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

# WORKSHOP AGREEMENT

**CWA 14050-8**

November 2000

ICS 35.200; 35.240.40

Extensions for Financial Services (XFS) interface specification - Release 3.0 - Part 8: Depository Device Class Interface

**Ref. No CWA 14050-8:2000 E**

# Table of Contents

## Foreword

This CWA is revision 3.0 of the XFS interface specification.

The move from an XFS 2.0 specification (CWA 13449) to a 3.0 specification has been prompted by a series of factors.

Initially, there has been a technical imperative to extend the scope of the existing specification of the XFS Manager to include new devices, such as the Card Embossing Unit.

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

Finally, it is also clear that our customers and the market are asking for an update to a specification, which is now over 2 years old. Increasing market acceptance and the need to meet this demand is driving the Workshop towards this release.

The clear direction of the CEN/ISSS XFS Workshop, therefore, is the delivery of a new Release 3.0 specification based on a C API. It will be delivered with the promise of the protection of technical investment for existing applications and the design to safeguard future developments.

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 2000-10-18. 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 3.0.

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 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 Class Interface - Programmer's Reference

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

Part 16: Application Programming Interface (API) - Service Provider Interface (SPI) - Migration from Version 2.0 (see CWA 13449) to Version 3.0 (this CWA) - Programmer's Reference

Part 17: Printer Device Class Interface - Migration from Version 2.0 (see CWA 13449) to Version 3.0 (this CWA) - Programmer's Reference

Part 18: Identification Card Device Class Interface - Migration from Version 2.0 (see CWA 13449) to Version 3.0 (this CWA) - Programmer's Reference

Part 19: Cash Dispenser Device Class Interface - Migration from Version 2.0 (see CWA 13449) to Version 3.0 (this CWA) - Programmer's Reference

Part 20: PIN Keypad Device Class Interface - Migration from Version 2.0 (see CWA 13449) to Version 3.0 (this CWA) - Programmer's Reference

Part 21: Depository Device Class Interface - Migration from Version 2.0 (see CWA 13449) to Version 3.0 (this CWA) - Programmer's Reference

Part 22: Text Terminal Unit Device Class Interface - Migration from Version 2.0 (see CWA 13449) to Version 3.0 (this CWA) - Programmer's Reference

Part 23: Sensors and Indicators Unit Device Class Interface - Migration from Version 2.0 (see CWA 13449) to Version 3.0 (this CWA) - Programmer's Reference

Part 24: Camera Device Class Interface - Migration from Version 2.0 (see CWA 13449) to Version 3.0 (this CWA) - Programmer's Reference

Part 25: Identification Card Device Class Interface - PC/SC Integration Guidelines

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.cenorm.be/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.

Revision History:

| | | |
|---|---|---|
| 1.0 | May 24, 1993 | Initial release of API and SPI specification |
| 1.11 | February 3, 1995 | Separation of specification into separate documents for API/SPI and service class definitions |
| 2.00 | November 11, 1996 | Update release encompassing the self-service environment |
| 3.00 | October 18, 2000 | Addition of Reset command which replaces the CLEAR_TRANSPORT command.<br>Addition of Threshold Event to ENTRY and RETRACT commands and new status values for container and envelope supply status.<br>UNICODE support<br>Addition of the events WFS_SRVE_DEP_ENVINSERTED and WFS_SRVE_DEP_MEDIADETECTED.<br>Created a References chapter.<br>For a detailed description see CWA 14050-21 Depository Migration from Version 2.00 to Version 3.00 Programmer’s Reference Revision 1.00, October 18, 2000 |

# 1. Introduction

## 1.1 Background to Release 3.0

The CEN XFS Workshop is a continuation of the Banking Solution Vendors Council workshop and maintains a technical commitment to the Win 32 API. However, the XFS Workshop has extended the franchise of multi vendor software by encouraging the participation of both banks and vendors to take part in the deliberations of the creation of an industry standard. This move towards opening the participation beyond the BSVC's original membership has been very succesful with a current membership level of more than 20 companies.

The fundamental aims of the XFS Workshop are to promote a clear and unambiguous specification for both service providers and application developers. This has been achieved to date by sub groups working electronically and quarterly meetings.

The move from an XFS 2.0 specification to a 3.0 specification has been prompted by a series of factors. Initially, there has been a technical imperative to extend the scope of the existing specification of the XFS Manager to include new devices, such as the Card Embossing Unit.

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

Finally, it is also clear that our customers and the market are asking for an update to a specification, which is now over 2 years old. Increasing market acceptance and the need to meet this demand is driving the Workshop towards this release.

The clear direction of the XFS Workshop, therefore, is the delivery of a new Release 3.0 specification based on a C API. It will be delivered with the promise of the protection of technical investment for existing applications and the design to safeguard future developments.

## 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 the Extensions for Financial Services is to standardize command codes and structures for the broadest variety of services. For example, using the **WFSExecute** function, the commands to read data from various services are as similar as possible to each other in their syntax and data structures.

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

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

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

- The requested capability is defined for the class of service providers by the XFS specification, the particular vendor implementation of that service does not support it, and the unsupported capability ***is*** considered to be fundamental to the service. In this case, a WFS_ERR_UNSUPP_COMMAND error is returned to the calling application. An example would be a request from an application to a cash dispenser to dispense coins; the service provider recognizes the command but, since the cash dispenser it is managing dispenses only notes, returns this error.
- The requested capability is ***not*** defined for the class of service providers by the XFS specification. In this case, a WFS_ERR_INVALID_COMMAND error is returned to the calling application.

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

# 2. Depository Unit

This specification describes the functionality of the services provided by the Depository (DEP) services under XFS, by defining the service-specific commands that can be issued, using the **WFSGetInfo, WFSAsyncGetInfo**, **WFSExecute** and **WFSAsyncExecute** functions.

A Depository is used for the acceptance and deposit of media into the device or terminal. There are two main types of depository: an envelope depository for the deposit of media in envelopes and a night safe depository for the deposit of bags containing bulk media.

An envelope depository accepts media, prints on the media and deposits the media into a holding container or bin. Some envelope depositories offer the capability to dispense an envelope to the customer at the start of a transaction. The customer takes this envelope, fills in the deposit media, possibly inscribes it and puts it into the deposit slot. The envelope is then accepted, printed and transported into a deposit container.

The envelope dispense mechanism may be part of the envelope depository device mechanism with the same entry/exit slot or it may be a separate mechanism with separate entry/exit slot.

Envelopes dispensed and not taken by the customer can be retracted back into the device. When the dispenser is a separate mechanism the envelope is retracted back into the dispenser container. When the dispenser is a common mechanism the envelope is retracted into the depository container.

A night safe depository normally only logs the deposit of a bag and does not print on the media.

## 3. References

1. XFS Application Programming Interface (API)/Service Provider Interface ( SPI), Programmer's Reference Revision 3.00, October 18, 2000

# 4. Info Commands

## 4.1 WFS_INF_DEP_STATUS

**Description** This command reports the full range of information available, including the information that is provided by the service provider.

**Input Param** None.

**Output Param**

```
LPWFSDEPSTATUS      lpStatus;

typedef struct _wfs_dep_status
   {
   WORD         fwDevice;
   WORD         fwDepContainer;
   WORD         fwDepTransport;
   WORD         fwEnvSupply;
   WORD         fwEnvDispenser;
   WORD         fwPrinter;
   WORD         fwToner;
   WORD         fwShutter;
   WORD         wNumOfDeposits;
   LPSTR        lpszExtra;
   } WFSDEPSTATUS, * LPWFSDEPSTATUS;
```

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

| Value | Meaning |
|---|---|
| WFS_DEP_DEVONLINE | The device is online (i.e., powered on and operable). |
| WFS_DEP_DEVOFFLINE | The device is off-line (e.g. the operator has taken the device offline by turning a switch or pulling out the device. |
| WFS_DEP_DEVPOWEROFF | The device is powered off or physically not connected. |
| WFS_DEP_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_DEP_DEVHWERROR | The device is inoperable due to a hardware error. The device is present but a hardware fault prevents it from being used. |
| WFS_DEP_DEVUSERERROR | The device is present but a person is preventing proper operation. The application should suspend the device operation or remove the device from service until the service provider generates a device state change event indicating the condition of the device has changed i.e. the error is removed (WFS_DEP_DEVONLINE) or a permanent error condition has occurred (WFS_DEP_DEVHWERROR). |
| WFS_DEP_DEVBUSY | The device is busy and not able to process an Execute command at this time. |

*fwDepContainer*
Specifies the state of the deposit container that contains the deposited envelopes or bags as one of the following flags:

| Value | Meaning |
|---|---|
| WFS_DEP_DEPOK | The deposit container is in a good state. |
| WFS_DEP_DEPHIGH | The deposit container is almost full (threshold). |
| WFS_DEP_DEPFULL | The deposit container is full. |
| WFS_DEP_DEPINOP | The deposit container is inoperable. |
| WFS_DEP_DEPMISSING | The deposit container is missing. |
| WFS_DEP_DEPUNKNOWN | Due to a hardware error or other condition, the state of the deposit container cannot be determined. |

| | |
|---|---|
| WFS_DEP_DEPNOTSUPP | The physical device is not able to determine the status of the deposit container. |

*fwDepTransport*
Specifies the state of the deposit transport mechanism that transports the envelope into the deposit container. Specified as one of the following flags:

| Value | Meaning |
|---|---|
| WFS_DEP_DEPOK | The deposit transport is in a good state. |
| WFS_DEP_DEPINOP | The deposit transport is inoperative due to a hardware failure or media jam. |
| WFS_DEP_DEPUNKNOWN | Due to a hardware error or other condition, the state of the deposit transport cannot be determined. |
| WFS_DEP_DEPNOTSUPP | The physical device has no deposit transport. |

*fwEnvSupply*
Specifies the state of the envelope supply unit as one of the following flags:

| Value | Meaning |
|---|---|
| WFS_DEP_ENVOK | The envelope supply unit is in a good state (and locked). |
| WFS_DEP_ENVLOW | The envelope supply unit is present but low. |
| WFS_DEP_ENVEMPTY | The envelope supply unit is present but empty. No envelopes can be dispensed. |
| WFS_DEP_ENVINOP | The envelope supply unit is in an inoperable state. No envelopes can be dispensed. |
| WFS_DEP_ENVMISSING | The envelope supply unit is missing. |
| WFS_DEP_ENVNOTSUPP | The physical device has no envelope supply. |
| WFS_DEP_ENVUNLOCKED | The envelope supply unit is unlocked. |
| WFS_DEP_ENVUNKNOWN | Due to a hardware error or other condition, the state of the envelope supply cannot be determined. |

*fwEnvDispenser*
Specifies the state of the envelope dispenser. Specified as one of the following flags.

| Value | Meaning |
|---|---|
| WFS_DEP_ENVOK | The envelope dispenser is present and in a good state. |
| WFS_DEP_ENVINOP | The envelope dispenser is present but in an inoperable state. No envelopes can be dispensed. |
| WFS_DEP_ENVUNKNOWN | Due to a hardware error or other condition, the state of the envelope dispenser cannot be determined. |
| WFS_DEP_ENVNOTSUPP | The physical device has no envelope dispenser. |

*fwPrinter*
Specifies the state of the printer. Specified as one of the following flags:

| Value | Meaning |
|---|---|
| WFS_DEP_PTROK | The printer is present and in a good state. |
| WFS_DEP_PTRINOP | The printer is inoperative. |
| WFS_DEP_PTRUNKNOWN | Due to a hardware error or other condition, the state of the printer cannot be determined. |
| WFS_DEP_PTRNOTSUPP | The physical device has no printer. |

*fwToner*
Specifies the state of the toner (or ink) for the printer. Specified as one of the following flags:

| Value | Meaning |
|---|---|
| WFS_DEP_TONERFULL | The toner cassette is full. |
| WFS_DEP_TONERLOW | The toner in the printer is low. |
| WFS_DEP_TONEROUT | The toner in the printer is empty. |
| WFS_DEP_TONERUNKNOWN | Due to a hardware error or other condition, the state of the toner for the printer cannot be determined. |
| WFS_DEP_TONERNOTSUPP | The physical device has no toner. |

*fwShutter*
Specifies the state of the shutter or door. Specified as one of the following flags:

| Value | Meaning |
|---|---|
| WFS_DEP_SHTCLOSED | The shutter is closed. |
| WFS_DEP_SHTOPEN | The shutter is open. |
| WFS_DEP_SHTJAMMED | The shutter is jammed. |
| WFS_DEP_SHTUNKNOWN | Due to a hardware error or other condition, the state of the shutter cannot be determined. |
| WFS_DEP_SHTNOTSUPP | The physical device has no shutter. |

*wNumOfDeposits*
Specifies the number of envelopes or bags in the deposit container. This value is persistent, i.e. maintained through power failures, opens, closes and system resets.

*lpszExtra*
Specifies 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 will be null-terminated, with the final string terminating with two null characters.

**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.

## 4.2 WFS_INF_DEP_CAPABILITIES

**Description** This command is used to retrieve the capabilities of the Depository.

**Input Param** None.

**Output Param**

```
LPWFSDEPCAPS        lpCaps;

typedef struct _wfs_dep_caps
   {
   WORD         wClass;
   WORD         fwType;
   WORD         fwEnvSupply;
   BOOL         bDepTransport;
   BOOL         bPrinter;
   BOOL         bToner;
   BOOL         bShutter;
   BOOL         bPrintOnRetracts;
   WORD         fwRetractEnvelope;
   WORD         wMaxNumChars;
   WORD         fwCharSupport;
   LPSTR        lpszExtra;
   } WFSDEPCAPS, * LPWFSDEPCAPS;
```

*wClass*
Specifies the logical service class, value is:
WFS_SERVICE_CLASS_DEP

*fwType*
Specifies the type of the depository device as a combination of the following flags:

| Value | Meaning |
|---|---|
| WFS_DEP_TYPEENVELOPE | Depository accepts envelopes. |
| WFS_DEP_TYPEBAGDROP | Depository accepts bags. |

*fwEnvSupply*
Defines what type of Envelope Supply Unit exists as one of the following flags:

| Value | Meaning |
|---|---|
| WFS_DEP_ENVMOTORIZED | Envelope Supply can dispense envelopes. |
| WFS_DEP_ENVMANUAL | Envelope Supply is manual and must be unlocked to allow envelopes to be taken. The Service Event, WFS_SRVE_DEP_ENVTAKEN, can not be sent and the Execute Command, WFS_CMD_DEP_RETRACT can not be supported. |
| WFS_DEP_ENVNONE | No Envelope Supply or Envelope Supply is manual and envelopes can be taken at any time. The Service Event, WFS_SRVE_DEP_ENVTAKEN, can not be sent. and the Execute Command, WFS_CMD_DEP_RETRACT can not be supported. |

*bDepTransport*
Specifies whether a deposit transport mechanism is available and is either TRUE or FALSE.

*bPrinter*
Specifies whether a printer is available and is either TRUE or FALSE.

*bToner*
Specifies whether the printer has a toner (or ink) cassette and is either TRUE or FALSE.

*bShutter*
Specifies whether a deposit transport shutter is available and is either TRUE or FALSE.

*bPrintOnRetracts*
Specifies whether the device can print the string specified in the lpszPrintData or lpszUNICODEPrintData field of the WFS_CMD_DEP_RETRACT command on retracted envelopes and is either TRUE or FALSE.

*.fwRetractEnvelope*
Specifies the ability of the envelope dispenser to retract envelopes as one of the following flags:

| Value | Meaning |
|---|---|
| WFS_DEP_NORETRACT | The envelope dispenser does not have the capability to retract envelopes. |
| WFS_DEP_RETRACTDEP | Retracted envelopes are put in the deposit container. |
| WFS_DEP_RETRACTDISP | Retracted envelopes are retracted back to the envelope dispenser. |

*wMaxNumChars*
Specifies the maximum number of characters that can be printed on the envelope.

*fwCharSupport*
One or more flags specifying the Character Sets supported by the service provider:

| Value | Meaning |
|---|---|
| WFS_DEP_ASCII | ASCII is supported for execute command data values. |
| WFS_DEP_UNICODE | UNICODE is supported for execute command data values. |

*lpszExtra*
Specifies 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 will be null-terminated, with the final string terminating with two null characters.

**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.

# 5. Execute Commands

## 5.1 WFS_CMD_DEP_ENTRY

**Description** This command starts the entry of an envelope and deposits it into the deposit container. If the envelope entered has an incorrect size and the deposit was not completed, the envelope is returned to the exit slot for removal by the customer, if the deposit device is capable of this operation (either hardware capability or hardware problems such as a jam may prohibit the envelope from being returned). A WFS_SRVE_DEP_ENVTAKEN is sent when the envelope is removed.

If a deposit takes place then this command will report a successful operation and any errors detected during the operation will be returned by the WFS_EXEE_DEP_DEPOSITERROR event. If the successful deposit causes the deposit bin to reach a high or full threshold, a WFS_USRE_DEP_DEPTHRESHOLD event will be sent.

**Input Param**

```
LPWFSDEPENVELOPE    lpEnvelope;
```

```
typedef struct _wfs_dep_envelope
   {
   LPSTR        lpszPrintData;
   LPWSTR       lpszUNICODEPrintData;
   } WFSDEPENVELOPE, * LPWFSDEPENVELOPE;
```

*lpszPrintData*
Specifies the data that will be printed on the envelope that is entered by the customer.

*lpszUNICODEPrintData*
Specifies the UNICODE data that will be printed on the envelope that is entered by the customer.

The *lpszUNICODEPrintData* field should only be used if the service provider supports UNICODE. The *lpszPrintData* and *lpszUNICODEPrintData* fields are mutually exclusive.

**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_DEP_DEPFULL | The deposit container is full. |
| WFS_ERR_DEP_DEPJAMMED | An envelope jam occurred in the deposit transport between the entry slot and the deposit container. |
| WFS_ERR_DEP_ENVSIZE | The envelope entered has an incorrect size. |
| WFS_ERR_DEP_PTRFAIL | The printer failed. |
| WFS_ERR_DEP_SHTNOTCLOSED | The shutter failed to close. |
| WFS_ERR_DEP_SHTNOTOPENED | The shutter failed to open. |
| WFS_ERR_DEP_CONTMISSING | The deposit container is not present. |
| WFS_ERR_DEP_DEPUNKNOWN | The result of the deposit is not known. |
| WFS_ERR_DEP_CHARSETNOTSUPP | Character set(s) supported by service provider is inconsistent with use of lpszPrintData or lpszUNICODEPrintData fields. |
| WFS_ERR_DEP_TONEROUT | Toner or ink supply is empty or printing contrast with ribbon is not sufficient. This error can only occur when a print string was passed in the input parameter. |

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

| Value | Meaning |
|---|---|
| WFS_SRVE_DEP_ENVTAKEN | The envelope has been taken by the user. |
| WFS_EXEE_DEP_ENVDEPOSITED | The envelope has been deposited in the deposit container. |

| | |
|---|---|
| WFS_EXEE_DEP_DEPOSITERROR | An error occurred during the deposit operation. |
| WFS_USRE_DEP_DEPTHRESHOLD | This user event is used to specify that the state of the deposit container reached a threshold. |
| WFS_USRE_DEP_TONERTHRESHOLD | This user event is used to specify that the state of the toner supply reached a threshold. |
| WFS_SRVE_DEP_ENVINSERTED | An envelope has been inserted by the user. |

**Comments** If the data specified in lpszPrintData or lpszUNICODEPrintData is longer than the maximum allowed characters, the error code WFS_ERR_INVALID_DATA will be returned..

## 5.2 WFS_CMD_DEP_DISPENSE

**Description** This command is used to dispense an envelope from the envelope supply. This command will either action the dispensing of an envelope from the envelope supply or will unlock the envelope supply for manual access.

**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_DEP_ENVEMPTY | There is no envelope in the envelope unit. |
| WFS_ERR_DEP_ENVJAMMED | An envelope jam occurred in the dispenser transport between the envelope supply and the output slot. |
| WFS_ERR_DEP_SHTNOTOPENED | The shutter failed to open. |

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

| Value | Meaning |
|---|---|
| WFS_SRVE_DEP_ENVTAKEN | The envelope has been taken by the user. |
| WFS_USRE_DEP_ENVTHRESHOLD | This user event is used to specify that the state of the envelope supply reached a threshold. |

**Comments** None.

## 5.3 WFS_CMD_DEP_RETRACT

**Description** This command is used to retract an envelope that was not taken by a customer after an envelope dispense operation. The given string is printed on the envelope and the envelope is retracted into the deposit container or back to the envelope dispenser, depending on the capabilities of the physical device. If a retract to the deposit bin causes the deposit bin to reach a high or full threshold, a WFS_USRE_DEP_DEPTHRESHOLD event will be sent.

This command will only return with an error code if the retract has not taken place. The error code will then describe the reason for the failure.

**Input Param**

```
LPWFSDEPENVELOPE    lpEnvelope;

typedef struct _wfs_dep_envelope
    {
    LPSTR         lpszPrintData;
    LPSTR         lpszUNICODEPrintData;
    } WFSDEPENVELOPE, * LPWFSDEPENVELOPE;
```

*lpszPrintData*
Specifies the data that will be printed on the envelope that is retracted.

*lpszUNICODEPrintData*
Specifies the UNICODE data that will be printed on the envelope that is retracted.

The *lpszUNICODEPrintData* field should only be used if the service provider supports UNICODE. The *lpszPrintData* and *lpszUNICODEPrintData* fields are mutually exclusive.

**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_DEP_DEPFULL | The deposit container is full. |
| WFS_ERR_DEP_DEPJAMMED | An envelope jam occurred in the deposit transport between the entry slot and the deposit container (may only occur with hardware that retracts to the deposit container). |
| WFS_ERR_DEP_ENVJAMMED | An envelope jam occurred between the entry slot and the envelope container (may only occur with hardware that retracts to the envelope container). |
| WFS_ERR_DEP_NOENV | No envelope to retract. |
| WFS_ERR_DEP_PTRFAIL | The printer failed. |
| WFS_ERR_DEP_SHTNOTCLOSED | The shutter failed to close. |
| WFS_ERR_DEP_CONTMISSING | The deposit container is not present. |
| WFS_ERR_DEP_CHARSETNOTSUPP | Character set(s) supported by service provider is inconsistent with use of lpszPrintData or lpszUNICODEPrintData fields. |
| WFS_ERR_DEP_TONEROUT | Toner or ink supply is empty or printing contrast with ribbon is not sufficient. |

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

| Value | Meaning |
|---|---|
| WFS_USRE_DEP_DEPTHRESHOLD | This user event is used to specify that the state of the deposit container reached a threshold. |
| WFS_USRE_DEP_TONERTHRESHOLD | This user event is used to specify that the state of the toner supply reached a threshold. |

**Comments** If the data specified in lpszPrintData or lpszUNICODEPrintData is longer than the maximum allowed characters, the error code WFS_ERR_INVALID_DATA will be returned.

## 5.4 WFS_CMD_DEP_RESET_COUNT

**Description** This command is used to reset the present value for number of envelopes/bags in the deposit container to zero.

**Input Param** None.

**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_DEP_DEPTHRESHOLD | This user event is used to specify that the state of the deposit container reached a threshold. |

**Comments** None.

## 5.5 WFS_CMD_DEP_RESET

**Description** Sends a service reset to the service provider. The service provider may reset the deposit device and also the envelope dispenser, if possible. Any media found in the device can be either captured or completely ejected (depending on hardware). If a capture into the deposit bin causes the deposit bin to reach a high or full threshold, a WFS_USRE_DEP_DEPTHRESHOLD event will be sent. If the WFS_CMD_DEP_RESET command is requested to eject the media and the hardware is not capable of this operation either due to hardware capability or hardware error such as a jam, the service provider will retract the media in order to attempt to make the device operational. The WFS_SRVE_DEP_MEDIADETECTED event will indicate the position of the detected media following completion of the command. If the input parameter to the WFS_CMD_DEP_RESET command is NULL, the service provider will go through default actions to clear the deposit transport. The WFS_SRVE_DEP_MEDIADETECTED event will indicate the position of any detected media following completion of the command. The envelope dispenser will go through the most effective means to clear any jammed media.

**Input Param** LPDWORD lpdwDepMediaControl;

Specifies the action that should be done if deposited media is detected during the reset operation, as one of the following values:

| Value | Meaning |
|---|---|
| WFS_DEP_CTRLEJECT | Any media detected in the device should be completed ejected (depending on the hardware) |
| WFS_DEP_CTRLRETRACT | Any media detected in the device should be deposited into the deposit container during the reset operation. |

If lpdwDepMediaControl is set to NULL, the service provider will go through default actions to clear the deposit transport.

**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_DEP_DEPFULL | The deposit container is full. |
| WFS_ERR_DEP_DEPJAMMED | An envelope jam occurred in the deposit transport. |
| WFS_ERR_DEP_ENVJAMMED | An envelope jam occurred in the dispenser transport between the envelope supply and the output slot. |
| WFS_ERR_DEP_SHTNOTOPENED | The shutter failed to open. |
| WFS_ERR_DEP_SHTNOTCLOSED | The shutter failed to close. |
| WFS_ERR_DEP_CONTMISSING | The deposit container is not present. |

**Events** In addition to the generic events defined in [Ref. 1], the following events may be generated by this command, if the appropriate situation occurs and the device service has the capability to detect the situation:

| Value | Meaning |
|---|---|
| WFS_SRVE_DEP_ENVTAKEN | The envelope has been taken by the user. |
| WFS_USRE_DEP_DEPTHRESHOLD | This user event is used to specify that the state of the deposit container reached a threshold |
| WFS_SRVE_DEP_MEDIADETECTED | Media is detected in the device during a reset operation. |

**Comments** This command is used by an application control program to cause a device to reset itself to a known good condition. Persistent values may change, but will not be reset as a result of this command (i.e. If an envelope is captured, the ‘wNumOfDeposits’ value in the WFSDEPSTATUS structure will be incremented, but never reset to 0).

# 6. Events

## 6.1 WFS_SRVE_DEP_ENVTAKEN

**Description** This service event is used to specify that the envelope has been taken by the customer.

**Event Param** None.

**Comments** None.

## 6.2 WFS_EXEE_DEP_ENVDEPOSITED

**Description** This execute event is used to specify that the envelope has been deposited in the deposit container.

**Event Param** None.

**Comments** None.

## 6.3 WFS_EXEE_DEP_DEPOSITERROR

**Description** This execute event is used to specify that an error occurred during the deposit operation. For every error that occurred a single execute event is generated.

**Event Param**

```
LPLONG        lplError;
```

For a list of possible error conditions see the description of the WFS_CMD_DEP_ENTRY command.

**Comments** None.

## 6.4 WFS_USRE_DEP_DEPTHRESHOLD

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

**Event Param**

```
LPWORD lpwDepositThreshold;
```

Specified as one of the following flags:

| Value | Meaning |
|---|---|
| WFS_DEP_DEPOK | The deposit container is in a good state. |
| WFS_DEP_DEPHIGH | The deposit container is almost full (threshold). |
| WFS_DEP_DEPFULL | The deposit container is full. |

**Comments** None.

## 6.5 WFS_USRE_DEP_TONERTHRESHOLD

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

**Event Param**

```
LPWORD lpwTonerThreshold;
```

Specified as one of the following flags:

| Value | Meaning |
|---|---|
| WFS_DEP_TONERFULL | The toner or ink supply is full or the ribbon is OK. |
| WFS_DEP_TONERLOW | The toner or ink supply is low or the print contrast with a ribbon is weak. |
| WFS_DEP_TONEROUT | The toner or ink supply is empty or the print contrast with a ribbon is not sufficient any more. |

**Comments** None.

## 6.6 WFS_USRE_DEP_ENVTHRESHOLD

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

**Event Param** `LPWORD lpwEnvelopeThreshold;`

Specified as one of the following flags:

| Value | Meaning |
|---|---|
| WFS_DEP_ENVOK | The envelope supply is present and in a good state. |
| WFS_DEP_ENVLOW | The envelope supply is present but low. |
| WFS_DEP_ENVEMPTY | The envelope supply is present but empty. No envelopes can be dispensed. |

**Comments** None.

## 6.7 WFS_SRVE_DEP_CONTINSERTED

**Description** This service event is used to specify that the deposit container has been reinserted into the device.

**Event Param** None.

**Comments** None.

## 6.8 WFS_SRVE_DEP_CONTREMOVED

**Description** This service event is used to specify that the deposit container has been removed from the device.

**Event Param** None.

**Comments** None.

## 6.9 WFS_SRVE_DEP_ENVINSERTED

**Description** This service event is used to specify that an envelope has been inserted by the customer.

**Event Param** None.

**Comments** None.

## 6.10 WFS_SRVE_DEP_MEDIADETECTED

**Description** This event is generated when media is detected in the device during a reset operation. The media may be detected as a result of the reset operation on the envelope dispenser, the envelope depositor, or both.

**Event Param** `LPWFSDEPMEDIADETECTED            lpMediaDetected;`

```
typedef struct _wfs_dep_media_detected
   {
   WORD         wDispenseMedia;
   WORD         wDepositMedia;
   } WFSDEPMEDIADETECTED, * LPWFSDEPMEDIADETECTED;
```

*wDispenseMedia*
Specifies the dispensed envelope position after the reset operation, as one of the following values:

| Value | Meaning |
|---|---|
| WFS_DEP_NOMEDIA | No dispensed media was detected during the reset operation. |

| | |
|---|---|
| WFS_DEP_MEDIARETRACTED | The media was retracted into the deposit container during the reset operation. |
| WFS_DEP_MEDIADISPENSER | The media was retracted into the envelope dispenser during the reset operation. |
| WFS_DEP_MEDIAEJECTED | The media is in the exit slot. |
| WFS_DEP_MEDIAJAMMED | The media is jammed in the device. |
| WFS_DEP_MEDIAUNKNOWN | The media is in an unknown position. |

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

| Value | Meaning |
|---|---|
| WFS_DEP_NOMEDIA | No deposited media was detected during the reset operation. |
| WFS_DEP_MEDIARETRACTED | The media was retracted into the deposit container during the reset operation. |
| WFS_DEP_MEDIAEJECTED | The media is in the exit slot. |
| WFS_DEP_MEDIAJAMMED | The media is jammed in the device. |
| WFS_DEP_MEDIAUNKNOWN | The media is in an unknown position. |

**Comments** None.

# 7. C-Header file

```
/******************************************************************************
*                                                                            *
* xfsdep.h   XFS - Depository (DEP) definitions                              *
*                                                                            *
*            Version 3.00  (10/18/00)                                        *
*                                                                            *
******************************************************************************/

#ifndef __INC_XFSDEP__H
#define __INC_XFSDEP__H

#ifdef __cplusplus
extern "C" {
#endif

#include <xfsapi.h>

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


/* values of WFSDEPCAPS.wClass */

#define    WFS_SERVICE_CLASS_DEP             (6)
#define    WFS_SERVICE_CLASS_VERSION_DEP     (0x0003) /* Version 3.00 */
#define    WFS_SERVICE_CLASS_NAME_DEP        "DEP"

#define    DEP_SERVICE_OFFSET           (WFS_SERVICE_CLASS_DEP * 100)

/* DEP Info Commands */

#define    WFS_INF_DEP_STATUS           (DEP_SERVICE_OFFSET + 1)
#define    WFS_INF_DEP_CAPABILITIES     (DEP_SERVICE_OFFSET + 2)

/* DEP Execute Commands */

#define    WFS_CMD_DEP_ENTRY            (DEP_SERVICE_OFFSET + 1)
#define    WFS_CMD_DEP_DISPENSE         (DEP_SERVICE_OFFSET + 2)
#define    WFS_CMD_DEP_RETRACT          (DEP_SERVICE_OFFSET + 3)
#define    WFS_CMD_DEP_CLEAR_TRANSPORT  (DEP_SERVICE_OFFSET + 4)
#define    WFS_CMD_DEP_RESET_COUNT      (DEP_SERVICE_OFFSET + 5)
#define    WFS_CMD_DEP_RESET            (DEP_SERVICE_OFFSET + 6)


/* DEP Messages */

#define    WFS_SRVE_DEP_ENVTAKEN        (DEP_SERVICE_OFFSET + 1)
#define    WFS_EXEE_DEP_ENVDEPOSITED    (DEP_SERVICE_OFFSET + 2)
#define    WFS_EXEE_DEP_DEPOSITERROR    (DEP_SERVICE_OFFSET + 3)
#define    WFS_USRE_DEP_DEPTHRESHOLD    (DEP_SERVICE_OFFSET + 4)
#define    WFS_USRE_DEP_TONERTHRESHOLD  (DEP_SERVICE_OFFSET + 5)
#define    WFS_USRE_DEP_ENVTHRESHOLD    (DEP_SERVICE_OFFSET + 6)
#define    WFS_SRVE_DEP_CONTINSERTED    (DEP_SERVICE_OFFSET + 7)
#define    WFS_SRVE_DEP_CONTREMOVED     (DEP_SERVICE_OFFSET + 8)
#define    WFS_SRVE_DEP_ENVINSERTED     (DEP_SERVICE_OFFSET + 9)
#define    WFS_SRVE_DEP_MEDIADETECTED   (DEP_SERVICE_OFFSET + 10)

/* values of WFSDEPSTATUS.fwDevice */
#define    WFS_DEP_DEVONLINE            WFS_STAT_DEVONLINE
#define    WFS_DEP_DEVOFFLINE           WFS_STAT_DEVOFFLINE
#define    WFS_DEP_DEVPOWEROFF          WFS_STAT_DEVPOWEROFF
#define    WFS_DEP_DEVBUSY              WFS_STAT_DEVBUSY
#define    WFS_DEP_DEVNODEVICE          WFS_STAT_DEVNODEVICE
#define    WFS_DEP_DEVHWERROR           WFS_STAT_DEVHWERROR
#define    WFS_DEP_DEVUSERERROR         WFS_STAT_DEVUSERERROR

/* values of WFSDEPSTATUS.fwDepContainer, fwDepTransport */

#define    WFS_DEP_DEPOK                (0)
#define    WFS_DEP_DEPHIGH              (1)
#define    WFS_DEP_DEPFULL              (2)
```

```
#define    WFS_DEP_DEPINOP              (3)
#define    WFS_DEP_DEPMISSING           (4)
#define    WFS_DEP_DEPUNKNOWN           (5)
#define    WFS_DEP_DEPNOTSUPP           (6)

/* values of WFSDEPSTATUS.fwEnvSupply, fwEnvDispenser */

#define    WFS_DEP_ENVOK                (0)
#define    WFS_DEP_ENVLOW               (1)
#define    WFS_DEP_ENVEMPTY             (2)
#define    WFS_DEP_ENVINOP              (3)
#define    WFS_DEP_ENVMISSING           (4)
#define    WFS_DEP_ENVUNKNOWN           (5)
#define    WFS_DEP_ENVNOTSUPP           (6)
#define    WFS_DEP_ENVUNLOCKED          (7)

/* values of WFSDEPSTATUS.fwPrinter */

#define    WFS_DEP_PTROK                (0)
#define    WFS_DEP_PTRINOP              (1)
#define    WFS_DEP_PTRUNKNOWN           (2)
#define    WFS_DEP_PTRNOTSUPP           (3)

/* values of WFSDEPSTATUS.fwToner */

#define    WFS_DEP_TONERFULL            (0)
#define    WFS_DEP_TONERLOW             (1)
#define    WFS_DEP_TONEROUT             (2)
#define    WFS_DEP_TONERUNKNOWN         (3)
#define    WFS_DEP_TONERNOTSUPP         (4)

/* values of WFSDEPSTATUS.fwShutter */

#define    WFS_DEP_SHTCLOSED            (0)
#define    WFS_DEP_SHTOPEN              (1)
#define    WFS_DEP_SHTJAMMED            (2)
#define    WFS_DEP_SHTUNKNOWN           (3)
#define    WFS_DEP_SHTNOTSUPP           (4)


/* values of WFSDEPCAPS.fwType */

#define    WFS_DEP_TYPEENVELOPE         (0x0001)
#define    WFS_DEP_TYPEBAGDROP          (0x0002)

/* values of WFSDEPCAPS.fwEnvSupply */

#define    WFS_DEP_ENVMOTORIZED         (1)
#define    WFS_DEP_ENVMANUAL            (2)
#define    WFS_DEP_ENVNONE              (3)

/* values of WFSDEPCAPS.fwRetractEnvelope */

#define    WFS_DEP_NORETRACT            (1)
#define    WFS_DEP_RETRACTDEP           (2)
#define    WFS_DEP_RETRACTDISP          (3)

/* values of WFSDEPCAPS.fwCharSupport, WFSDEPENVELOPE.fwCharSupport */

#define    WFS_DEP_ASCII                (0x0001)
#define    WFS_DEP_UNICODE              (0x0002)

/* values of dwDepMediaControl  */

#define     WFS_DEP_CTRLEJECT           (0x0001)
#define     WFS_DEP_CTRLRETRACT         (0x0002)


/* values of WFSDEPMEDIADETECTED.wDispenseMedia, wDepositMedia */

#define     WFS_DEP_NOMEDIA             (1)
#define     WFS_DEP_MEDIARETRACTED      (2)
#define     WFS_DEP_MEDIADISPENSER      (3)
#define     WFS_DEP_MEDIAEJECTED        (4)
```

```
#define     WFS_DEP_MEDIAJAMMED         (5)
#define     WFS_DEP_MEDIAUNKNOWN        (6)

#define    WFS_ERR_DEP_DEPFULL           (-(DEP_SERVICE_OFFSET + 0))
#define    WFS_ERR_DEP_DEPJAMMED         (-(DEP_SERVICE_OFFSET + 1))
#define    WFS_ERR_DEP_ENVEMPTY          (-(DEP_SERVICE_OFFSET + 2))
#define    WFS_ERR_DEP_ENVJAMMED         (-(DEP_SERVICE_OFFSET + 3))
#define    WFS_ERR_DEP_ENVSIZE           (-(DEP_SERVICE_OFFSET + 4))
#define    WFS_ERR_DEP_NOENV             (-(DEP_SERVICE_OFFSET + 5))
#define    WFS_ERR_DEP_PTRFAIL           (-(DEP_SERVICE_OFFSET + 6))
#define    WFS_ERR_DEP_SHTNOTCLOSED      (-(DEP_SERVICE_OFFSET + 7))
#define    WFS_ERR_DEP_SHTNOTOPENED      (-(DEP_SERVICE_OFFSET + 8))
#define    WFS_ERR_DEP_CONTMISSING       (-(DEP_SERVICE_OFFSET + 9))
#define    WFS_ERR_DEP_DEPUNKNOWN        (-(DEP_SERVICE_OFFSET + 10))
#define    WFS_ERR_DEP_CHARSETNOTSUPP    (-(DEP_SERVICE_OFFSET + 11))
#define    WFS_ERR_DEP_TONEROUT          (-(DEP_SERVICE_OFFSET + 12))


/*=================================================================*/
/*    DEP Info Command Structures and variables                    */
/*=================================================================*/

typedef struct _wfs_dep_status
{
    WORD        fwDevice;
    WORD        fwDepContainer;
    WORD        fwDepTransport;
    WORD        fwEnvSupply;
    WORD        fwEnvDispenser;
    WORD        fwPrinter;
    WORD        fwToner;
    WORD        fwShutter;
    WORD        wNumOfDeposits;
    LPSTR       lpszExtra;
} WFSDEPSTATUS, * LPWFSDEPSTATUS;


typedef struct _wfs_dep_caps
{
    WORD        wClass;
    WORD        fwType;
    WORD        fwEnvSupply;
    BOOL        bDepTransport;
    BOOL        bPrinter;
    BOOL        bToner;
    BOOL        bShutter;
    BOOL        bPrintOnRetracts;
    WORD        fwRetractEnvelope;
    WORD        wMaxNumChars;
    WORD        fwCharSupport;
    LPSTR       lpszExtra;
} WFSDEPCAPS, * LPWFSDEPCAPS;

/*=================================================================*/
/*    DEP Execute Command Structures                   */
/*=================================================================*/

typedef struct _wfs_dep_envelope
{
    LPSTR       lpszPrintData;
    LPWSTR      lpszUNICODEPrintData;
} WFSDEPENVELOPE, * LPWFSDEPENVELOPE;

/*=================================================================*/
/*    DEP Message Structures                       */
/*=================================================================*/

typedef struct _wfs_dep_media_detected
{
    WORD        wDispenseMedia;
    WORD        wDepositMedia;
} WFSDEPMEDIADETECTED, * LPWFSDEPMEDIADETECTED;
```

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

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

#endif  /* __INC_XFSDEP__H */
```