CEN

WORKSHOP

AGREEMENT

CWA 13449-8

December 1998

ICS 35.200;35.240.40

English version

Extensions for Financial Services (XFS) interface specification - Part 8: Depository Device Class Interface - Programmer's Interface

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

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

Ref. No. CWA 13449-8:1998 E

Contents

Foreword

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

This specification was originally developed by the Banking Solutions Vendor Council (BSVC), and is endorsed by the CEN/ISSS Workshop on XFS. This Workshop gathers both suppliers (among others the BSVC members) as well as banks and other financial service companies. A list of companies participating in this Workshop and in support of this CWA is available from the CEN/ISSS Secretariat.

The specification is continuously reviewed and commented in the CEN/ISSS Workshop on XFS. It is therefore expected that an update of the specification will be published in due time as a CWA, superseding this revision 2.00.

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

0. Introduction

This is part 8 of the multi-part CWA 13449, describing Release 2.0 of the XFS interface specification.

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

Part 1: Application Programming Interface (API) - Service Provider Interface (SPI); Programmer's Reference
Part 2: Service Classes Definition; Programmer's Reference
Part 3: Printer Device Class Interface - Programmer's Reference
Part 4: Identification Card Device Class Interface - Programmer's Reference
Part 5: Cash Dispenser Device Class Interface - Programmer's Reference
Part 6: PIN Keypad Device Class Interface - Programmer's Reference
Part 7: Check Reader/Scanner Device Class Interface - Programmer's Reference
Part 8: Depository Device Class Interface - Programmer's Reference
Part 9: Text Terminal Unit Device Class Interface - Programmer's Reference
Part 10: Sensors and Indicators Unit Device Class Interface - Programmer's Reference
Part 11: Vendor Dependent Mode Device Class Interface - Programmer's Reference
Part 12: Camera Device Class Interface - Programmer's Reference

In addition to these Programmer's Reference specifications, the reader of this CWA is also referred to a complementary document, called Release Notes. The Release Notes contain clarifications and explanations on the CWA specifications, which are not requiring functional changes. The current version of the Release Notes is available from the CEN/ISSS Secretariat (contact isss@cenorm.be or download from http://www.cenorm.be/isss/ Workshop/XFS/release-notes.htm).

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

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

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

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

Revision History:

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

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

1. XFS Service-Specific Programming

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

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

In general, the specific command set for a service class is defined as the union of the 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_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. Info Commands

3.1 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 on-line. The device is present and operational (i.e. not busy processing a request and does not have a hardware error).
WFS_DEP_DEVOFFLINE	The device is off-line. The device is present and powered on but it is not operational (e.g. a switch may have been used to change it to an off-line state).
WFS_DEP_DEVPOWEROFF	The device is powered off. The device is present, but is currently powered off.
WFS_DEP_DEVBUSY	The device is busy processing a request. The device is present and an EXECUTE request is currently being processed.
WFS_DEP_DEVNODEVICE	There is no device connected.
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).

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.

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_ENVUNLOCKED	The envelope supply unit is unlocked

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_TONOK	The toner cassette is full.
WFS_DEP_TONLOW	The toner in the printer is low.
WFS_DEP_TONEMPTY	The toner in the printer is empty.
WFS_DEP_TONUNKNOWN	Due to a hardware error or other condition, the state of the toner for the printer cannot be determined.
WFS_DEP_TONNOTSUPP	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 opened.
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 There are no additional error codes generated by this command.

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

3.2 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;
   BOOL         bRetractToDeposit;
   WORD         wMaxNumChars;
   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 on retracted envelopes and is either TRUE or FALSE.

bRetractToDeposit
Specifies whether retracted envelopes are put in the deposit container and is either TRUE or FALSE. If TRUE, envelopes are retracted back to the deposit container. If FALSE, envelopes are retracted back to the envelope dispenser.

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

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 There are no additional error codes generated by this command.

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

4. Execute Commands

4.1 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. 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.

Input Param

```
LPWFSDEPENVELOPE    lpEnvelope;

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

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

Output Param None.

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

Value	Meaning
WFS_ERR_DEP_ENVJAMMED	An envelope jam occurred in the deposit transport.
WFS_ERR_DEP_DEPFULL	The deposit container is full.
WFS_ERR_DEP_CONTMISSING	The deposit container is not present.
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_DEPUNKNOWN	The result of the deposit is not known. This error code is only returned by the WFS_EXEE_DEP_DEPOSITERROR event.

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

Value	Meaning
WFS_EXEE_DEP_ENVDEPOSITED	The envelope has been deposited in the deposit container.
WFS_EXEE_DEP_DEPOSITERROR	An error occured during the deposit operation.
WFS_SRVE_DEP_ENVTAKEN	The envelope has been taken by the user.

Comments None.

4.2 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 The following additional 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.

WFS_ERR_DEP_SHTNOTOPENED	The shutter failed to open.

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

Value	Meaning
WFS_SRVE_DEP_ENVTAKEN	The envelope has been taken by the user.

Comments None.

4.3 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.

Input Param

```
LPWFSDEPENVELOPE    lpEnvelope;

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

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

Output Param None.

Error Codes The following additional 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_CONTMISSING	The deposit container is not present.
WFS_ERR_DEP_ENVJAMMED	An envelope jam occurred
WFS_ERR_DEP_NOENV	No envelope to retract.
WFS_ERR_DEP_PTRFAIL	The printer failed.
WFS_ERR_DEP_SHTNOTCLOSED	The shutter failed to close.

Events There are no additional events generated by this command.

Comments None.

4.4 WFS_CMD_DEP_CLEAR_TRANSPORT

Description This command is used to clear the envelope deposit transport from any envelopes or items left in the entry slot of the device. The envelopes can be either captured or completely ejected.

Input Param None.

Output Param None.

Error Codes The following additional 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_CONTMISSING	The deposit container is not present.
WFS_ERR_DEP_SHTNOTCLOSED	The shutter failed to close.

Events There are no additional events generated by this command.

Comments None.

4.5 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	There are no additional error codes returned by this command.
Events	There are no additional events generated by this command.
Comments	None.

5. Events

5.1 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.

5.2 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.

5.3 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.

5.4 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_DEPHIGH	The deposit container is almost full (threshold).
WFS_DEP_DEPFULL	The deposit container is full.

Comments None.

5.5 WFS_USRE_DEP_TONERTHRESHOLD

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

Event Param

```
LPWORD lpwTonerThreshold;
```

Specified as one of the following flags:

Value	Meaning
WFS_DEP_TONLOW	The toner (or ink) in the printer is low.
WFS_DEP_TONEMPTY	The toner (or ink) in the printer is empty.

Comments None.

5.6 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_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.

5.7 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.

5.8 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. C-Header file

```
/******************************************************************************
*                                                                             *
* xfsdep.h   XFS - Depository (DEP) definitions                               *
*                                                                             *
*            Version 2.00  (11/11/96)                                         *
*                                                                             *
******************************************************************************/

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

/* 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)

/* 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_TONOK                (0)
#define    WFS_DEP_TONLOW               (1)
#define    WFS_DEP_TONEMPTY             (2)
#define    WFS_DEP_TONUNKNOWN           (3)
#define    WFS_DEP_TONNOTSUPP           (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         (1)
#define    WFS_DEP_TYPEBAGDROP          (2)

/* values of WFSDEPCAPS.fwEnvSupply */

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




#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))


/*=================================================================*/
/*    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;
    BOOL        bRetractToDeposit;
    WORD        wMaxNumChars;
    LPSTR       lpszExtra;
} WFSDEPCAPS, * LPWFSDEPCAPS;

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

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

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


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

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

#endif  /* __INC_XFSDEP__H */
```