneticReadIndicator;
    LPWFSIPMIMAGEDATA     *lppImage;
    WORD                   fwInsertOrientation;
    LPWFSIPMMEDIASIZE      lpMediaSize;
    WORD                   wMediaValidity;
```

```
} WFSIPMMEDIADATA, *LPWFSIPMMEDIADATA;

/* WFS_CMD_IPM_SET_DESTINATION structures */
typedef struct _wfs_ipm_set_destination
{
    USHORT                  usMediaID;
    USHORT                  usBinNumber;
} WFSIPMSETDESTINATION, *LPWFSIPMSETDESTINATION;

typedef struct _wfs_ipm_next_item_out
{
    WORD                    wMediaFeeder;
} WFSIPMNEXTITEMOUT, *LPWFSIPMNEXTITEMOUT;

/* WFS_CMD_IPM_PRESENT_MEDIA structures */
typedef struct _wfs_ipm_present_media
{
    WORD                    wPosition;
} WFSIPMPRESENTMEDIA, *LPWFSIPMPRESENTMEDIA;

/* WFS_CMD_IPM_RETRACT_MEDIA structures */
typedef struct _wfs_ipm_retract_media
{
    WORD                    wRetractLocation;
    USHORT                  usBinNumber;
} WFSIPMRETRACTMEDIA, *LPWFSIPMRETRACTMEDIA;

typedef struct _wfs_ipm_retract_media_out
{
    USHORT                  usMedia;
    WORD                    wRetractLocation;
    USHORT                  usBinNumber;
} WFSIPMRETRACTMEDIAOUT, *LPWFSIPMRETRACTMEDIAOUT;

/* WFS_CMD_IPM_PRINT_TEXT structures */
typedef struct _wfs_ipm_print_text
{
    USHORT                  usMediaID;
    BOOL                    bStamp;
    LPWSTR                  lpszPrintData;
} WFSIPMPRINTTEXT, *LPWFSIPMPRINTTEXT;

/* WFS_CMD_IPM_GET_IMAGE_AFTER_PRINT structures */
typedef struct _wfs_ipm_get_image_after_print
{
    USHORT                  usMediaID;
    LPWFSIPMIMAGEREQUEST  *lppImage;
} WFSIPMGETIMAGEAFTERPRINT, *LPWFSIPMGETIMAGEAFTERPRINT;

/* WFS_CMD_IPM_ACCEPT_ITEM structures */
typedef struct _wfs_ipm_accept_item
{
    BOOL                    bAccept;
} WFSIPMACCEPTITEM, *LPWFSIPMACCEPTITEM;

/* WFS_CMD_IPM_RESET structures */
typedef struct _wfs_ipm_reset
{
    WORD                    wMediaControl;
    USHORT                  usBinNumber;
} WFSIPMRESET, *LPWFSIPMRESET;

/* WFS_CMD_IPM_SUPPLY_REPLENISH structures */
typedef struct _wfs_ipm_supply_replen
{
    WORD                    fwSupplyReplen;
} WFSIPMSUPPLYREPLEN, *LPWFSIPMSUPPLYREPLEN;

/* WFS_CMD_IPM_SET_GUIDANCE_LIGHT structures */
typedef struct _wfs_ipm_set_guidlight
```

```
{
    WORD                    wGuidLight;
    DWORD                   dwCommand;
} WFSIPMSETGUIDLIGHT, *LPWFSIPMSETGUIDLIGHT;

/* WFS_CMD_IPM_POWER_SAVE_CONTROL structure */
typedef struct _wfs_ipm_power_save_control
{
    USHORT                  usMaxPowerSaveRecoveryTime;
} WFSIPMPOWERSAVECONTROL, *LPWFSIPMPOWERSAVECONTROL;

/*=================================================================*/
/* IPM Message Structures */
/*=================================================================*/

/* WFS_EXEE_IPM_MEDIABINERROR structure */
typedef struct _wfs_ipm_mb_error
{
    WORD                    wFailure;
    LPWFSIPMMEDIABIN        lpMediaBin;
} WFSIPMMBERROR, *LPWFSIPMMBERROR;

/* WFS_SRVE_IPM_MEDIATAKEN structure */
typedef struct _wfs_ipm_position
{
    WORD                    wPosition;
} WFSIPMPOSITION, *LPWFSIPMPOSITION;

/* WFS_USRE_IPM_TONERTHRESHOLD,
   WFS_USRE_IPM_INKTHRESHOLD structures */
typedef struct _wfs_ipm_threshold
{
    WORD                    wThreshold;
} WFSIPMTHRESHOLD, *LPWFSIPMTHRESHOLD;

/* WFS_USRE_IPM_SCANNERTHRESHOLD structure */
typedef struct _wfs_ipm_scanner_threshold
{
    WORD                    wScanner;
    WORD                    wThreshold;
} WFSIPMSCANNERTHRESHOLD, *LPWFSIPMSCANNERTHRESHOLD;

/* WFS_SRVE_IPM_MEDIADETECTED structure */
typedef struct _wfs_ipm_media_detected
{
    WORD                    wPosition;
    USHORT                  usRetractBinNumber;
} WFSIPMMEDIADETECTED, *LPWFSIPMMEDIADETECTED;

/* WFS_EXEE_IPM_MEDIAPRESENTED structure */
typedef struct _wfs_ipm_media_presented
{
    WORD                    wPosition;
    USHORT                  usBunchIndex;
    USHORT                  usTotalBunches;
} WFSIPMMEDIAPRESENTED, *LPWFSIPMMEDIAPRESENTED;

/* WFS_EXEE_IPM_MEDIAREFUSED structure */
typedef struct _wfs_ipm_media_refused
{
    WORD                    wReason;
    WORD                    wMediaLocation;
    BOOL                    bPresentRequired;
    LPWFSIPMMEDIASIZE       lpMediaSize;
} WFSIPMMEDIAREFUSED, *LPWFSIPMMEDIAREFUSED;

/* WFS_EXEE_IPM_MEDIAREJECTED structure */
typedef struct _wfs_ipm_media_rejected
{
    WORD                    wReason;
} WFSIPMMEDIAREJECTED, *LPWFSIPMMEDIAREJECTED;

/* WFS_SRVE_IPM_DEVICEPOSITION structure */
typedef struct _wfs_ipm_device_position
{
    WORD                   wPosition;
} WFSIPMDEVICEPOSITION, *LPWFSIPMDEVICEPOSITION;

/* WFS_SRVE_IPM_POWERSAVECHANGE structure */
typedef struct _wfs_ipm_power_save_change
{
    USHORT          usPowerSaveRecoveryTime;
} WFSIPMPOWERSAVECHANGE, *LPWFSIPMPOWERSAVECHANGE;

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

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

#endif    /* __INC_XFSIPM__H */
```